ORACLE DATABASE 12c

JDBC と UCP を使用した、Java アプリケーションの接続管理戦略

Oracle Database 12c

Oracle ホワイト・ペーパー 2016 年 6 月

ORACLE

概要

アーキテクト、アプリケーション開発者、および DBA は、RDBMS、オペレーティング・システム、Java アプリケーションをチューニングすることで、最大限のシステム・スループットを達成するように努めています。データベース接続は、アプリケーションのパフォーマンス、スケーラビリティ、可用性、セキュリティ、および管理性に重要な役割を果たします。ほとんどの場合、短期間での設計決定、不適切なデータベース構成、不十分なチューニング、データベース機能に対する不十分な理解のために、アプリケーションは最適には実行されていません。

Oracle データベースと Oracle JDBC ドライバおよび Oracle Universal Connection Pool（UCP）の組合せは、パフォーマンス、スケーラビリティ、可用性、セキュリティ、および管理性のサービス品質を向上するための、さまざまな接続管理戦略を提供します。これらの戦略は、適切な接続記述子とプロパティの設定、適切な接続プールの選択、アプリケーション動作に合わせたプールのチューニング、接続アフィニティの使用、接続ラベル付けの使用、計画メンテナンスをスムーズに実施して計画外停止を隠すための高可用性機能（アプリケーション継続性（AC）やトランザクション・ガード（TG）など）の利用、セキュリティ機能と管理機能の使用などで構成されます。このホワイト・ペーパーでは、単一インスタンスのデータベース、Oracle Real Application Clusters（Oracle RAC）、Oracle Multitenant、Oracle Data Guard（Oracle DG）、Oracle Active Data Guard（Oracle ADG）、Oracle Global Data Services（Oracle GDS）などの個々のデータベース構成に基づいて推奨事項を示しています。

このホワイト・ペーパーは、パフォーマンス、スケーラビリティ、可用性、セキュリティ、および管理性を中心に構成されています。各セクションで、それぞれの Oracle Database 12c 構成に関連する推奨事項を示しています。

それぞれの構成の接続管理コード・サンプル [1]は、Oracle Technology Network（OTN）および GITHUB でダウンロードできます。

[1] OTN @ http://www.oracle.com/technetwork/database/features/jdbc/default-2345085.htm、または GITHUB @ http://github.com/oracle/jdbc-ucp

JDBCとUCPの概要

Oracle JDBC ドライバは最新の JDBC 仕様で実装され、この仕様に準拠します。Java アプリケーションのクラス・パスに **ojdbc7.jar**（JDK 7 および JDK 8 の場合）または **ojdbc6.jar**（JDK 6 の場合）を設定している必要があります。詳しくは、『Oracle Database JDBC 開発者ガイド』[2]を参照してください。次のコード・サンプルは、JDBC 接続を取得する方法を示しています。

```
//Use connection descriptors
final static String DB_URL=
"jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(HOST=myhost)(PORT=1521)(PROTOCOL=tcp ))
(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))";
final static String DB_USER = "hr";
final static String DB_PASSWORD = sukuriputo "hr";
//Set connection level properties
Properties connProps = new Properties();
connProps.put(OracleConnection.CONNECTION_PROPERTY_USER_NAME,DB_USER);
connProps.put(OracleConnection.CONNECTION_PROPERTY_PASSWORD,DB_PASSWORD);

OracleDataSource ods = new OracleDataSource();
ods.setURL(DB_URL);
ods.setConnectionProperties(connProps);
OracleConnection connection = (OracleConnection) ods.getConnection();
```

Oracle Universal Connection Pool（UCP）は、すべての Oracle データベース構成と緊密に統合される機能豊富な Java 接続プールであり、高可用性、スケーラビリティ、ワークロード分散を実現します。また、UCP は、オラクル以外のデータベースにオラクル以外の JDBC ドライバで使用することも可能です。UCP を使用するには、**ojdbc7.jar**（JDK 7 および JDK8 の場合）または **ojdbc6.jar**（JDK 6 の場合）に加えて、Java アプリケーションまたはコンテナのクラス・パスに **ucp.jar**[3]を設定する必要があります。多くのサード・パーティ接続プールで、基本機能が提供されています。UCP は、Oracle Real Application Clusters、Active Data Guard、および Global Data Services と緊密に統合されているため、他の接続プールよりも優れています。詳しくは、『Oracle Universal Connection Pool for JDBC 開発者ガイド』[4]を参照してください。

次のコード・フラグメントは、UCP で接続を取得する方法を示しています。

```
// Get the PoolDataSource for UCP
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
//Set the connection factory first before all other properties
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL(DB_URL);
pds.setUser(DB_USER);
```

[2] 『Oracle Database JDBC 開発者ガイド』@https://docs.oracle.com/database/121/JJDBC/toc.htm

[3] OTN からの 12.1.0.2 UCP.jar のダウンロード@ http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

[4] 『Oracle Universal Connection Pool for JDBC 開発者ガイド』@ https://docs.oracle.com/database/121/JJUCP/toc.htm

```
pds.setPassword(DB_PASSWORD);
//Set the pool level properties
pds.setConnectionPoolName("JDBC_UCP_POOL");
pds.setInitialPoolSize(5);
pds.setMinPoolSize(5);
pds.setMaxPoolSize(20);
Connection conn = pds.getConnection();
```

パフォーマンスを向上するための接続管理戦略

Oracle データベースで Java アプリケーションを使用する場合、さまざまな要因がパフォーマンスに影響します。ここでは、各データベース構成に適切な戦略について重点的に説明します。

1.1 単一データベース/単一インスタンス

単一インスタンスのデータベースでは、Oracle データベースとインスタンス間の関係は 1 対 1 です。

推奨事項 1：**単一インスタンスのデータベースに接続 URL を使用する**

接続 URL の推奨される形式は、記述子を使用した長い形式です。記述子を使用すると、CONNECT_TIMEOUT などのパラメータを渡すことができ、パフォーマンスも向上できるためです。

jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=15)(RETRY_COUNT=20)(RETRY_DELAY=3)(ADDRESS=(PROTOCOL=tcp)(HOST=myhost-vip)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename))

CONNECT_TIMEOUT：このパラメータを有効にすると、指定の秒数（15 秒など）の間、接続の確立が完了するのを Oracle Net Services が待機します。このパラメータは、クライアントが Oracle データベース・インスタンスへの接続を確立する時間を指定する、SQLNET.OUTBOUND_CONNECT_TIMEOUT と同等です。CONNECT_TIMEOUT は SQLNET.OUTBOUND_CONNECT_TIMEOUT よりも優先されます。

RETRY_COUNT：このパラメータは、エラー・メッセージをクライアントに返す前にネットワーク接続を再試行する回数を指定します。上記の例では、クライアントにエラー・メッセージを返す前に Oracle Net は 3 回試行します。このパラメータは、接続の可能性を高めるのに役立つため、パフォーマンスを向上するのにも役立ちます。

RETRY_DELAY：このパラメータは、再接続試行間の待機時間（秒単位）を指定します。RETRY_COUNT と組み合わせることで機能します。そのため、不要な CPU サイクルを避けるため、RETRY_DELAY と RETRY_COUNT を一緒に使用することを推奨します。

推奨事項 2：**パフォーマンス関連の接続プロパティを選択する**

注：接続レベルのプロパティの一覧については、OracleConnection[5]を参照してください。

セッション・データ・ユニット（SDU）：SDU は、Oracle Net が Java アプリケーションにデータを転送するために使用する、ネットワーク・バッファのサイズを設定します。ネットワーク経由で送信されるデータ・パケットのスループットを最適化するように SDU サイズを調整できるため、パフォーマンス、ネットワーク使用率、およびメモリ消費が改善されます。クライアントの SDU サイズは、`sqlnet.ora`、`tnsnames.ora`、または接続 URL で構成できます。sqlnet.ora ではすべての接続の DEFAULT_SDU_SIZE を変更でき、tnsnames.ora では特定のサービスの DEFAULT_SDU_SIZE を変更でき、接続 URL では特定のアプリケーションの DEFAULT_SDU_SIZE を変更できます。Oracle Database 12c では、デフォルトの SDU サイズは 8K であり、必要に応じて 2MB にまで増やすことが可能です。SDU は各接続に適用されるため、SDU を増やす場合は慎重に行う必要があります。

jdbc:oracle:thin:@(DESCRIPTION=**(SDU=11280)**
(ADDRESS=(PROTOCOL=tcp)(HOST=myhost-vip)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))

次のシナリオで、SDU を増やします。

- 長い遅延が伴う Wide Area Network（WAN）を使用する場合
- 大量のデータが返される場合
- XML ドキュメントや JSON ドキュメントを転送する場合

次のシナリオでは、デフォルトの SDU 値を変更しないでください。

- データ転送の影響がごくわずかな、高速ネットワークを使用する場合
- リクエストによってサーバーから少量のデータが返される場合

CONNECTION_PROPERTY_THIN_READ_TIMEOUT：ソケットの読取りタイムアウトを有効にすると、ファイアウォールのタイムアウトのために接続が切断され、RDBMS からの応答を待機してアプリケーションがハング状態になるのを回避できます。タイムアウト値はミリ秒単位です。

推奨事項 3：**データベースでの Java の使用**

"**データベースでの Java**"を使用すると、SQL 操作を Java データ・ロジックでグループ化し、データベースにロードしてインプレース・データ処理を実行できます。データベースでの Java には、ローカル・コールによってデータと PL/SQL サブプログラムに直接アクセスするための"サーバー側内部ドライバ"が伴います。データベースでの Java は、データ処理集中型のアプリケーションに推奨します。このようなアプリケーションでは、クライアント・アプリケーションで生じるようなネットワーク・コールが発生しないためです。アプリケーションのパフォーマンスが向上し、実行速度が高速になります。サーバー側ドライバを使用して接続するための接続文字列は、次のとおりです。

[5] JDBC Javadoc の OracleConnection @ http://docs.oracle.com/database/121/JAJDB/toc.htm

```
OracleDataSource ods = new OracleDataSource();
ods.setURL("jdbc:oracle:kprb");
// alternatively any of the following URLs
// ods.setURL("jdbc:default:connection");
Connection connection = ods.getConnection();
```

接続管理コード・サンプルの **InternalT2Driver.java**、**InternalT2Driver .sql**、**InternalT4Driver.sql**、および **InternalT4Driver.java** を参照してください。

推奨事項 4：**Universal Connection Pool（UCP）**を使用する

Universal Connection Pool（UCP）

データベース接続の作成と削除は高いコストのかかる操作であり、作成と削除を繰り返すとパフォーマンスやスケーラビリティの低下につながります。接続プールを使用すると、接続オブジェクトの再利用が促進され、接続オブジェクトが作成/削除される回数が低減します。また、接続がすでに作成されているため、アプリケーションが接続を待機する時間が短縮されます。Java アプリケーションのパフォーマンスとシステム・リソースの使用率を向上するために、Java 接続プール Universal Connection Pool[6]を使用してデータベース接続を管理することを推奨します。

UCP のパフォーマンス・プロパティ

UCP のプロパティを使用して、接続プールのサイズ制御、古い接続の処理、および短い応答時間のバランス調整を行います。最適なプール・プロパティ設定は、アプリケーション・リソースとハードウェア・リソースによって異なります。多くの場合、最適なバランスを見いだすには、さまざまな設定を試す必要があります。以下に、パフォーマンス・プロパティをいくつか示します。

MaxPoolSize は、システム・リソースが使い果たされないように、プールが維持する最大接続数を指定します。このプロパティの値を、アプリケーションからの想定接続数に基づいて設定し、必要に応じて調整することを推奨します。また、次の計算式で得られた値を設定することもできます。

MaxPoolSize = (rdbms コア数 * n) / sum (中間層あたりのプール数) ここで、n は整数であり、通常は値 9 または 10 を推奨します。

例：

ノードあたり 4 個のコアを使用し、5 つの中間層がそれぞれ 1 つのプールで 1 つの JVM を実行する、単一ノードのデータベース・サーバーについて考えてみます。

MaxPoolSize = (4 * 10) / (5 * 1) = (4 * 10) / 5 = 40 / 5 = 8

このため、おおざっぱな推定として、各中間層の MaxPoolSize は 8 になります。

中間層ごとに偶数の接続数が必要となるアプリケーションに、上記の計算式を使用できます。ただし、必要な接続が中間層ごとに異なるシナリオでは、データベースが維持できる最適な接続ワークロードを計算してから、これらの接続を各中間層に分割することを推奨します。上記の例では、接続要件に基づいて 5 つの中間層に分割できる接続総数は 40 です。

[6] 詳しくは、UCP 開発者ガイド（http://docs.oracle.com/database/121/JJUAR/toc.htm）を参照

MinPoolSize は、プールが維持する使用可能な最小接続数を指定します。この値を、アプリケーションで常に必要となる最小接続数に設定し、必要に応じて調整します。MinPoolSize は、MaxPoolSize 以下にする必要があります。デフォルト値は 0 です。

InitialPoolSize は、プールが作成された場合または再初期化された場合に作成される接続数を指定します。このプロパティの値を MinPoolSize に近い値にする必要があります。これにより、接続プールがすばやく開始されます。デフォルト値は 0 で、接続が事前に作成されないことを意味します。

同時にアクティブになる接続数が分かっており、メモリが問題とならないアプリケーションでは、*MaxPoolSize*、*InitialPoolSize*、*MinPoolSize* を同じ値に設定します。

MaxStatements は、各接続の SQL 文のサイズを指定します。文キャッシュによって、文を再解析せずにカーソルを再実行できるため、文の解析が繰り返されず、パフォーマンスとスケーラビリティが向上します。デフォルトでは、文キャッシュは無効化されます。

MaxStatements の値は、アプリケーションによって頻繁に使用される SQL 文の数に設定する必要があります。そのため、大きすぎず、かつ小さすぎない数を使用します。

TimeToLiveConnectionTimeout を使用すると、取得した接続を、事前に設定された期間だけ取得したままにできます。この期間が過ぎると、接続はプールに戻されます。接続が使用中の場合でも、強制的に戻されます。このタイムアウトは、接続の再利用を最大限に高めて、システム・リソースを節約するのに役立ちます。このプロパティの値は、アプリケーションのプロファイルに基づいて十分に大きい値に設定する必要があります。

AbandonedConnectionTimeout：取得した接続は、長時間アクティビティがない場合は使用されていないと見なされて、プールに戻されます。このタイムアウトは、接続の再利用を最大限に高めて、システム・リソースを節約するのに役立ちます。このプロパティの値は、フェイルオーバー時間を含め、接続をチェックアウトできる最大時間の 2 倍または 3 倍に設定する必要があります。

InactiveConnectionTimeout は、使用可能な接続を閉じてプールから削除する前に、アイドル状態にできる時間を指定します。このプロパティは、使用可能な接続にのみ適用され、取得した接続には適用されません。このタイムアウトは、リソースを節約するのに役立ちます。デフォルト値は 0 です。0 以外の値は、アプリケーション要件によって異なります。

ConnectionWaitTimeout は、接続を取得する前にアプリケーションが待機できる時間を指定します。このタイムアウトが過ぎると、例外がスローされます。このタイムアウトによって、アプリケーションがブロックされる時間が最小限に抑えられるため、全体的なアプリケーション使用可能性が向上します。また、正常なリカバリを実行できるようになります。デフォルト値は 3 秒です。アプリケーションに基づいて値を選択します。

接続管理コード・サンプルのコード・サンプル **UCPSample.java** および **UCPWithTimeoutProperties.java** を参照してください。

推奨事項 5：**接続ラベル付けを使用する**

Universal Connection Pool では、アプリケーションがカスタム・ラベル、つまり非トランザクション・ステート（NLS、トランザクション分離、カスタム・ステートなど）を接続に関連付けて、同じ接続を後で検索することが可能です。接続ラベル付けを使用すると、接続の再初期化にかかる時間とコストを回避できます。"接続ラベル付け"について詳しくは、『*Oracle Universal Connection Pool for JDBC 開発者ガイド*』[7]を参照してください。

接続管理コード・サンプルの **UCPConnectionLabelingSample.java** を参照してください。

1.2 Oracle Real Application Clusters（Oracle RAC）

Oracle Real Application Clusters[8]構成では、*Oracle データベースと 1 つまたは複数のインスタンス間の関係は 1 対多です。*Oracle RAC は、高パフォーマンス、スループットの向上、高可用性、スケーラビリティの強化の各要件を満たします。Oracle RAC は、Oracle Multitenant および Oracle Active Data Guard とともにデプロイできます。

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が Real Application Clusters に適用されます。また、次の推奨事項も適用されます。

推奨事項 6：**Oracle RAC データベースの接続 URL**

ここでは、パフォーマンス関連の接続記述子について説明します。

```
jdbc:oracle:thin:@
(DESCRIPTION=
 (CONNECT_TIMEOUT=15)(RETRY_COUNT=20) (RETRY_DELAY=3)
  (ADDRESS_LIST =
   (LOAD_BALANCE=ON)
(ADDRESS=(PROTOCOL=tcp)(HOST=primaryscan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))
```

CONNECT_TIMEOUT、RETRY_COUNT、RETRY_DELAY、SERVICE_NAME：これらの記述子について詳しくは、「推奨事項 1：単一インスタンスのデータベースに接続 URL を使用する」を参照してください。

*HOST：*常に SCAN 名を使用します。これにより、管理性が向上します。詳しくは、「推奨事項 33：Single Client Access Name（SCAN）を使用する」を参照してください。

*SERVICE_NAME：*常にアプリケーション・サービス名を使用します。サービスは位置の透過性を提供し、ロードバランシングを促進し、作業のリカバリを可能にし、パフォーマンス属性を提供します。デフォルトのデータベース・サービスは、db_name、db_unique_name、または PDB 名に相当します。データベース管理者専用に使用されているデフォルト・データベース名や、Oracle Enterprise Manager や DBA 向けに予約されているデフォルト・データベース名は使用しないようにします。

---

[7] 『Oracle Universal Connection Pool for JDBC 開発者ガイド』@ https://docs.oracle.com/database/121/JJUCP/toc.htm

[8] 利点について詳しくは、データシート『Oracle Real Application Clusters』（www.oracle.com/technetwork/jp/database/options/clustering/rac-ds-12c-1898881-ja.pdf）を参照

推奨事項 7：**Universal Connection Pool（UCP）を使用する**

Universal Connection Pool は Oracle RAC と緊密に統合されており、Oracle RAC データベースによって提供されるすべてのメリットと機能を活用できることが実証されているソリューションです。UCP および UCP のパフォーマンス・プロパティについて詳しくは、「単一データベース/単一インスタンス」を参照してください。Oracle RAC 環境では、次の公式で得られる値に *MaxPoolSize* を設定します。

*MaxPoolSize = (sum (インスタンスあたりの rdbms コア数) * n) / sum (中間層あたりのプール数)*

ここで、n は整数であり、通常は値 9 または 10 を推奨します。例：

ノードあたり 24 個のコアを使用し、10 の中間層がそれぞれ 1 つのプールで 2 つの JVM を実行する 3 ノードの Oracle RAC クラスタについて、n=10 として考えてみます。

MaxPoolSize = (sum (24, 24, 24) * 10) / (10 * 2)

= (72 * 10) / 20 = 720 / 20 = 36

おおざっぱな推定として、MaxPoolSize は 36 になります。

中間層ごとに偶数の接続数が必要となるアプリケーションに、上記の計算式を使用できます。ただし、必要な接続が中間層ごとに異なるシナリオでは、最適な接続ワークロードを計算してから、これらの接続を各中間層に分割します。上記の例では、接続要件に基づいて 10 の中間層に分割できる接続総数は 720 です。

推奨事項 8：**接続アフィニティを使用する**

接続アフィニティは、特定の Oracle RAC インスタンスに向けられる接続をプールが選択できるようにする、Oracle RAC のパフォーマンス機能です。プールは、実行時接続ロードバランシング（構成されている場合）を使用してインスタンス固有の接続を選択します。同じアプリケーションからの以降の接続リクエストは、アフィニティによって同じインスタンスに割り当てられます。UCP は、（a）Web セッション・アフィニティと（b）トランザクション・ベースのアフィニティ [9]（XA アフィニティ）の 2 つのタイプの接続アフィニティをサポートしています。

UCP の Web セッション・アフィニティ：

Web セッション・アフィニティの場合、UCP は、同じ Web セッションでの連続した接続リクエストに対して、接続を同じデータベース・インスタンスに割り当てるように試みます。アプリケーションがアフィニティ・コールバックを実装する必要があります。

以下に、Web セッションのアフィニティ・コールバックのコード・フラグメントを示します。接続管理コード・サンプルの **UCPWebSessionAffinitySample.java** を参照してください。

[9] トランザクション・アフィニティ（XA アフィニティ）については、このホワイト・ペーパーでは説明していません。

```
MyConnectionAffinityCallback callback = new MyConnectionAffinityCallback();
poolDataSource.registerConnectionAffinityCallback(callback);
// Web Session Affinity Callback implementation
public class MyConnectionAffinityCallback implements
ConnectionAffinityCallback {
   // Affinity policy used by callback
   ConnectionAffinityCallback.AffinityPolicy affinityPolicy =
      ConnectionAffinityCallback.AffinityPolicy.WEBSESSION_BASED_AFFINITY;

   public MyConnectionAffinityCallback() {
   }
   // Other Affinity callback methods go here
   ...
}
```

1.3 マルチテナント・データベース – 複数データベース/単一インスタンス

Oracle Multitenant は Oracle Database 12c のオプションであり、統合、プロビジョニング、アップグレードなどを簡素化します。Oracle Multitenant は、コンテナ・データベース（CDB）によって多数のプラガブル・データベース（PDB）を管理できるようにします。PDB は、顧客とアプリケーションのデータとメタデータを格納する本格的なデータベースです。Oracle Multitenant は Oracle Real Application Clusters や Oracle Active Data Guard など、他のオプションを完全に補完します。Oracle Net 経由での PDB への接続は、非 CDB データベースへの接続と同様です。

「単一データベース/単位インスタンス」で説明したすべての接続管理戦略が、マルチテナント・データベースに適用されます。また、次の推奨事項も適用されます。

推奨事項 9：**マルチテナント・データベースの接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION=
(ADDRESS=(PROTOCOL=tcp)(HOST=myhost-vip)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=mypdbservice)))
```

SERVICE_NAME：サービス名が特定の PDB に関連付けられた有効なサービス名であり、CDB 内で一意である必要があります。db_unique_name（データベース名または PDB 名）と相関関係のあるサービス名は使用しないでください。

推奨事項 10：**共有の UCP プールを使用する**

"共有の UCP プール"を使用すると、1 つの接続プールを複数の PDB（またはテナント）で共有できるため、パフォーマンスが向上し、データベース・リソースの使用率も向上します。Oracle Database 12.1 で使用可能なソリューションでは、1 つの接続を複数の PDB で再利用するために必要となる、構成手順やプログラミング手順はごくわずかです。共有の UCP プールを使用するには、UCP の接続ラベル付け、グローバル・データベース・ユーザー（すべての PDB へのアクセス権限を持つ）、およびコールバック関数での新しい SET CONTAINER 文が必要です。複数の PDB に共有プールを作成するためのステップ・バイ・ステップの手順については、『Oracle Universal Connection Pool for JDBC 開発者ガイド』[10]を参照してください。接続管理コード・サンプルの UCPMultitenantSample.java および UCPMultitenantSample.sql も参照してください。

1.4 Data GuardまたはActive Data Guard - 複数データセンター/単一インスタンス

Oracle Data Guard は、企業データの高可用性、データ保護、障害時リカバリを確保します。Oracle Data Guard は、スタンバイ・データベースを本番データセンターのコピーとして保持し、本番データセンターが使用できなくなると、スタンバイ・データベースを本番ロールに切り替えて、停止に伴うダウンタイムを最小限に抑えます。Oracle Data Guard は Oracle Enterprise Edition に付属しており、追加のライセンス料金なしに使用できます。

Oracle Active Data Guard[11]は Oracle Data Guard のスーパーセットであり、追加機能としてリアルタイム問合せ（パフォーマンスと ROI）、Far Sync（いかなる距離でもデータ損失ゼロの保護）、自動ブロック修復（HA）、データベース・ローリング・アップグレードを備えています。Active Data Guard を Oracle Database Enterprise Edition で使用するには、追加のライセンスが必要です。

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Data Guard または Active Data Guard のデータベース構成に適用されます。また、接続 URL に以下の関連する接続記述子を使用する必要があります。

推奨事項 11：**Oracle DG または Oracle ADG の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on) (LOAD_BALANCE=off)
   (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST =
          (ADDRESS = (PROTOCOL = TCP) (HOST=primary-vip) (PORT=1521)))
      (ADDRESS_LIST =
          (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-vip) (PORT=1521)))
      (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

CONNECT_TIMEOUT、*RETRY_COUNT*、*RETRY_DELAY*、*SERVICE_NAME*：これらの記述子について詳しくは、「推奨事項 1：単一インスタンスのデータベースに接続 URL を使用する」を参照してください。

---

[10] 『Oracle Universal Connection Pool for JDBC 開発者ガイド』@https://docs.oracle.com/database/121/JJUCP/toc.htm

[11] 『Oracle Active Data Guard』を参照@ http://www.oracle.com/technetwork/jp/database/availability/active-data-guard-wp-12c-1896127-ja.pdf

1.5 Oracle DGまたはOracle ADGとOracle RAC - 複数データセンター/複数インスタンス

Oracle RAC と Data Guard のアーキテクチャは Oracle RAC と Oracle DG の両方のメリットを兼ね備えているため、Maximum Availability Architecture（MAA）の推奨アーキテクチャとなっています。

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG と Oracle RAC に適用されます。Oracle ADG/DG と Oracle RAC に関連する接続記述子を接続 URL に使用することが、重要な戦略となります。

推奨事項 12：**Oracle DG または Oracle ADG と Oracle RAC の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on) (LOAD_BALANCE=off)
   (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521)))
      (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-scan) (PORT=1521)))
   (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

CONNECT_TIMEOUT、*RETRY_COUNT*、*RETRY_DELAY*、*SERVICE_NAME*：これらの記述子について詳しくは、「推奨事項 1：単一インスタンスのデータベースに接続 URL を使用する」を参照してください。

1.6 Oracle Global Data Services（Oracle GDS） – 地理的に複数の場所

Oracle Global Data Services[12]は、ユーザーに近接しているインスタンスから地理的に分散された顧客に対して効率的にサービスを提供する、アプリケーションを対象としています。Oracle GDS は Oracle Data Guard Broker と統合されており、Oracle Active Data Guard に含まれています。

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Oracle GDS データベース構成に適用されます。また、Oracle GDS に関連する接続記述子を接続 URL に使用する必要があります。

[12] GDS ホワイト・ペーパーを参照@ http://www.oracle.com/technetwork/jp/database/availability/global-data-services-12c-wp-1964780-ja.pdf

推奨事項 13：**Global Data Services の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION= (FAILOVER=on) (LOAD_BALANCE=off)
   (CONNECT_TIMEOUT=15) (RETRY_COUNT=20) (RETRY_DELAY=3)
    (ADDRESS_LIST=
       (LOAD_BALANCE=ON)
       (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm1) (PORT=1571)))
    (ADDRESS_LIST=
       (LOAD_BALANCE=ON)
       (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm2) (PORT=1572)))
   (CONNECT_DATA= (SERVICE_NAME=gold-cloud-service-name) (REGION=region1))
```

CONNECT_TIMEOUT、*RETRY_COUNT*、*RETRY_DELAY*：これらの記述子について詳しくは、「推奨事項 1：Oracle RAC データベースの接続 URL」を参照してください。
SERVICE_NAME：グローバル・サービスは、データ・レプリケーションによって同期される複数のデータベースにより提供されるデータベース・サービスです。Oracle GDS の場合は、**グローバル・サービス**名であるサービス名を使用することを推奨します。

スケーラビリティとロードバランシングを実現するための接続管理戦略

ここでは、各データベース構成に適切な戦略を選択することによる、スケーラビリティの向上とロードバランシングへの対処について重点的に説明します。

2.1 単一データベース/単一インスタンス

推奨事項 14：**データベース・サービスを使用する**

データベース・サービスは単一データベースを表し、ワークロードを相互にバラバラのグループに分割します。データベース・サービスは、一般的な属性、サービス・レベルしきい値、および優先順位が設定されたワークロードに使用します。データベース・サービスを使用すると、単一データベースのワークロードを構成、管理、および有効化/無効化でき、ワークロードを単一エンティティとして測定できます。データベース・サービスを使用する際、アプリケーション・コードの変更は不要です。たとえば、Oracle E-Business Suite によって、各部分（総勘定元帳、売掛金、受注など）のデータベース・サービスが定義されます。

jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=15)
(ADDRESS=(PROTOCOL=tcp)(HOST=myhost-vip)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename))
SERVICE_NAME：データベース・サービス名を使用します。

推奨事項 15：**UCP を使用して実行時ロードバランシングを有効化する**

Oracle RAC と Oracle GDS のサービス側で、実行時ロードバランシングを有効化します。詳しくは、ホワイト・ペーパー[13]の「**Run Time Load Balancing**」の項を参照してください。

'実行時ロードバランシングの目標'を SERVICE_TIME または THROUGHPUT に設定します。

---

[13] 『Java Programming with Oracle Database 12c RAC and Active Data Guard』@ http://www.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

また、Oracle RAC と Oracle GDS の両方のサーバー側で、'接続ロードバランシングの目標'を SHORT に設定します。

```
$srvctl modify service –db <db_name> -service <service_name> -clbgoal SHORT
$srvctl modify service –db <db_name> -service <service_name> -rlbgoal SERVICE_TIME
```

```
$gdsctl modify service –db <db_name> -service <service_name> -rlbgoal SERVICE_TIME
$gdsctl modify service –db <db_name> -service <service_name> -clbgoal SHORT
```

推奨事項 16：**共有サーバーを使用する**

Oracle データベースのサーバー・プロセスは、専用（デフォルト）、共有サーバー（MTS）、またはプール・サーバー（DRCP）のいずれかです。

共有サーバー・アーキテクチャでは、ディスパッチャが複数の着信ネットワーク・セッション要求を共有サーバーのプロセス・プールに転送するため、各接続に対する専用サーバーのプロセスが必要なくなり、多数のユーザー数に対応する場合にシステム・リソースが低減されます。

いずれのデータベース構成でも、接続数が中程度のアプリケーションでスケーラビリティを達成するには、共有サーバーを推奨します。接続数が少ないアプリケーションには、専用セッションを推奨します。

共有サーバーの欠点は、ディスパッチャによる追加のホップ、一部のデータベース機能のサポート制限、およびセットアップとチューニングの複雑さです。

一般的なガイドラインとして、データベースへの同時接続の数がオペレーティング・システムで処理できる数を超えている場合のみ、共有サーバーを使用します。初期化パラメータ SHARED_SERVERS および MAX_SHARED_SERVERS を使用すると、作成される共有サーバーの数を制御できます。データベース・サーバーの最適な接続数を計算するには、次の計算式を使用します。

*MaxConnections = (sum (インスタンスあたりの rdbms コア数) * n)*

ここで、n は整数であり、通常は値 9 または 10 を推奨します。

例：ノードあたり 4 個のコアを使用する単一ノード・データベース・サーバーについて、n = 10 として考えてみます。

MaxConnections = (4 * 10) = 40 となります。

サーバー側の構成：

初期化パラメータをいくつか設定して、共有サーバーを構成および有効にする必要があります。データベース側で共有サーバーを有効化する手順について、詳しくは『Oracle Database 管理者ガイド』[14]を参照してください。

クライアント側の構成：

```
jdbc:oracle:thin:@
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=proddbcluster)(PORT=5521)))
    (CONNECT_DATA=(SERVICE_NAME=proddb)(SERVER=SHARED)))
```

*SERVER：*このパラメータを使用して特定のサービス・ハンドラ・タイプを指定します。データベースへの接続時に、接続記述子は、共有サーバーの使用を示す(SERVER=SHARED)を使用してディスパッチャをリクエストします。(SERVER=DEDICATED)は専用サーバーを示し、これがデフォルト動作です。

推奨事項 17：**Oracle データベース常駐接続プール（DRCP）を使用する**

データベース常駐接続プールは、接続ブローカによって管理される一連の"専用サーバー"で構成されるサーバー側接続プールです。接続ブローカは"プール・サーバー"を管理し、リクエストに基づいてこれらのプール・サーバーを複数の中間層間で共有します。

多数の中間層（Web サーバー）が同じデータベース・サーバーにアクセスする必要があり、アクティブな接続の数がオープンな接続の数よりも大幅に少ない環境でスケーラビリティを向上するには、すべてのデータベース構成に DRCP を使用する必要があります。DRCP について詳しくは、『*Oracle Database JDBC 開発者ガイド*』[15]を参照してください。

DRCP の欠点は、認証を行ってプールから接続を取得するためには、接続ブローカに追加のホップが必要になることです。ただし、接続が確立されると、DRCP は専用サーバーとまったく同じように動作します。また、DRCP を使用すると、サーバー側の CPU 消費とメモリ消費が増加します。データベース接続を複数の中間層プロセスで共有する必要がある場合のみ、DRCP を使用します。

サーバー側の構成：

サーバー側で DRCP プールを構成して起動するには、`DBMS_CONNECTION_POOL` パッケージを使用します。プールを起動するには `DBMS_CONNECTION_POOL.START_POOL()`を使用し、プールを停止するには`DBMS_CONNECTION_POOL.STOP_POOL()`を使用します。また、サーバー側で文キャッシュを有効にするには、`DBMS_CONNECTION_POOL.CONFIGURE_POOL`（`session_cached_cursors=>50`）を起動します。

[14] 『Oracle Database 管理者ガイド』 - http://docs.oracle.com/database/121/ADMIN/manproc.htm

[15] 『Oracle Database JDBC 開発者ガイド』 - https://docs.oracle.com/database/121/JJDBC/toc.htm

Client Side Configurations:

```
jdbc:oracle:thin:@
(DESCRIPTION=
  (ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=tcp)(HOST=proddbcluster)(PORT=5521)))
(CONNECT_DATA=(SERVICE_NAME=proddb-service)(SERVER=POOLED)))
```

SERVER：接続記述子(SERVER=POOLED)を使用して DRCP を有効化します。このパラメータに許可される値は、DEDICATED、SHARED、および POOLED です。

接続を取得している間、接続プロパティ `oracle.jdbc.DRCPConnectionClass` に NULL 以外の値または空でない文字列値を設定する、コード変更が必要です。

DRCP で、クライアント側接続プールとして UCP を使用する

UCP は、接続ブローカへの通信や接続を維持するのに役立ちます。クライアント側の接続プールは、attachServerConnection()および detachServerConnection()によって接続ブローカへの接続を接続および切断する必要があります。サード・パーティのクライアント・プールではなく UCP を使用する利点は、UCP ではサーバー接続の接続と切断が透過的に処理される点です。接続管理コード・サンプルの **UCPWithDRCPSample.java** を参照してください。

UCP なしに DRCP を使用する

DRCP をサード・パーティのクライアント側接続プールと組み合わせて使用できます。サード・パーティのクライアント側接続プールは、attachServerConnection()メソッドおよび detachServerConnection()メソッドによって接続を接続ブローカに明示的に接続および切断する必要があります。　また、サンプルに示されているように、接続クラスを設定する必要があります。接続管理コード・サンプルの **DRCPSample.java** を参照してください。

```
OracleDataSource ods = new OracleDataSource();
// DRCP Property: Connection Class
// Set the connection-class to share connections across pool
Properties connproperty = new Properties();
connproperty.setProperty("oracle.jdbc.DRCPConnectionClass","DRCP_conn_class");
ods.setConnectionProperties(connproperty);
try (OracleConnection connection = (OracleConnection) (ods.getConnection())) {
   System.out.println("DRCP enabled: " + connection.isDRCPEnabled());
   connection.attachServerConnection();
   doSQLWork(connection);
   connection.detachServerConnection((String)null);
}
```

2.2　Oracle Real Application Clusters（Oracle RAC）

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Oracle Real Application Clusters に適用されます。また、接続 URL にスケーラビリティ/ロードバランシングの記述子を含める必要があります。

推奨事項 18：**Oracle RAC に接続 URL を使用する**

```
jdbc:oracle:thin:@
(DESCRIPTION=
 (CONNECT_TIMEOUT=15)(RETRY_COUNT=20) (RETRY_DELAY=3)
  (ADDRESS_LIST =
   (LOAD_BALANCE=ON)
    (ADDRESS=(PROTOCOL=tcp)(HOST=primaryscan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))
```

*ADDRESS_LIST：*アドレスが 1 つのみの場合、ADDRESS_LIST は必要ありません。Oracle RAC の場合、ADDRESS_LIST を使用すると、各 ADDRESS に接続記述子 LOAD_BALANCE および FAILOVER を指定できます。そのため、SCAN に指定されているすべてのサイトまたは ADDRESS に指定されているすべてのホスト名で、接続時ロードバランシングが可能になります。

*LOAD_BALANCE：*このパラメータを使用すると、接続時ロードバランシングが可能になるため、ノードの負荷に基づいてリスナーがルーティングを決定できます。このパラメータを OFF に設定すると、Oracle Net は成功するまでプロトコル・アドレスを順番に試行します。

SCAN ベースのアドレスに対してこのパラメータを ON に設定すると、新しい接続要求は、DNS によって解決される 3 つの SCAN ベースの IP アドレスのいずれかにランダムに割り当てられます。このランダム化によって、着信接続要求を処理するジョブをすべてのリスナーが共有できます。SCAN を使用しないクライアントの場合、Oracle Net Services はアドレス・リスト内のアドレスをランダムに選択して、そのノードのリスナーに接続します。

2.3 マルチテナント・データベース

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、マルチテナント・データベースに適用されます。

2.4 Data GuardまたはActive Data Guard

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Data Guard または Active Data Guard に適用されます。また、接続 URL にスケーラビリティ/ロードバランシングの記述子を含める必要があります。

推奨事項 19：**Oracle DG または Oracle ADG の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on) (LOAD_BALANCE=off)
   (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP) (HOST=primary-vip) (PORT=1521)))
      (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-vip) (PORT=1521)))
   (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

*LOAD_BALANCE（接続時ロードバランシング）：*このパラメータを ON に設定すると、Oracle Net はプロトコル・アドレスのリストをランダムな順番で処理し、さまざまなホスト名アドレスでロードバランシングを実行します。OFF に設定すると、Oracle Net は、成功するまでアドレスを順番に試します。単一インスタンスの場合は OFF にします。

2.5 Oracle DGまたはOracle ADGとOracle RAC

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG と Oracle RAC に適用されます。また、接続 URL にスケーラビリティ/ロードバランシングの記述子を含める必要があります。

推奨事項 20：**Oracle DG または Oracle ADG と Oracle RAC の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on)
   (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST =
         (LOAD_BALANCE=on)
         (ADDRESS = (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521)))
      (ADDRESS_LIST =
         (LOAD_BALANCE=on)
         (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-scan) (PORT=1521)))
   (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

HOST: Always use SCAN name

*LOAD_BALANCE（接続時ロードバランシング）：*詳しくは、「推奨事項 19：Oracle DG または Oracle ADG の接続 URL」を参照してください。

2.6 Oracle Global Data Services（Oracle GDS）

「単一データベース/単一インスタンス」で説明したすべての接続管理戦略が Oracle GDS に適用されます。また、接続 URL にスケーラビリティ/ロードバランシングの記述子を含める必要があります。

推奨事項 21：**Oracle GDS の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION= (FAILOVER=on)
   (CONNECT_TIMEOUT=15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm1) (PORT=1571)))
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm2) (PORT=1572)))
   (CONNECT_DATA= (SERVICE_NAME=gold-cloud-service-name) (REGION=region1))
```

*HOST：*常にグローバル・サービス・マネージャ（GSM）リスナーを使用します。このリスナーは、Oracle RAC データベースの SCAN リスナーに似ています。GSM は、Oracle GDS のデータベース間サービスのフェイルオーバーとロードバランシングを実行するのに役立つ"グローバル・リスナー"です。

*REGION：*これも、Oracle GDS の重要な接続記述子であり、この記述子でリージョンを指定します。実行時ロードバランシング・アドバイザリは Oracle GDS の特定のリージョン用にカスタマイズされ

るため、リージョン名が必要です。GDS リージョンの例は、アジア、ヨーロッパなどです。

高可用性（HA）を実現するための接続管理戦略

停止時間（リソースを使用できない時間）をエンドユーザーから隠すように、アプリケーションを設計する必要があります。停止時間には、計画停止時間と計画外停止時間があります。高可用性のおもな特徴として、信頼性、リカバリ可能性、タイムリーなエラー検出、および継続的な運用があります。Oracle Database 12c は、計画外停止時間を短縮または回避し、障害からの迅速なリカバリを可能にし、計画停止時間を最小限に抑える、統合型の HA テクノロジーを提供します。Oracle Real Application Clusters、Oracle Data Guard、Oracle Active Data Guard、およびアプリケーション継続性は、高可用性を実現する主要コンポーネントです。ここでは、各データベース構成で高可用性を達成するための戦略について重点的に説明します。

3.1 単一インスタンスのデータベース

単一インスタンスのデータベースは高可用性機能を提供しません。そのため、高可用性機能を利用するために、単一インスタンスのデータベースを Oracle RAC データベースまたは Oracle RAC One Node データベースに変換することを推奨します。詳しくは、「*3.2 Oracle RAC One Node*」および「*3.3 Oracle Real Application Clusters（Oracle RAC）*」を参照してください。

3.2 Oracle RAC One Node

Oracle Real Application Clusters One Node（Oracle RAC One Node）は、クラスタ内の 1 つのノードで実行される、単一インスタンスの Oracle Real Application Clusters データベースです。Oracle RAC One Node は、Oracle RAC と同じインフラストラクチャで使用できる利点があります。

オンライン・データベース再配置機能によって、メンテナンス中に Oracle RAC One Node が別のノードに再配置されるため、サービスの可用性が確保されます。Oracle RAC One Node データベースの 2 つ目のインスタンスは、計画的なオンライン・データベース再配置時にのみ作成されます。計画メンテナンスの開始時に、プライマリ・インスタンスを停止したら、サービス再配置（`srvctl relocate service`）によってすべてのサービスを 2 つ目のインスタンスに移動させます。すべての接続が再配置先のインスタンスに移動するまで待ってから、プライマリ・データベース・インスタンスを停止させて、残りの接続を再配置先のインスタンスに移動させます。

1 分以内のリカバリを必要とする顧客は、複数ノードの Oracle Real Application Clusters にデータベースをデプロイする必要があります。Oracle RAC によって、最高の可用性と、最短の障害時リカバリが実現されます。

計画メンテナンスを隠す

計画メンテナンスは、テクノロジー・インフラストラクチャの定期的なメンテナンス、ソフトウェアのアップグレード/パッチ適用、またはハードウェアのメンテナンスを実施するために必要です。計画的な中断を最小限に抑えるようにシステムを設計することが重要です。

計画停止時間を隠すために推奨される接続管理戦略は、サーバーのメンテナンス・スケジュールからワークロードをすべて排出することです。高速接続フェイルオーバー（FCF）を有効にすると、UCP プロパティによってこの排出が処理されます。

推奨事項 22：**FAN イベントと排出を有効化する**

Oracle Notification Service（ONS）を使用して、DOWN、Planned DOWN、UP、RLB%などの高速アプリケーション通知（FAN）イベントがサブスクライバ（データベース・ドライバ、コンテナ）に高速かつ確実に通知されます。高速接続フェイルオーバー（FCF）は、UCP が FAN イベントにサブスクライブして、迅速な接続フェイルオーバーやワークロードのリバランスをイベントに基づいて支援するメカニズムです。"Apache Tomcat での UCP 構成"は、以下に示すようになります。UCP と FCF を使用するアプリケーションは、計画メンテナンスを適切に処理します。スムーズで透過的な計画メンテナンスを達成するためには、DBA と開発者はホワイト・ペーパー[16]に記載されている手順を実行する必要があります。

```
<Context docBase="UCPTomcat" path="/UCPTomcat"
    reloadable="true" source="org.eclipse.jst.jee.server:UCPTomcat">
<Resource name="tomcat/UCPPool" auth="Container"
      factory="oracle.ucp.jdbc.PoolDataSourceImpl" type="oracle.ucp.jdbc.PoolDataSource"
      description="UCP Pool in Tomcat"
      connectionFactoryClassName="oracle.jdbc.pool.OracleDataSource"
      minPoolSize="5"  maxPoolSize="50"  initialPoolSize="15"  autoCommit="true"
      user="scott"  password="tiger"
      fastConnectionFailoverEnabled="true"
      url
="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))"
/>
```

計画外停止時間を隠す

Oracle データベースは、あらゆるタイプの計画外停止において、停止時間を防止、許容、短縮する高可用性ソリューションを提供します計画外停止時間は、サイト障害、コンピュータ障害、ストレージ障害、データ破損、または人為的ミスが原因で生じる可能性があります。

すべてのデータ構成において、計画外停止を隠すために推奨される接続管理戦略は、アプリケーション継続性を使用することです。アプリケーション継続性はトランザクション・ガード（TG）を内部的に起動し、計画外停止をエンドユーザーから隠すように試みます。

[16] ホワイト・ペーパー『Design and Deploy Tomcat Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP』@ http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-tomcat-2265175.pdf

推奨事項 23：**トランザクション・ガード（TG）を使用する**

トランザクション・ガードは、コミットを 1 回のみ実行するための API を提供します。TG は、トランザクションがコミットおよび完了されたかどうかを示す、確実なコミット結果を提供します。すべてのトランザクションが論理トランザクション識別子（LTXID）でタグ付けされます。コミットが失敗すると、アプリケーションはこの識別子を使用して、実行中であったトランザクションが失敗前にコミットされていたかどうかを確認できます。トランザクション・ガードは、アプリケーションが重複したトランザクションを送信するのを防ぎます。TG を有効にするためのアプリケーションの変更およびサーバー側の変更については、ホワイト・ペーパー[17]を参照してください。

推奨事項 24：**アプリケーション継続性（AC）を有効化する**

アプリケーション継続性は、数多くの計画外停止を隠すことで、エンドユーザー・エクスペリエンスを向上させます。計画停止時と計画外停止時に、アプリケーション継続性は実行中であったアプリケーション・リクエストを再実行し、停止時間を隠すように試みます。リクエストは、単一の Web リクエストの DML 文やその他のデータベース呼出しに対応する、アプリケーション作業ユニットです。

アプリケーション継続性 [18]をサーバー側とクライアント側で有効にする必要があります。Java アプリケーションでアプリケーション継続性を実現するためには、次の構成が必要です。

» Oracle Database 12c Release 1（12.1）

» Java アプリケーションで Oracle Universal Connection Pool と Oracle 再実行データソース（**`oracle.jdbc.replay.OracleDataSourceImpl`**）を使用する。また、UCP の `setFastConnectionFailover`（`true`）を使用して Java アプリケーションを FAN イベントにサブスクライブする。

» Oracle Real Application Clusters または Oracle Data Guard で、Oracle Notification Service を使用して FAN を構成する。Oracle RAC と Oracle DG では、ONS はデフォルトで構成されます。

» アプリケーション・サービスを使用する。Oracle Enterprise Manager と DBA に予約されているデフォルトのデータベース・サービスは使用しない。

» 再実行とロードバランシングを実行するために、サービスに必要なプロパティを設定する（下記参照）。

```
$srvctl modify service -db <db_name> -service <service_name>   -
failovertype TRANSACTION   -replay_init_time 600 -failoverretry 30 -
failoverdelay 10 -commit_outcome TRUE
```

» **ojdbc7.jar、ucp.jar、ons.jar** をクラス・パスに設定する。これらのすべての jar が同じデータベース・バージョン（12.1.0.2 など）のものである必要があります。

---

[17] 『Java Programming with Oracle Database 12c RAC and Active Data Guard』
www.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

[18] ホワイト・ペーパー『Application Continuity with Oracle Database 12c』
http://www.oracle.com/technetwork/jp/database/database-cloud/private/application-continuity-wp-12c-1966213-ja.pdf

» Oracle JDBC 具象クラス [19]を標準の JDBC インタフェースに置き換える。AC は具象クラスをサポートしていないため、AC を使用する前に具象クラスを置き換える必要があります。

» 意図しない影響を避けるため、再実行を無効化する。メール送信や印刷チェックなどに関連するトランザクションを再実行すると、意図しない影響が発生する可能性があるため、`ReplayableConnection` インタフェースの `disableReplay()`メソッドを使用してこのような重要なセクションの再実行を無効化する必要があります。

3.3 Oracle Real Application Clusters（Oracle RAC）

「3.2：Oracle RAC One Node」で説明したすべての接続管理戦略が、Oracle Real Application Clusters に適用されます。また、接続 URL に高可用性の記述子を含める必要があります。

推奨事項 25：**Oracle RAC の接続 URL**

```
jdbc:oracle:thin:@
(DESCRIPTION=
 (CONNECT_TIMEOUT=15)(RETRY_COUNT=20) (RETRY_DELAY=3)
   (ADDRESS_LIST =
      (LOAD_BALANCE=ON)
         (ADDRESS=(PROTOCOL=tcp)(HOST=primaryscan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))
```

接続 URL：以下のことに留意してください。

(a) Oracle RAC への接続に簡易接続 URL（EZConnect。例：jdbc:oracle:thin:@//localhost:1521/myorcldbservicename）は使用しないでください。簡易接続 URL を使用すると接続記述子を指定できないため、高可用性やロードバランシングなどを利用できません。

(b) 古いバージョンの構文は使用しないでください。上記に示すように、常に最新バージョンの接続 URL 構文を使用します。

HOST：常に SCAN を使用し、データベース・ホスト名は使用しないことを推奨します。データベース・ホスト名を使用すると、高可用性機能を利用できないためです。

3.4 Data GuardまたはActive Data Guard

「3.2：Oracle RAC One Node」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG に適用されます。また、接続 URL に高可用性の記述子を含める必要があります。

[19] MOS Note『New Jdbc Interfaces for Oracle types』（Doc ID 1364193.1）を参照してください。

推奨事項 26：**Oracle DG または Oracle ADG の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on) (LOAD_BALANCE=off)
  (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
     (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST=primary-vip) (PORT=1521)))
     (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-vip) (PORT=1521)))
  (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

FAILOVER：接続時フェイルオーバーを有効化します。ON に設定すると、Oracle Net は、最初のリスナーが失敗した場合は別のリスナーにフェイルオーバーします。試行するアドレスの数は、リスト内のアドレスの数で決まります。OFF に設定すると、Oracle Net は 1 つのアドレスのみ試行します。Oracle ADG または Oracle DG のデータベース構成では、常に FAILVOER を ON に設定します。

3.5 Oracle DGまたはOracle ADGとOracle RAC

「3.2：Oracle RAC One Node」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG と Oracle RAC に適用されます。また、接続 URL に高可用性の記述子を含める必要があります。

推奨事項 27：**Oracle DG または Oracle ADG と Oracle RAC の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION = (FAILOVER=on)
  (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
     (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521)))
     (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP) (HOST=secondary-scan) (PORT=1521)))
  (CONNECT_DATA= (SERVICE_NAME = gold-cloud-service-name)))
```

FAILOVER：「推奨事項 26：Oracle DG または Oracle ADG の接続 URL」に記載されている FAILOVER の説明を参照してください。

3.6 Oracle Global Data Services（Oracle GDS）

「3.2：Oracle RAC One Node」で説明したすべての接続管理戦略が Oracle GDS に適用されます。また、接続 URL に高可用性の記述子を含める必要があります。

推奨事項 28：**Oracle GDS の接続 URL**

```
jdbc:oracle:thin:@(DESCRIPTION= (FAILOVER=on)
   (CONNECT_TIMEOUT=15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm1) (PORT=1571)))
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm2) (PORT=1572)))
   (CONNECT_DATA= (SERVICE_NAME=gold-cloud-service-name) (REGION=region1))
```

FAILOVER：「推奨事項 26：Oracle DG または Oracle ADG の接続 URL」に記載されている FAILOVER の説明を参照してください。

セキュリティを向上するための接続管理戦略

Oracle データベースが装備している豊富なセキュリティ機能を使用すると、ユーザー・アカウント、認証、権限とロール、アプリケーション・セキュリティ、暗号化、ネットワーク・トラフィック、および監査に対処できます。

接続管理戦略は、企業のセキュリティ要件/制限に基づいて以下のいずれかのセキュリティ機能を選択することです。これらのセキュリティ機能を、単一インスタンスのデータベース、Oracle Real Application Clusters、Oracle Multitenant、Oracle Data Guard、Oracle Active Data Guard、Oracle Global Data Services などのデータベース構成に適用できます。

- **高度なセキュリティ**：データ暗号化とデータ整合性を含む
- **厳密認証方式**（SSL、Kerberos、RADIUS など）
- **プロキシ認証**

推奨事項 29：**高度なセキュリティを検討する**

Oracle JDBC ドライバは、高度なセキュリティ API を実装するクラスを組み込みます。暗号化と整合性に関するセキュリティ・パラメータは通常、`sqlnet.ora` ファイルに設定します。接続プロパティまたはシステム・プロパティとして設定することもできます。

データ暗号化とデータ整合性

企業ネットワークおよびインターネット経由で転送される機密情報を、暗号化アルゴリズムを使用して保護できます。このアルゴリズムによって、復号化キーを使用した場合のみ暗号解読できる形式に情報が変換されます。サポートされる暗号化アルゴリズムは、RC4、DES、3DES、AES などです。Oracle Advanced Security は、次のハッシュ・アルゴリズムを使用してセキュアなメッセージ・ダイジェストを生成し、このダイジェストをネットワーク経由で送信される各メッセージに含めます。このセクションのデータ暗号化とデータ整合性に関連する、接続レベルのプロパティを参照してください。

```
Properties connProps = new Properties();
// For Data Integrity Check
connProps.setProperty(OracleConnection.CONNECTION_PROPERTY_THIN_NET_CHECKSUM_TY
PES, "( MD5, SHA1, SHA256, SHA384 or SHA512 )");
connProps.setProperty(OracleConnection.CONNECTION_PROPERTY_THIN_NET_CHECKSUM_LE
VEL, "REQUIRED");
// For Data Encryption
connProps.setProperty(OracleConnection.CONNECTION_PROPERTY_THIN_NET_ENCRYPTION
_LEVEL, REQUIRED");
connProps.setProperty(OracleConnection.CONNECTION_PROPERTY_THIN_NET_ENCRYPTION
_TYPES,"(DES40C)");
// OracleDataSource – Oracle JDBC Connection
OracleDataSource ods = new OracleDataSource();
ods.setConnectionProperties(connProps);
// PoolDataSource – While Using UCP as connection pool
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionProperties(connProps);
```

推奨事項 30：**強力な認証を使用する**

基本的なユーザー名/パスワードよりも強力な認証を必要としている顧客では、Oracle Advanced Security を利用すると、RADIUS、Kerberos、証明書ベースの認証、トークン・カード、およびスマート・カードを使用してユーザーが外部認証を実行できます。Oracle JDBC ドライバは、次の強力な認証をサポートしています。

- SSL（証明書ベースの認証）
- Kerberos
- RADIUS

SSL 認証

Secure Sockets Layer（SSL）は、認証、データ暗号化、およびデータ整合性を提供します。SSL では、X.509v3 標準に準拠しているデジタル証明書を認証に使用し、公開鍵と秘密鍵のペアを暗号化に使用します。また、SSL では、データのプライバシと整合性を確保するために、秘密鍵の暗号化とデジタル署名を使用します。SSL 経由のネットワーク接続が開始されると、クライアントとサーバーは SSL ハンドシェイクを実行します。コード・サンプルおよび詳細については、ホワイト・ペーパー『*Oracle JDBC シン・ドライバを使用した SSL*』[20]を参照してください。以下に、SSL 認証のセットアップに使用する接続レベルのプロパティをいくつか示します。SSL のプロパティは、括弧内にプロパティ名を指定し、システム・レベルのプロパティとして設定することもできます。

[20] 『Oracle JDBC シン・ドライバを使用した SSL』@ www.oracle.com/technetwork/database/enterprise-edition/wp-oracle-jdbc-thin-ssl-130128.pdf

接続プロパティ名	説明
CONNECTION_PROPERTY_THIN_NET_AUTH ENTICATION_SERVICES	使用する認証サービスを指定します。値として SSL を使用します。
CONNECTION_PROPERTY_THIN_SSL_CIPH ER_SUITES（oracle.net.ssl_cipher_suites）	使用可能な暗号スイートのサブセットを有効にします。可能な限り高いレベルのセキュリティを確保するために、もっとも強いものから開始してもっとも弱いものまで、暗号スイートに優先順位を設定します。
CONNECTION_PROPERTY_THIN_SSL_SERV ER_DN_MATCH（oracle.net.ssl_server_dn_match）	サーバーの DN が一致しているかどうかをドライバによって検証します。許可される値は、ON、OFF、TRUE、または FALSE です。接続 URL の例： jdbc:oracle:thin:@(DESCRIPTION=(ADDRE SS=(PROTOCOL=tcps)(HOST=servername)(PORT=2484))(CONNE CT_DATA=(SERVICE_NAME=servicename))**(SECURITY=(SS L_SERVER_CERT_DN=\"CN=server_test, C=US\")**))
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_KEYSTORETYPE（javax.net.ssl.keyStoreType）	SSL によってサポートされる有効なタイプのキーストアを示します。例：JKS、PKCS12、SSO など。
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_KEYSTORE（javax.net.ssl.keyStore）	キーストアの場所を指定します。
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_KEYSTOREPASSWORD（javax.net.ssl.keyStorePassword）	データへのアクセス前にデータ整合性をチェックするために使用する、キーストアのパスワードを指定します。
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_TRUSTSTORETYPE（javax.net.ssl.trustStoreType）	SSL によってサポートされる有効なタイプの信頼ストアを示します。例：JKS、PKCS12、SSO など。
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_TRUSTSTORE（javax.net.ssl.trustStore）	信頼ストアの場所を指定します。
CONNECTION_PROPERTY_THIN_JAVAX_NE T_SSL_TRUSTSTOREPASSWORD （javax.net.ssl.trustStorePassword）	データへのアクセス前にデータ整合性をチェックするために使用する、信頼ストアのパスワードを指定します。
CONNECTION_PROPERTY_THIN_SSL_VERS ION（oracle.net.ssl_version）	SSL バージョン（1.2、1.1 など）を指定します。
CONNECTION_PROPERTY_WALLET_LOCATI ON（oracle.net.wallet_location）	Oracle Wallet の場所を指定します。ウォレットには、SSL によって使用される証明書と鍵が含まれます。listener.ora ファイルまたは tnsnames.ora ファイルには、次のものが含まれます。 WALLET_LOCATION=(SOURCE=(METHO D=FILE)(METHOD_DATA=(DIRECTORY=/ server/wallet/path))) SSL_CLIENT_AUTHENTICATION=TRUE
CONNECTION_PROPERTY_WALLET_PASSW ORD	ウォレットのパスワードを指定します。ウォレットの自動ログインが有効になっていない場合のみ必要になります。

Kerberos 認証

Kerberos プロトコルは、セキュアでないネットワーク接続経由でクライアントまたはサーバーが Kerberos のサーバーまたはクライアントに身元を証明できるように、強力な暗号化を使用します。Kerberos のアーキテクチャは、Key Distribution Center（KDC）と呼ばれる信頼された認証サービスに基づいています。Kerberos 環境のユーザーとサービスはプリンシパルと呼ばれ、各プリンシパルは KDC との間で秘密（パスワードなど）を共有します。ユーザー（HR など）やデータベース・サーバー・インスタンスをプリンシパルにできます。

Kerberos を使用するには、KDC の初期セットアップが必要です。Kerberos のサンプル・コードについては、JDBC 開発者ガイド [21]を参照してください。Kerberos のセットアップに関連する接続レベルのプロパティには、次のようなものがあります。

接続プロパティ名	説明
CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES	使用する認証サービスを指定します。KERBEROS を使用します。
CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_KRB5_CC_NAME	Kerberos 資格証明キャッシュの場所を指定します。
CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_KRB5_MUTUAL	Kerberos 相互認証をオンにするには、このプロパティを"true"に設定します。

RADIUS 認証

Remote Authentication Dial-In User Service（RADIUS）は、リモート認証とリモート・アクセスを行うための、もっともよく知られているクライアント/サーバー・セキュリティ・プロトコルです。トークン・カードやスマート・カードなどのさまざまな認証メカニズムを、RADIUS と組み合わせて使用できます。RADIUS のサンプル・コードについては、*JDBC 開発者ガイド* [22]を参照してください。以下に、RADIUS のセットアップに関連する接続レベルのプロパティを示します。

接続プロパティ名	説明
CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES	使用する認証サービスを指定します。RADIUS を使用します。

推奨事項 31：プロキシ認証を検討する

プロキシ認証 [19] は N 層認証とも呼ばれ、ユーザー認証に中間層を使用します。3 つの形式のプロキシ認証（ユーザー名、識別名、証明書）を使用してクライアントをセキュアにプロキシするように、中間層サーバーを設計できます。つまり、プロキシ認証では、1 つの JDBC 接続を他の JDBC 接続のプロキシとして機能させることができます。

例："ユーザー名のプロキシ認証"では、以下に示すように、"appuser"がプロキシ・ユーザー名である場合、"jeff"と"smith"が特定のロールで作成されたプロキシ対象のユーザーとなります。"jeff"と"smith"は"appuser"を使用してデータベースに接続でき、接続時に"appuser"が"jeff"と"smith"の認証を実行します。

```
Rem Connect as system user to grant necessary roles to the DB users "jeff"
and "smith"
connect system/manager
grant select_role, insert_role, delete_role to jeff;
grant select_role, insert_role, delete_role to smith;
```

[21] JDBC 開発者ガイド http://docs.oracle.com/database/121/JJUAR/toc.htm

[22] JDBC 開発者ガイド http://docs.oracle.com/database/121/JJUAR/toc.htm

```
Rem grant the users "jeff" and "smith" to connect through “appuser” with
specified roles
alter user jeff grant connect through appuser with role select_role,
insert_role;
alter user smith grant connect through appuser with role select_role,
insert_role;
```

プロキシ認証のセットアップに使用する接続レベルのプロパティには、次のようなものがあります。

接続プロパティ名	説明
PROXY_CERTIFICATE	プロキシ証明書を指定します。
PROXY_DISTINGUISHED_NAME	ユーザーの識別名を指定します。
PROXY_ROLES	アクセスをプロキシに対して許可する、ロールを指定します。
PROXY_USER_NAME	ユーザー名を指定します。
PROXY_USER_PASSWORD	ユーザー・パスワードを指定します。PROXY_USER_NAME と組み合わせて使用する必要があります。

接続管理コード・サンプルの **ProxySessionSample.sql** および **ProxySessionSample.java** を参照してください。

管理性を向上するための接続管理戦略

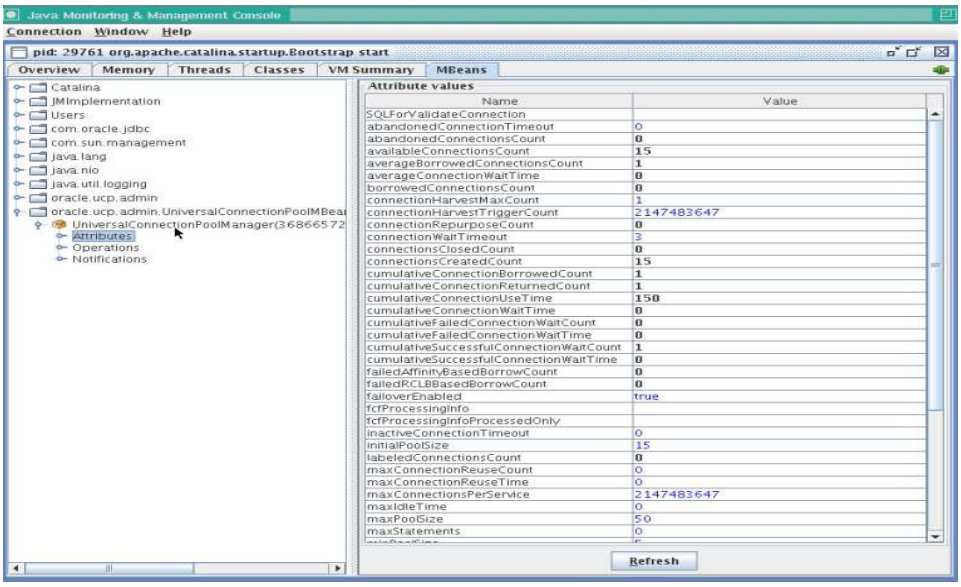
Java アプリケーションの管理性は、適切な監視ツールを使用して接続に関するアプリケーション健全性を監視する操作と、データベース・サーバーの維持と管理を容易にするベスト・プラクティスで構成されます。

4.1 単一インスタンスのデータベース

推奨事項 32：**JMX ベースの管理ツールを使用する**

JMX（Java Management Extensions）を使用し、作成、インストール、および実装したリソースを監視および管理します。JMX では、特定のリソースが、MBean（Managed Bean）と呼ばれる 1 つまたは複数の Java オブジェクトによって利用されます。また、MBean によって Universal Connection Pool を管理することもできます。**`UniversalConnectionPoolManagerMBean`** は、従来の接続プール・マネージャの機能をすべて備えたマネージャ MBean です。**`UniversalConnectionPoolMBean`** は、プール・プロパティとプール統計の動的構成に対応するプール MBean です。

JConsole（または MBean 対応ツール）を起動し、選択できるすべてのプロセスが表示されたら、UCP を使用するアプリケーションまたはプログラムのプロセス ID（pid）を選択します。以下に、JConsole での MBean スクリーンショット例を示します。



4.2 Oracle Real Application Clusters（Oracle RAC）

「4.1：単一インスタンスのデータベース」で説明したすべての接続管理戦略が、Oracle Real Application Clusters に適用されます。また、以下に示すように、SCAN を使用する必要があります。

推奨事項 33：**Single Client Access Name（SCAN）を使用する**

SCAN は、クライアントが Oracle データベース・クラスタにアクセスするための単一名を提供する、Oracle RAC 機能です。SCAN はクラスタのライフ・サイクルを通じて変更されないため、管理性が向上します。Oracle RAC データベースへのすべての接続について、クライアント接続文字列に SCAN を使用することを推奨します。

また SCAN は、ドメイン・ネーム・サーバー（DNS）または常に 3 つの IP アドレスに解決されるグリッド・ネーミング・サービス（GNS）のいずれかで定義されます。たとえば、10 のインスタンスがある場合、SCAN のセットアップでは*/etc/hosts* に 3 つの SCAN IP アドレスを入力します。接続がリクエストされると、Net Services はラウンドロビン方式でこれらの 3 つの IP アドレスから SCAN リスナーを取得します。SCAN（リモート）リスナーは、サービスを使用可能な、負荷のもっとも小さい 10 のインスタンス（ローカル・リスナー）のいずれかを選択します。

Scan IP's – contents of `/etc/hosts`

10.20.30.40 primaryscan.us.oracle.com primaryscan
10.20.30.41 primaryscan.us.oracle.com primaryscan
10.20.30.42 primaryscan.us.oracle.com primaryscan

SCAN を使用しない接続文字列：

この接続文字列では、4 つのホスト/ノードを使用し、ホストごとに、URL で明示的に示される固有のドメイン名を使用します。Oracle RAC に 10 のインスタンスがある場合、接続文字列に 10 のホスト名が含まれるため、エラーが生じやすくなり、管理に手間がかかる可能性があります。

```
jdbc:oracle:thin:@
(DESCRIPTION =
 (CONNECT_TIMEOUT= 15) (RETRY_COUNT=20) (RETRY_DELAY=3)
  (ADDRESS_LIST =
   (LOAD_BALANCE=ON)
    (ADDRESS = (PROTOCOL = tcp)(HOST = myhost-a-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = tcp)(HOST = myhost-b-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = tcp)(HOST = myhost-c-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = tcp)(HOST = myhost-d-vip)(PORT = 1521)))
 (CONNECT_DATA=(SERVICE_NAME= myorcldbservicename)))
```

SCAN を使用した接続文字列：

SCAN を使用すると、クライアント接続文字列が簡潔になり、管理性が向上します。クラスタに対してノード/インスタンスの追加や削除が行われた場合でも、接続文字列は変更されません。Oracle RACに10のインスタンスがある場合、SCAN名によって接続文字列が簡潔になり、管理も可能です。

```
jdbc:oracle:thin:@
(DESCRIPTION=
 (CONNECT_TIMEOUT=15)(RETRY_COUNT=20)(RETRY_DELAY=3)
  (ADDRESS_LIST =
    (LOAD_BALANCE=ON)
      (ADDRESS=(PROTOCOL=tcp)(HOST=primaryscan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))
```

4.3 Oracle DGまたはOracle ADG

「4.1 単一インスタンスのデータベース」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG に適用されます。

4.4 Oracle DGまたはOracle ADGとOracle RAC

「4.1 単一インスタンスのデータベース」および「4.2 Oracle Real Application Clusters（Oracle RAC）」で説明したすべての接続管理戦略が、Oracle DG または Oracle ADG と Oracle RAC に適用されます。

4.5 Oracle Global Data Services（Oracle GDS）

「4.1：単一インスタンスのデータベース」で説明したすべての接続管理戦略が、Global Data Services に適用されます。また、以下に示すように、グローバル・サービス・マネージャ（GSM）リスナーを使用する必要があります。

推奨事項 34：**グローバル・サービス・マネージャ（GSM）リスナーを使用する**

Oracle Global Data Services のグローバル・サービス・マネージャ・リスナーは、Oracle RAC データベースの SCAN に似ています。GDS 構成には、リージョンごとに複数のグローバル・サービス・マネージャが含まれます。GSM は、リアルタイム・ロード特性と、レプリケート・データベースに関するユーザー定義のサービス配置ポリシーを把握する、"グローバル・リスナー"です。

GSM リスナーを使用した接続文字列：

```
jdbc:oracle:thin:@(DESCRIPTION= (FAILOVER=on)
   (CONNECT_TIMEOUT=15) (RETRY_COUNT=20) (RETRY_DELAY=3)
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm1) (PORT=1571)))
      (ADDRESS_LIST=
         (LOAD_BALANCE=ON)
         (ADDRESS= (PROTOCOL= TCP) (HOST=myorclgsm2) (PORT=1572)))
   (CONNECT_DATA= (SERVICE_NAME=gold-cloud-service-name) (REGION=region1))
```

結論

このホワイト・ペーパーでは、Oracle Database 12c で JDBC と UCP を使用する場合の接続管理戦略について、総合的かつ実用面に重点を置いて説明しました。このホワイト・ペーパーでは、単一インスタンスのデータベース、Oracle Real Application Clusters、Oracle Multitenant、Oracle Data Guard、Oracle Active Data Guard、および Oracle Global Data Services などの各種データベース構成でパフォーマンス、可用性、スケーラビリティ、セキュリティ、および管理性を実現するためのベスト・プラクティス、構成、およびプロパティについて説明しました。Oracle Database 管理者、Java アーキテクト、および Web アプリケーション設計者は、このホワイト・ペーパーの戦略を活用すると、堅牢性、信頼性、パフォーマンス、スケーラビリティ、可用性、セキュリティ、および管理性に優れた Web アプリケーションを設計でき、ユーザー・エクスペリエンスを向上できます。このホワイト・ペーパーで言及している接続管理コード・サンプル一式は OTN[23]に掲載される予定であり、GITHUB（https://github.com/oracle/jdbc-ucp）にも掲載される予定です。

[23] コード・サンプルは現在 OTN にアップロードされていません。間もなくアップロードされる予定です。@ http://www.oracle.com/technetwork/database/features/jdbc/default- 2345085.html

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services