ORACLE DATABASE 12c

Oracle Database 12c Release 2（12.2.0.1）のJDBCとUCPによるパフォーマンス、スケーラビリティ、高可用性の実現

Oracle JDBC と Oracle Universal Connection Pool（UCP）

Oracle ホワイト・ペーパー 2017 年 3 月

ORACLE

免責事項

下記事項は、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。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないで下さい。オラクルの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。

目次

はじめに

世界でもっとも使用されているデータベースの最新世代である Oracle Database 12c Release 2（12.2.0.1）が Oracle Cloud およびオンプレミスで利用可能になり、今日の Web/モバイル・アプリケーションが直面している、パフォーマンス、スケーラビリティ、高可用性、セキュリティの問題に対するソリューションが提供されます。

高品質のサービスを実現する上で生じる周知の課題には、**可用性**と**管理性**を低下させずに顧客向けアプリケーションを数千ユーザーに**拡大**する、マルチテナント・アーキテクチャでデータベース・リソースを**最適化**する、アプリケーション・コードを変更せずにユーザーへの影響なしに計画メンテナンスを実行する、などの課題があります。

Oracle Database 12.2.0.1 はこれらの課題に対処するとともに、Java を使用する開発者に対して多数の役立つ機能を提供します。ソリューションの例をあげると、12.2.0.1 は、アプリケーションを簡単に拡張するのに役立つとともに、シングル・ポイント障害を排除するインフラストラクチャを提供する、**データベース・シャーディング**・アーキテクチャをサポートしています。また、**マルチテナント・アーキテクチャ**では、高いコストのかかるデータベース・リソースの使用を最適化する多面的な Universal Connection Pool（UCP）を利用しています。さらに、高速アプリケーション通知（FAN）イベントをサポートして計画メンテナンスをユーザーへの影響なしに実行できるように、JDBC ドライバが拡張されています。

このホワイト・ペーパーでは、Oracle Database 12c Release 2（12.2.0.1）、Oracle JDBC ドライバ（**ojdbc8.jar**）[1]、および Universal Connection Pool（**ucp.jar**）を使用して Java アプリケーションでこれらのソリューションを実装する方法について詳しく説明します。これらの新しいソリューションにより、Java 開発者、Java アーキテクト、または Oracle DBA はエンタープライズ Java 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、管理性を向上できます。

詳しくは、JDBC 開発者ガイド [2]および UCP 開発者ガイド [3]を参照してください。

クラウドのデータベースへのJava接続

クラウドの Oracle Database では、1 つの Oracle Database にアクセスして、その機能と処理をすべて利用できます。各クラウド・サービスに固有のセキュリティ要件があります。

以下の項では、これらのデータベース・クラウド・サービスの JDBC 接続とセキュリティ要件に対処しているクラウド・サービスをいくつか説明します。

[1] 12.2.0.1 の JDBC および UCP のダウンロード・ページ @ http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html
[2] JDBC 開発者ガイド @ https://docs.oracle.com/database/122/JJDBC/toc.htm
[3] Universal Connection Pool 開発者ガイド @ https://docs.oracle.com/database/122/JJUCP/toc.htm

Oracle Database Cloud Service（Oracle DBCS）

Oracle Database Cloud Service[4]では、エンタープライズ・アプリケーションに適したクラウド内でOracle Database をフルに活用できます。Oracle Net Services を使用してデータベース・クラウド・サービスに接続できます。セキュリティ要件として、データベース作成時にポート 1521 がブロックされます。データベースに接続する前に、ポート 1521 のブロックを解除する必要があります。また、Oracle DBCS ではコンピュート・ノードへの SSH アクセスが許可されるため、データベースを完全に制御できます。

詳しくは、『**Using Java Applications & IDEs with Oracle Database Cloud Service (DBCS)**』[5]の手順を参照してください。

Oracle Database Exadata Express Cloud Service

Oracle Exadata Express Cloud Service（Oracle EECS）[6]は、中小サイズのデータに適した完全管理のデータベースです。プラガブル・データベース（PDB）コンテナ・テクノロジーを利用しており、数分でプロビジョニングが完了し、Oracle Exadata エンジニアド・システム上で実行されます。

セキュリティ要件として、TLSv1.2 プロトコルを使用したセキュア接続が必要です。TLSv1.2 は、12.2.0.1 の JDBC ドライバを使用する場合はデフォルトで有効になります。ただし、12.1.0.2 を使用する場合は、別のプロパティ `oracle.net.ssl_version=1.2` をシステム・プロパティまたは接続プロパティとして設定する必要があります。JDBC ドライバは、Java キーストア（JKS）ファイルを使用することでセキュア接続を確保します。サービス・コンソールで、JKS ファイル（`keystore.jks` と `trustore.jks`）および必要な構成ファイル（`tnsnames.ora`）をダウンロードする必要があります。

JDBC ドライバと UCP を使用して Oracle Database Exadata Express Cloud Service に接続する手順について詳しくは、OTN の **Java SE、Java EE コンテナ、Java Cloud Service の Java/UCP** 接続に関するページ[7]を参照してください。

また、JDeveloper、Eclipse、Intellij、NetBeans などの Java Developer ツールを Oracle Database Exadata Express Cloud Service にシームレスに接続してスキーマを参照できます。手順については、OTN ページの **Java IDE（JDeveloper、NetBeans、Eclipse、Intellij）の JDBC/UCP 接続に関するページ**[8]を参照してください。

JDBCとUCP - 最新のJava標準のサポート

Java SE 8 サポートと JDBC 4.2

[4] Oracle Database Cloud Service のガイド @ http://docs.oracle.com/cloud/latest/dbcs_dbaas/index.html
[5] Oracle Database Cloud Service への JDBC 接続手順 @ http://www.oracle.com/technetwork/database/application-development/jdbc/documentation/default-3396167.html
[6] Oracle Database Exadata Express Cloud Service のガイド @ http://docs.oracle.com/cloud/latest/exadataexpress-cloud/CSDBP/toc.htm
[7] Java SE、Java EE コンテナ、Java Cloud Service の JDBC/UCP 接続 @ http://www.oracle.com/technetwork/database/application- development/jdbc-eecloud-3089380.html
[8] Java IDE（JDeveloper、NetBeans、Eclipse、Intellij）の JDBC/UCP 接続 @ http://www.oracle.com/technetwork/database/application-development/jdbc/index-3093936.html

Oracle Database 12c Release 2（12.2.0.1）の JDBC ドライバと UCP は、Java SE 8 の JDBC 4.2[9]で導入された新機能をサポートしています。

JDBC ドライバと UCP は、Java SE 8 のラムダ式と Builder パターンを使用しています。新しい API には、Builder パターンを使用する API もあります。Builder パターンのサンプルについては、シャーディングとマルチテナントの項を参照してください。

以下に、JDBC 4.2 に関連する新しいメソッドをいくつか示します。

`INTEGER.MAX_VALUE` を超える数の行を返す必要がある場合に、これらのメソッドを使用できます。

- `executeLargeUpdate(String sql)`
- `executeLargeUpdate(String sql, int[] columnIndexes)`
- `executeLargeUpdate(String sql, String[] columnNames)`
- `getLargeMaxRows()`
- `getLargeUpdateCount()`
- `setLargeMaxRows(long max)`

また、DB 12.1 の JDBC ですでに提供されている `getObject()`に加えて、`setObject()`メソッドもサポートされています。

例：次のコード部分は、`setObject(…)`と `getObject(…)`の使用方法を示しています。

```
// Statement and ResultSet are AutoCloseable and closed automatically.
try (PreparedStatement preparedStatement = connection.prepareStatement("SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEES WHERE EMPNO = ?"))
    {
        preparedStatement.setObject(1, Integer.valueOf(id));
        try (ResultSet resultSet = prepareStatement.executeQuery()) {
           System.out.println("FIRST_NAME" + "    + "LAST_NAME");
           System.out.println("--------------------");
           while (resultSet.next())
                        System.out.println(resultSet.getObject(1, String.class) + " "
                                        + resultSet.getObject(1, String.class) + " ");
            }
     }
```

JDBC と UCP による新しいデータ型のサポート

Oracle Database リリース 12.2.0.1 では、JDBC によって Java プログラマーは新しい PLSQL ブール・データ型を利用できます。

[9] JDBC 4.2 の仕様 @ https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/jdbc_42.html

PLSQLのブール・データ型のサポート

PLSQL のブール・データ型は true 値または false 値を使用するため、数値（0/1）を使用する SQL のブールとは異なります。12.2.0.1 の JDBC ドライバは PLSQL のブールをサポートしており、PLSQL ブロックへのバインド値として使用できます。 `OracleTypes.PLSQL_BOOLEAN` を IN、OUT、または IN OUT パラメータに使用できます。

例：

```
// Shows how a PLSQL procedure is used to find if the
// first number is bigger than the second number passed as input.
String sql = "{? = call function_bigger (?, ?)}";
CallableStatement cstmt = conn.prepareCall(sql);
// Test boolean as output in a function
cstmt.registerOutParameter(1, OracleTypes.PLSQL_BOOLEAN);
// Test values for true
cstmt.setInt(2, 9);
cstmt.setInt(3, 7);
// Execute the callable statement
cstmt.execute();
// Returns a boolean (true/false)
boolean biggerThan = cstmt.getBoolean(1);
cstmt.close();
```

Oracle JDBCおよびUCPによるJavaパフォーマンス

Oracle Database 12c Release 2（12.2.0.1）では、JDBC ドライバと UCP に新しいパフォーマンス強化機能が組み込まれています。これらの強化機能として、データベース・ノードの優先度付け解除、UCP の再設計、およびネットワーク・レベルと接続プール・レベルでの強化機能があります。

WAN 経由でのネットワーク圧縮

ネットワーク帯域幅に制約がある Wide Area Network（WAN）環境で、データ圧縮が可能になっており、大量のデータを適切なパフォーマンスで送信できます。セッション・データ・ユニット（SDU）バッファ内のデータを圧縮することで圧縮を実現しているため、ネットワーク経由で SQL 問合せと結果を送信するのに必要な時間が短縮されます。データを圧縮すると、使用する帯域幅が少なくなります。圧縮プロセスはアプリケーション・レイヤーに対して透過的に実行されます。

例：

```
OracleDataSource ds = new OracleDataSource();
Properties prop = new Properties();
prop.setProperty("user","user1");
prop.setProperty("password",<password>);

// Enabling Network Compression
prop.setProperty("oracle.net.networkCompression","on");
//Optional configuration for setting the client compression threshold.
prop.setProperty("oracle.net.networkCompressionThreshold","1024");

ds.setConnectionProperties(prop);
ds.setURL(url);
Connection conn = ds.getConnection();
```

データベース・ノードの優先度付け解除

Oracle Database 12c Release 2（12.2.0.1）では、JDBC ドライバがさらにスマートになり、接続確立時に停止中であったインスタンスについて優先度付けを解除できます。たとえば、A、B、C の 3 つのインスタンスがあり、A が停止中である場合、接続はまずインスタンス B とインスタンス C から割り当てられて、最後にインスタス A への接続が試行されます。優先度付け解除は、有効期限（デフォルトでは 10 分に設定）に達するまで有効になります。有効期限に達すると、インスタンス A の優先度付け解除が無効になり、3 つのすべてのインスタンスから同等に接続が割り当てられます。システム・プロパティ `oracle.net.DOWN_HOSTS_TIMEOUT` を使用してデフォルトの有効期限をオーバーライドできます。

UCP接続状態チェック頻度

Universal Connection Pool では、チェックアウト時に接続を評価するように指定する `setValidateConnectionOnBorrow(Boolean)`プロパティが提供されていました。接続検証は、デフォルトでは無効になります。これは重要な機能ですが、接続をチェックアウトするのに要する時間が長くなる可能性があります。そのため、パフォーマンスを向上し、接続のチェックアウトに要する時間を最小限に抑えるために、新しい UCP プロパティ `setSecondsToTrustIdleConnection(int)`を使用できます。このプロパティに適切な値を設定すると、*最近使用したデータベース接続と最近テストしたデータベース接続が信頼されて接続検証がスキップされます。これにより、接続検証の頻度が減り、パフォーマンスが向上します。

接続プールの再設計

待機時間なしのUCP

大量のスレッドが伴う同時実行性の高い環境では、パフォーマンスが極めて重要です。このような環境で接続のチェックアウトに要する時間を短縮するように、UCP が再設計されています。接続の流用にかかるコストを低減するように、おもな接続プーリング操作がいくつか微調整されています。これらのパフォーマンス向上は 12.1.0.2 の一部として可能になっています。オラクル社内のベンチマークによると、改良された新しい待機時間なしの UCP では、古いバージョンの UCP よりもパフォーマンスが大幅に向上することが示されています。詳しくは、JavaOne のプレゼンテーション[10]を参照してください。

多次元ツリー検索

接続要求にインスタンス名、サービス名、データベース名などの追加のプロパティが含まれている場合があります。これらのプロパティ/ラベルを接続に関連付けると、接続の再初期化にかかる時間とコストを回避できます。ラベルによって特定の接続をさらに高速に取得するというのは、高パフォーマンスのスケーラブルなシステムでは難しい問題です。直接関連しない複数のキーを用いて UCP が検索を実行している場合は、マップなどの通常の Java コレクションは適切ではありません。これを解決するため、多数のグラフィカル・アプリケーション、AI アプリケーション、マシン学習アプリケーション、科学技術アプリケーションで知られている KD ツリーに基づいてパフォーマンスを向上するように、検索メカニズムが再設計されています。最適化された接続プールでは、プールのサイズが増加した場合でも、常に接続の流用がより高速に実行されることが示されています。

JDBCとUCPによるスケーラビリティ

顧客やデータの数が急増しているため、スケーラビリティが Java アプリケーションの重要な要件となっています。大量のデータを処理でき、管理が容易で運用時のコスト効率に優れた、適切なアーキテクチャを選択する必要があります。

12.2.0.1 では、JDBC ドライバと UCP の両方が、線形のデータベース・スケーラビリティと管理性の向上を実現するソリューションである Oracle Sharding をサポートしています。

12.2.0.1 では、クライアント側接続プールである UCP がマルチテナント環境に"共有プール"を提供するため、データベース・リソースが最適化されます。UCP と同様に、サーバー側接続プールであるデータベース常駐接続プール（DRCP）は、プロキシ接続の共有と複数ラベル/タグのサポートを実現する新しい強化機能を備えているため、接続の取得が高速化されます。

JDBCとUCPのシャーディング・サポート

データやトランザクション、ユーザーが日々増加している最新の Web アプリケーションでは、*Oracle Sharding* を利用するとパフォーマンスや可用性を低下させずにスケーリングできます。

Java アプリケーションにシャード・データベースを使用する場合、新しいシャーディング API を使用する必要があります。シャードは、1 つまたは複数のシャーディング・キーと、必要に応じて 1 つのスーパー・シャーディング・キーに基づいて作成されます。JDBC および UCP アプリケーショ

[10] JavaOne による待機時間なしの UCP @ http://www.oracle.com/technetwork/database/application-development/con2158-javaoneucpsession-2769404.pdf

ンでシャーディング・キーと、必要に応じてスーパー・シャーディング・キーを作成し、これらのキーを使用して特定のシャードへの接続を要求する必要があります。

Universal Connection Pool（UCP）は、シャーディングの効率的なサポートを提供します。UCP がシャード・テクノロジーをキャッシュするため、すべての接続要求にシャード・ディレクタを使用せずに済みます。このように、UCP はシャード・ディレクタとして機能し、シャード・データベースへの高速パスを提供します。また、UCP は、チャンク移動に関連するイベントにサブスクライブすることで、シャードまたはチャンクのライフ・サイクル管理を透過的に処理します。

以下のコード・スニペットは、シャーディング API の例を示しています。詳しくは、JDBC 開発者ガイド [11]および UCP 開発者ガイド [12]を参照してください。

例：

```
// Get the PoolDataSource for UCP
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
Connection connection;
// Set the connection factory first before all other properties
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL(DB_URL);
pds.setUser(DB_USER);
pds.setPassword(DB_PASSWORD);
pds.setConnectionPoolName("UCP_SHARDING_POOL");
// Set the initial number of connections created when UCP is started.
pds.setInitialPoolSize(5);
// Set the minimum number of connections maintained by UCP at runtime.
pds.setMinPoolSize(5);
// Set the maximum number of connections allowed on the connection pool.
pds.setMaxPoolSize(20);
// set max connection per shard to ensure fair use of the pool
pds.setMaxConnectionsPerShard(10);
// Build the Sharding key by passing the email id
OracleShardingKey shardKey =     pds.createShardingKeyBuilder()
                                    .subkey(email, OracleType.VARCHAR2)
                                    .build();
// Get the connection by passing  the sharding key
connection = pds.createConnectionBuilder()
                .shardingKey(shardKey)
                .build();
// Perform a database operation
doSQLWork(connection, email);
```

マルチテナント・データソースのシャード・プール

Oracle Database 12c Release 2（12.2.0.1）では、UCP によって、Oracle データベース・マルチテナント・アーキテクチャにアクセスするマルチテナント Java アプリケーションが共通の接続プールにアクセスできます。空き接続がない場合は、UCP によって複数のテナント（PDB）間で接続を再利用または再割当てできるため、パフォーマンスとスケーラビリティが向上します。

[11] JDBC 開発者ガイド @ https://docs.oracle.com/database/122/JJDBC/toc.htm

[12] Universal Connection Pool 開発者ガイド@ https://docs.oracle.com/database/122/JJUCP/toc.htm

UCP でサポートされているユースケースまたは手法は、以下の 2 つです。

I. 単一のデータソースをすべてのテナントで共有

II. テナントごとに 1 つのデータソースを使用

ユースケース 1：単一のデータソースをすべてのテナントで共有

この構成では、複数のテナントが共通のデータソースを使用します。共有プールを使用し、テナントごとに異なるサービス名を使用して、各テナントへの接続が行われます。以下に示すように、新しい API を使用して共有プールからの接続を取得します。共有プールを利用するには、共通ユーザー（例：c##commonuser）を使用する必要があります。

例：

```
PoolDataSource multiTenantDS = PoolDataSourceFactory.getPoolDataSource();
...
// password enabled role for tenant-1
Properties tenant1Roles = new Properties();
tenant1Roles.put("tenant1-role", "tenant1-password");

//Create Connection to Tenant-1 and apply the tenant specific PDB roles.
Connection tenant1Connection =
        multiTenantDS.createConnectionBuilder()
                .serviceName("tenant1Svc.oracle.com")
                .pdbRoles(tenant1Roles)
                .build();

// password enabled role for tenant-2
Properties tenant2Roles = new Properties();
tenant1Roles.put("tenant2-role", "tenant2-password");

//Create Connection to Tenant-2 and apply the tenant specific PDB roles.
Connection tenant2Connection =
        multiTenantDS.createConnectionBuilder()
                .serviceName("tenant2Svc.oracle.com")
                .pdbRoles(tenant2Roles)
                .build();
```

ユースケース 2：テナントごとに 1 つのデータソースを使用

この構成では、マルチテナント・アプリケーションに、テナントごとに特定のサービス名が指定された別々のデータソースと、共通の共有接続プールが割り当てられます。このシナリオでは、構成を XML ファイルに指定する必要があります。

この機能を使用するための前提条件には、以下のようなものがあります。

(a) 接続を再割当てする共通ユーザー（例：c##commonuser）。

(b) アプリケーション・サービスを使用する必要があり、そのアプリケーション・サービスが同種である必要があります。同種のサービスとは、アプリケーション・コンティニュイティ（AC）、トランザクション・ガード（TG）、DRCP などの設定が同じであるサービスのことです。スイッチ・サービス機能を透過的に使用することで、同種のサービスに対して接続が再割当てされます。

(c) 次のような共有プール構成を指定した UCP XML 構成ファイル。

`connection-pool-name`、`shared=true`, `data-source`、
`max-connections-per-service` など

UCP XML 構成ファイルの使用方法について詳しくは、「JDBC および UCP の管理性」を参照してください。

プロパティ“`max-connections-per-service`”を使用していることを確認します。このプロパティを使用すると、テナントがチェックアウトできる接続の数を制限できるため、テナント間で接続を均等に配分できます。

以下に示すように、接続の確立、共有プールの再構成、データソースの再構成を実現するために、新しいメソッドが導入されています。

```
// Get the first datasource instance "pds1" from the UCP config XML
PoolDataSource pds1 = PoolDataSourceFactory.getPoolDataSource("pds1");
Connection pds1Conn = pds1.getConnection();

// Get the second datasource instance "pds2" from the UCP config XML.
PoolDataSource pds2 = PoolDataSourceFactory.getPoolDataSource("pds2");
Connection pds2Conn = pds2.getConnection();

PoolDataSource multiTenantDS = PoolDataSourceFactory.getPoolDataSource();
// Reconfigures the datasource properties
multiTenantDS.reconfigureDataSource(Properties prop)
// Reconfigure connection pool("pool1") using the new properties

Properties newPoolProps = new Properties();
newPoolProps.put("initialPoolSize", <newInitialPoolSizeValue>);
newPoolProps.put("maxPoolSize", <newMaxPoolSizeValue>);
UniversalConnectionPoolManager ucpMgr =
UniversalConnectionPoolManagerImpl.getUniversalConnectionPoolManager();
ucpMgr.reconfigureConnectionPool("pool1", newPoolProps);
```

```
// Configure a new datasource(pds3) to running pool
Properties dataSourceProps = new Properties();
dataSourceProps.put("serviceName", <serviceName>);
dataSourceProps.put("connectionPoolName", <poolName>);
dataSourceProps.put("dataSourceName", <dataSourceName>); PoolDataSource pds3 =
                    PoolDataSourceFactory.getPoolDataSource(dataSourceProps);
```

DRCPの機能拡張

DRCPプロキシ・セッション共有

Oracle Database 12c Release 2（12.2.0.1）以降では、同じデータベース・ユーザーに属しているが別のプロキシ・ユーザーに属している DRCP セッションをユーザー間で共有できるようになりました。以前のリリースでは、プロキシ・ユーザーは別のプロキシ・ユーザーに属しているアイドル・セッションを、両方のプロキシ・ユーザーが同じデータベース・ユーザーにマップされている場合でも使用できませんでした。このリリースでは、この制限がなくなりました。

DRCPによる複数タグ付け

接続のタグ付けとは、タグが接続に適用されて、そのタグを使用して同じ接続が取得されるメカニズムです。接続のタグ付けにより、特定のセッションをさらに高速に取得できるため、セッション・プーリングが強化されます。

Oracle Database 12c Release 2（12.2.0.1）では、DRCP は複数のタグ付けをサポートしています。複数のプロパティを 1 つの接続に関連付けて、これらのプロパティを使用しながら、同じプロパティを持つ接続を取得できます。この機能を有効にするには、`oracle.jdbc.UseDRCPMultipletag` 接続プロパティを `TRUE` に設定します。

Oracle JDBCおよびUCPによるJava高可用性

Oracle Database は、継続的なサービスを提供することを目標としています。Oracle Real Application Clusters（Oracle RAC）、Data Guard（DG）、Active Data Guard（ADG）、ドライバ、および接続プールは、この目標を達成するための構成要素です。12.2.0.1 では、高可用性を強化するために新機能が追加されています。

ドライバによる高速アプリケーション通知（FAN）のサポート

Oracle Database 12c Release 2（12.2.0.1）では、JDBC ドライバが Oracle RAC FAN イベントをサポートしているため、サード・パーティの接続プールを使用するアプリケーションが 12.2 JDBC ドライバを通じて高可用性機能を利用できます。

12.2 JDBC ドライバを使用して FAN を有効にするには、以下の 3 つの簡単な手順を実行します。

(1) クラスパスに **simplefan.jar**、**ons.jar**、および **ojdbc8.jar** があることを確認します。3 つの jar すべてが 12.2.0.1 バージョンのものである必要があります。

(2) 安全なドレイン・ポイントを設定し、ドライバによって接続を安全に終了できるようにします。

(a) サード・パーティの接続プールで、次の安全なドレイン API を使用する必要があります。

- `java.sql.Connection.isValid(int timeout)`
- `oracle.jdbc.OracleConnection.pingDatabase()`
- `oracle.jdbc.OracleConnection.pingDatabase(int timeout)`
- `oracle.jdbc.OracleConnection.endRequest()`

(b) 標準の JDBC および Oracle JDBC 拡張機能 EXECUTE***コールの場合、検証の SQL に SQL ヒント（`/*+ CLIENT_CONNECTION_VALIDATION */`） をコメント以外の最初のトークンとして含める必要があります。

XAデータソースでのアプリケーション・コンティニュイティ

アプリケーション・コンティニュイティ（AC）は、データベース・インスタンスの計画外停止時に、実行中のデータベース処理を透過的にリカバリするために、Oracle Database 12c で導入されました。停止は修復後のシステムや通信、ハードウェア、または構成変更に関連して発生する可能性があります。AC により、すべてのデータベース要求が保護されます。

Oracle Database 12c Release 2（12.2.0.1）では、JDBC ドライバおよび UCP により、ローカル・トランザクションが含まれた XA データソース（`javax.sql.XADataSource`）に対応するようにアプリケーション・コンティニュイティ（AC）のサポートが拡張されています。

XA データソースのサポートに関して、以下のことを考慮してください。

(a) AC による XA 対応データソースのサポートは、グローバル/XA トランザクションに昇格可能なローカル・トランザクションに制限されます。

(b) 接続がグローバル/XA トランザクションに参加する場合または XA 操作に含まれる場合、再実行機能は無効になります。

(c) アプリケーションは、実行時に引き続き XA 操作を実行できますが、AC による保護は適用されません。

(d) XA/グローバル・トランザクションをローカル・トランザクションに切り替えた場合、AC は自動的には有効になりません。

この機能を利用するには、アプリケーションで XA 再実行ドライバ `oracle.jdbc.replay.OracleXADataSource` を使用する必要があります。詳しくは、JDBC 開発者ガイドのサンプル・コードを参照してください。

XAデータソースでのトランザクション・ガード（TG）

COMMIT の実行時に通信障害などの予期しない障害が発生するというシナリオでは、トランザクション結果を確定するのは非常に難しいです。トランザクション・ガード（TG）は、トランザクション結果をスケーラブルに保証して確定することで、この問題を解決します。

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`）でタグ付けされるため、障害時にアプリケーションがこの識別子を使用してトランザクションがコミットされたかどうかをチェックできます。このようにして、トランザクション・ガードはすべてのトランザクションが 1 回のみ実行されるようにし、重複するトランザクションがアプリケーションによって送信されないようにします。

Oracle Database 12c Release 2（12.2.0.1）以降、トランザクション・ガードは XA トランザクションをサポートしており、1 フェーズ・コミットの最適化、読取り専用の最適化、および昇格可能な XA に対応しています。トランザクション・ガードと XA により、XA トランザクションのリカバリ可能な停止後に、トランザクションを安全に再実行できます。XA サポートの追加により、トランザクション・マネージャでは、トランザクション・ガードを使用して再実行が可能になりました。

高速アプリケーション通知（FAN）API による高可用性の実現

Oracle Web Logic Service（Oracle WLS）、Active Grid Link（AGL）、Universal Connection Pool（UCP）などの Oracle 機能を使用せずにサード・パーティの接続プールとオラクル以外の Java コンテナを使用するというシナリオで、Oracle Database の高可用性機能を利用して応答性に優れたアプリケーションを構築したいと思いませんか。Oracle RAC 高速アプリケーション通知（FAN）API[13]を利用すると、これを実現できます。クラスパスに **simplefan.jar** ファイルおよび **ons.jar** ファイルがあることを確認します。

FAN イベントは、サーバー・レベルまたはノード・レベルの FAN イベントをサブスクライバ（接続プール、ドライバ）に知らせるために、クラスタによって送信される通知です。サポートされる FAN イベントは、Service UP、Service DOWN、Node UP、Node DOWN、Planned DOWN、Load Balancing Advisory です。

以下に、DOWN イベント・ペイロードのサンプルを示します。

```
VERSION=1.0 event_type=INSTANCE service=myorclservce instance=INSTANCE1
database=myorcldb db_domain=us.oracle.com host=myorclhost status=DOWN
reason=FAILURE timestamp=2016-11-01 19:32:43 timezone=-08:00
```

Oracle JDBCとUCPによるJavaアプリケーションのセキュリティ

クラウド上の Oracle Database には、データを保護するために高度なセキュリティが必要です。強固なセキュリティを適用するために、Oracle Database Exadata Express Cloud Service へのデータベース接続では、キーストア（JKS）ファイルを使用することで SSL 暗号化が実行され、TLSv1.2 を使用することでセキュリティが強化されます。12.2.0.1 では、TLSv1.2 はデフォルトでサポートされています。

TLSv1.1 および TLSv1.2 のサポート

Java SE 7 および Java SE 8 では、SSLv3、TLSv1、TLSv1.1、および TLSv1.2 の各プロトコルをサポートしています。Oracle JDBC Thin ドライバのバージョン 12.1.0.2 は元々、TLSv1.1 または TLSv1.2 をサポートするように拡張されています。12.1.0.2 の JDBC ドライバを使用する場合、TLSv1.2 を使用するには、プロパティ `oracle.net.ssl_version=1.2` を明示的に設定する必要があります。ただし、Oracle Database 12c Release 2（12.2.0.1）の JDBC ドライバでは、TLSv1.2 はデフォルトでサポートされているため、このプロパティは必要ありません。

[13] Oracle FAN API @ https://docs.oracle.com/database/122/JAFAN/toc.htm

Oracle JDBCとUCPによるJavaアプリケーションの管理性

アプリケーションを簡単に管理できる必要があります。12.2.0.1 では、UCP を XML 構成ファイルで構成でき、すべての接続プール・プロパティを XML 属性として定義できます。"マルチテナント・アーキテクチャ用の共有プール"など、一部の新機能には XML 構成ファイルが必要です。

UCP XML 構成ファイル

この新機能により、Java コンテナの接続プール・プロパティを 1 か所で定義できるようになりました。そのため、コンソールでプールを更新したり初期化したりする必要がなくなりました。すべてのプール・プロパティを XML ファイルに定義できます。

UCP XML 構成ファイルの実際のパスは、システム・プロパティ `oracle.ucp.jdbc.xmlConfigFile="file:/user_directory/ucp.xml"`で指定する必要があります。最初の XML 構成ファイルの場所は、URI として指定する必要があります。接続プールを作成するために必要となる特定のプール・プロパティを確認するには、`ucp.jar` ファイルに含まれている `configuration.xsd` スキーマ・ファイルを確認します。

例：

```
<ucp-properties>
<connection-pool
connection-pool-name="pool1"
connection-factory-class-name="oracle.jdbc.pool.OracleDataSource"
url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=myhost)
(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=myorcldbservice)))"
user= "C##1"
password= "password"
initial-pool-size="10"
min-pool-size="2"   max-pool-size="30"
max-connections-per-service="15" shared="true" >
<!-- First Datasource properties -->
<data-source data-source-name="pds1" service="pdb1" description="pdb1 data
source" />
<!-- Second Datasource properties -->
<data-source data-source-name="pds2" service="pdb2" description="pdb2 data
source" />
</connection-pool>
</ucp-properties>
```

システム変更番号（SCN）の拡張

システム変更番号（SCN）は、トランザクションにタイムスタンプを設定するために使用される、Oracle データベースの内部クロックです。SCN の上限は 6 バイトであるため、トランザクション数が非常に多い場合は限界に達する可能性があります。トランザクションのスループット増加に対処しながら、限界に達することがないように、SCN が 6 バイトから 8 バイトに拡張されています。これは、Java アプリケーションには透過的な内部変更です。

結論

このホワイト・ペーパーでは、Web アプリケーションが直面している課題に対処するために利用できる、JDBC ドライバと UCP のユースケースと新機能について説明しました。これらの新機能として、クラウド内の Oracle Database への Java 接続、JDBC による最新の Java 標準と新しいデータ型のサポート、WAN 経由でネットワーク圧縮を有効にする方法とタイミング、新しい UCP パフォーマンス強化機能、シャーディング・アーキテクチャでの JDBC/UCP API の使用、マルチテナント環境でのデータベース接続の最適化、アプリケーション・コードの変更が不要な計画メンテナンスの実現、アプリケーション・コンティニュイティによる XA データソースのサポート、FAN API の対応範囲の拡大、TLSv1.2 と TLSv1.1 のサポート、XML 構成ファイルを使用したプールの作成による管理性の向上があります。これらの機能を利用して、高いパフォーマンス、スケーラビリティ、可用性、セキュリティ、管理性を実現するように Java アプリケーションを強化できます。

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


Oracle is committed to developing practices and products that help protect the environment