Oracle Data Provider For .NET 11g

Oracle テクニカル・ホワイト・ペーパー
2007 年10 月

ORACLE

Oracle Data Provider For .NET 11*g*

Oracle Corporation 発行「Oracle Data Provider For .NET 11*g*」の翻訳版です。

はじめに

Oracle 製品を使用する大きなメリットの 1 つに、複数のプログラミング・フレームワークのサポートがあります。.NET、Java/J2EE、PHP、および C/C++アプリケーションがサポートされており、すべての開発者は Oracle の高度なデータベース機能を使用でき、開発部門で真の柔軟性が実現します。オラクルの各データ・アクセス・ドライバは各フレームワークのパフォーマンスを最大にし、最新のデータベース機能を利用できるように設計されています。

.NET 領域内で、オラクルは、Oracle データベースでアプリケーションを開発する多くの製品を提供しています。これには、Oracle Developer Tools for Visual Studio、Oracle Database Extensions for .NET、Oracle Providers for ASP.NET、Oracle Data Provider for .NET（ODP.NET）が含まれます。

ODP.NET は、Oracle データベースのネイティブな.NET データ・アクセス・プロバイダです。.NET Framework 1.x、.NET Framework 2.0、およびそれ以降のリリースに標準の ADO.NET データ・アクセスを提供します。別の ADO.NET プロバイダを使用している開発者は、基本的な ODP.NET データ・アクセスで新しく学習する内容がほとんどありません。Microsoft Data Access Application Blocks（DAAB）for .NET などの既存の製品と組み合わせて使用できます。このため、ADO.NET 開発者は、迅速に Oracle データソースの使用を開始できます。

ODP.NET は、一般的な基本要素をほかの ADO.NET プロバイダと共有しますが、ODP.NET 特有の価値は Oracle データベースとの緊密な統合です。ODP.NET は、Oracle Real Application Clusters（Oracle RAC）チューニング、高度なセキュリティ、複雑なデータ型などのデータベースの特有な多くの機能を.NET 開発者に公開します。これらの機能を使用すると、.NET 中間層で Oracle データベース特有の機能を活用できます。

このホワイト・ペーパーでは、ODP.NETデータ・アクセスとその特有な機能を中心に説明し、Oracle Database 11*g*および以前のリリースの一部である.NET開発機能を取り上げます。多くのODP.NET 11*g*機能は、Oracle Database 10*g*およびOracle Database 9*i*などの古いOracleデータベース・サーバー・リリースでも使用できます。このホワイト・ペーパーでは、Oracle Database 11*g* on Windowsのあとにリリースされ たODP.NET機能（ODP.NET 11.1.0.6.20 など）は説明しません。

Oracle Corporation 発行「Oracle Data Provider For .NET 11*g*」の翻訳版です。

Oracle Data Provider for .NET

Oracle Data Provider for .NETによって、Oracleデータベースへの高速なデータ・アクセス・パフォーマンスが実現します。Oracle Database 11*g*で使用できる最新のデータベース機能がサポートされます。

ほかの.NET データ・プロバイダと同様に、C# .NET、Visual Basic .NET、ASP.NET を含む.NET アプリケーションで ODP.NET を呼び出すことができます。ODP.NET は、中間層でもっとも一般的に採用されていますが、.NET ストアド・プロシージャを通じてデータベース・サーバー内でも使用できます。

また、ほかのどの.NET データ・プロバイダよりも優れた Oracle データベース機能のパフォーマンスとアクセスを提供します。ODP.NET は、とくに Oracle データベース機能を最大限に利用するために設計されたものです。

ODP.NET 11*g*は、Oracleデータベースの既存の機能を拡張し、Oracle Database 11*g*の新しいパフォーマンス機能を導入しています。このため、新しいデータベース・アプリケーションを配置するか既存のアプリケーションを拡張するかに関係なく、すべての開発者が最新のODP.NETバージョンを使用できます。これらの特有なOracle機能には、データ・アクセス・パフォーマンス・チューニング、データベース変更通知、RAC接続プーリング、XMLサポート、ネイティブなOracleデータ型サポート、さらにこのホワイト・ペーパーで後述するほかの機能が含まれます。

パフォーマンス

ODP.NET は、多くのパフォーマンス・チューニング機能を搭載し、データの取得および変更を最適化します。これらのチューニング・オプションには、接続プーリング、ステートメント・キャッシュ、LOB データ型の使用、PL/SQL 連想配列の採用が含まれます。64 ビットの ODP.NET は、Windows x64 および Windows Itanium でサポートされます。

ほかのプロバイダと比較した ODP.NET の差別化要因は、標準のパフォーマンスおよび多くのチューニング・オプションです。内部での多くの最適化によって、特定のパフォーマンス・コーディングを実行しなくても Oracle データソースへの高速な.NET アクセスが自動的に保証されます。また、ODP.NET には、特定のデータ取得とデータ更新シナリオに使用されるチューニングが可能な多くのパラメータがあります。LOB や REF Cursor などの従来の大きいデータ型を取得および操作するために、こうした多くの最適化がおこなわれました。

接続プーリングおよびステートメント・キャッシュ

もっとも幅広く使用されているパフォーマンスの最適化は接続プーリングです。これは、多くのユーザーがデータベースとの接続および接続の解除を使用するアプリケーションにとって重要です。ODP.NET は、接続のライフタイムおよびタイムアウト、最小および最大のプール・サイズ、一度にプールから増加または減少する接続数を含んだ、チューニング可能な設定で接続プールを作成します。こうしたパラメータによって、アプリケーションが大規模なユーザー数をどのように扱い、時間とともに利用者数をどのように変化させるかを、開発者は強力に制御できるようになります。これは、アプリケーションの応答時間とエンドユーザーのサービス品質の改善につながります。

特定の問合せまたは PL/SQL 文が何度も実行される場合、ODP.NET は、ステートメント・キャッシュを使用して、文の実行を高速化できます。最初の文の実行中に作成されたサーバー・カーソルをキャッシングすることで、ステートメント・キャッシュで後続の実行処理の前に各文を再解析する必要がなくなります。後続の各文では、保存された解析情報が再利用され、文が実行されます。結果セット・データ自体はキャッシュされません。解析された文の情報だけです。ODP.NET は、データベース・サーバーから最新のデータを取得します。ステートメント・キャッシュによって、これらの問合せを迅速に実行できます。

ステートメント・キャッシュを採用する場合、リテラル値ではなくパラメータを SQL または PL/SQL 文で使用してください。これを実行すると、ステートメント・キャッシュをフル活用できます。後続の実行でパラメータ値が変更されてもパラメータ化された文から解析情報を再利用できるためです。リテラル値を使用してそのリテラル値が変更された場合、解析情報を再利用できません。データベースで新しく文を解析する必要があります。

デフォルトで、ODP.NET は、最近実行された 10 の文をキャッシュします。アプリケーション・レベルまたはマシン・レベルで、キャッシュする文の数およびキャッシュする文を構成できます。

データ・フェッチ・サイズの制御

データ取得パフォーマンスをチューニングする場合、ODP.NET では、各データベース・ラウンドトリップに返す一連のデータ量を指定できます。多くの場合、開発者は、問合せを実行したデータを一度に取得する必要がない場合がほとんどです。エンドユーザーは、一部のデータを一定期間使用する場合があります。

2 つの ODP.NET OracleCommand プロパティ（FetchSize と RowSize）を使用して、開発者が定義した個別のチャンクに問合せのデータ・フェッチを配置できます。FetchSize は、データベース・ラウンドトリップごとに取得するデータ量を ODP.NET に通知します。RowSize は、各行のデータ・サイズを示します。これは問合せの実行後に設定される読取り専用のプロパティです。開発者がデータベース・ラウンドトリップごとに 10 行のデータをフェッチする場合、RowSize に 10 を乗算した値を FetchSize に設定するだけです。RowSize の長所は、実行時に値を設定できることです。そのため、あとでスキーマまたは問合せを変更した場合も、ラウンドトリップごとに 10 行のデータをフェッチするためにコードを変更する必要がありません。

LOB データの最適化

類似したフェッチ・サイズ・チューニング機能は LOB データ型にあります。イメージおよびドキュメントを格納するためにこれらのデータ型が使用されます。サイズがギガバイトになる場合もあります。LOB の使用可能なデータ・サイズおよび LOB データの使用方法に起因して、LOB アプリケーションではパフォーマンスが重要な問題になることが多くあります。サーバーおよびクライアント間でギガバイトのデータを送信すると、データの取得が効率的に処理されない場合にネットワークが過負荷になる可能性があります。

開発者は、ODP.NET を使用して、LOB データの取得方法を指定できます。LOB 問合せの実行時に、開発者は、すべての LOB データをすぐにフェッチするか、ユーザーがデータを読み取るまで LOB のフェッチを遅延するかを選択できます。エンドユーザーは、問合せの実行後に直接データを読み取る必要がない可能性があります。開発者が LOB のフェッチの遅延を選択する場合、LOB の読取りが呼び出されるたびに取得するデータ量を指定できます。エンドユーザーが 10KB のデータだけを一度に読み取る必要がある場合、開発者は LOB の読取りごとに 10KB のデータを取得できます。これによって、サーバーとクライアント間のネットワーク・リソースの使用方法が最適化されます。

また、ODP.NET 開発者は、ランダム・アクセスで一部の LOB を取得できます。エンドユーザーは、1GB の LOB から最後の 100MB のデータだけを必要とする場合があります。開発者は、LOB の取得をチューニングして、最初の 900MB のデータをクライアントに返さずに最後の 100MB だけをフェッチできます。これらのチューニング・オプションは、優れた実行アプリケーションを構築する柔軟性を.NET 開発者に提供します。

LOB データ・サイズは一般的に大きいので、デフォルトでは LOB データ・フェッチを問合せの実行後に遅延します。多くの大きい LOB を取得する場合は、この動作が最適です。クライアントに配信される LOB データでネットワークに過負荷がかかることを防止します。ただし、小さい LOB の場合はこの動作が遅くなり、必要以上に多くのデータベース・ラウンドトリップが発生する可能性があります。

すべての小さい LOB データをすぐにフェッチできるように、ODP.NET には、OracleCommand および OracleDataReader クラスに InitialLOBFetchSize プロパティがあります。InitialLOBFetchSize をゼロ以上の値に設定すると、問合せが実行されたすべての LOB から最初の LOB データが、このプロパティで指定される文字またはバイトの数まで、1 回のラウンドトリップでフェッチされます。たとえば、InitialLOBFetchSize が 10KB に設定された場合、選択されたすべての LOB の最初の 10KB が 1 回のデータベース・ラウンドトリップでクライアントに送信されます。これによって、多くの小さい LOB を使用しているアプリケーションを大幅に高速化できます。

配列データ

ODP.NET 特有の機能の 1 つに、データベースと.NET Framework 間で配列を渡す機能があります。配列を使用すると、データベースとクライアント間で同じデータ型の大きいデータセットを簡単に共有できます。ODP.NET は、データベースの PL/SQL 連想配列を使用して、.NET 配列でデータを受け渡します。

64 ビットの.NET Framework

64 ビットの.NET Framework を使用する場合、.NET 開発者は、スケーラブルで高いパフォーマンスのハードウェア・システムにアクセスできます。Windows x64 の AMD64 プロセッサか Intel EM64T プロセッサ、または Windows Itanium の Itanium プロセッサを選択できます。64 ビットのシステムには、32 ビットのシステムよりも大量のメモリを直接処理する機能があります。高パフォーマンスの計算のために最適化されたハードウェア・コンポーネントが含まれます。10.2.0.3 リリースから、ODP.NET は、各プラットフォームのネイティブな 64 ビット・データ・アクセス・ドライバを使用した両方の 64 ビットの.NET Framework をサポートしています。開発者は、64 ビットのアプリケーションとして ODP.NET 中間層を配置し、スケーラブルなハードウェアを活用できます。

パフォーマンス - ODP.NET 11*g*の新機能

ODP.NETパフォーマンスを強化する新しいOracle Database 11*g*機能を使用できます。これらの機能には、クライアント結果キャッシュ、高速なLOBフェッチ、ステートメント・キャッシュによる高速なパフォーマンスが含まれます。

Oracle Database 11*g*では、新しいパフォーマンスの最適化が導入されています。.NETアプリケーション開発者は、既存のクライアント・コードを変更せずにその多くの機能を使用できます。これらの新機能には、クライアント結果キャッシュ、高速なLOBの取得、ステートメント・キャッシュの強化が含まれます。

クライアント結果キャッシュ

Oracle Database 11*g*のサーバーおよびクライアントを使用する場合、ODP.NETアプリケーションは、Oracleクライアント結果キャッシュを使用して、繰り返し実行される問合せの応答時間を向上できます。この機能によって、メモリ内のSQL問合せ結果セットのクライアント側のキャッシュを使用できます。クライアント結果キャッシュは、ODP.NETアプリケーションに対して完全に透過的です。結果が変更される、あらゆるセッションまたはデータベース・サーバー側の変更に対して、結果セット・データのキャッシュの一貫性は自動的に保持されます。

問合せ結果がローカルで取得されるので、同じ問合せを何度も呼び出す.NET アプリケーションのパフォーマンスが向上します。問合せおよびフェッチの結果を再実行するデータベース・ラウンドトリップを実行するよりもローカル・クライアント処理は高速になります。アプリケーションが頻繁に同じ問合せを実行すると、結果がクライアントにキャッシュされる場合に大幅にパフォーマンスが向上し、データベース・サーバーの負荷がが削減されます。

データベース・サーバーのクライアント・キャッシュは、問合せ結果を処理して返すために使用されるサーバーCPU およびネットワーク・トラフィックの負荷を削減します。これによって、サーバーのスケーラビリティが向上します。同一のスキーマ、SQL テキスト、バインド値、およびセッション設定が存在する場合、複数のセッションの ODP.NET 文がクライアント・プロセス・メモリのキャッシュされた結果セットを適合させることができます。存在しない場合は、サーバーで問合せが実行されます。これは、複数の ODP.NET ユーザーが同じ結果キャッシュにアクセスできることを意味します。これによって、キャッシュの冗長性が最小限に抑えられ、メモリが節約されます。

クライアント・キャッシュとデータベース・サーバー・データの整合性が自動的に維持されるので、開発者はキャッシュとサーバーを同期するためにコードを記述する必要がありません。クライアント・キャッシュ・データを無効化するサーバーの変更が発生した場合、Oracle クライアントは、キャッシュを自動的に無効にして、次の問合せの実行時にキャッシュを更新します。

高速な LOB のフェッチ

ODP.NET 11*g*は、LOBのデータ、長さ、およびチャンク・サイズをプリフェッチするために必要なデータベースのラウンドトリップの数を削減して、小さいサイズのLOBを取得するパフォーマンスを向上させます。この拡張機能は、従来のLOBまたはSecureFileと組み合わせてOracle Database 11*g*から使用できます。この拡張機能は、開発者に対して透過的です。既存のODP.NET LOBコードを変更せずに、ほかのLOBデータ型のように使用できます。

強化されたステートメント・キャッシュ

ODP.NET 11*g*は、既存のステートメント・キャッシュのキャッシング・インフラストラクチャを拡張して、ODP.NETパラメータ・コンテキストをキャッシュします。この拡張機能は、現在サポートされているすべてのOracleデータベース・サーバー・バージョンで使用できます。.NET開発者は、ステートメント・キャッシュの問合せの実行時にパフォーマンスが向上していることを確認できます。この拡張機能は開発者に対して透過的であり、コードを変更する必要がありません。

データベース変更通知

クライアント側のキャッシュで常に課題になるのは、サーバーのデータ変更とデータの同期です。データベース変更通知を使用すると、データベースのアクティブな接続がなくても、サーバーのデータが変更された場合に ODP.NET クライアントに通知されます。これによって、クライアントがデータベースとデータ・キャッシュの同期を維持できます。

データベース変更通知を使用すると、クライアントのデータベース・サーバー接続が解除されていても、目的のデータベース・オブジェクトに DML または DDL の変更がおこなわれた場合にクライアント・アプリケーションで通知を受信できます。.NET 開発者は、キャッシュされたデータとデータベースの同期を気にすることなく中間層にデータをキャッシュできます。キャッシュされたデータ・オブジェクトまたはデータ行で変更が発生した場合、ODP.NET はデータベースから通知を受け取ります。.NET Framework 1.x、.NET Framework 2.0、およびそれ以降のリリースでこの機能を使用できます。

データベース変更通知を使用するには、クライアント・アプリケーションで問合せをデータベースに登録します。基本となるデータベース・オブジェクトの依存性が問合せに含まれ、依存オブジェクトの変更がコミットされる場合、データベースで変更通知がクライアント・アプリケーションに発行されます。通知には、変更されたデータまたはオブジェクトのメタデータだけが含まれます。変更されたデータは含まれません。.NET 開発者は、変更されたデータを取得するために登録された問合せを再発行するクライアント・イベント・ハンドラを作成できます。

データベース変更通知は、キャッシュされた結果を使用するアプリケーションにとくに役立ちます。従来、コストのかかるデータベースへのラウンドトリップを実行せずに、データへの迅速なアクセスを実現するデータ・キャッシュがアプリケーションのスケーラビリティの向上に効果的でした。ただし、このスケーラビリティにはトレードオフがあり、最初の問合せのあとにデータとデータベース・サーバーの同期が保証されません。このため、キャッシュされたクライアント・データが古くなるというリスクがあります。

データベース変更通知は、古いデータ・キャッシュの問題を解決します。特定のイベントに応答する点でデータベース変更通知とトリガーは似ています。ただし、トリガーはすぐにアクションを実行しますが、データベース通知は単なるアラートであり、アクションではありません。実行するアクションと実行時間の決定はアプリケーションに依存します。アプリケーションは、古いオブジェクトの即時リフレッシュ、リフレッシュの遅延、または通知の無視を実行できます。各.NET アプリケーションで、特定のデータベース変更に異なる応答をおこなう場合があります。また、追加のアプリケーションがデータベースに追加になります。多くの場合、データベース・トリガーの変更よりもクライアント・アプリケーションのイベント・ハンドラの変更の方が容易にです。トリガーを変更すると、既存のアプリケーションと新しいデータベース・トリガー・コードの互換性の再テストが必要な場合があります。一方で、新しい.NET アプリケーションを変更すると、テストの境界が効果的に分離されます。

多くの場合、Web アプリケーションはさまざまなデータをキャッシュします。すべてをリアルタイムで更新する必要はありません。たとえば、天気予報は定期的に更新されます。エンドユーザーが、Web ページにアクセスするたびにデータベースの問合せを実行する必要はありません。多くのユーザーが同じデータを要求するため、結果をキャッシュしてキャッシュのデータを取得することで、アプリケーションのパフォーマンスおよびスケーラビリティが大幅に向上します。特定の時点で、天気予報を更新し、キャッシュをリフレッシュする必要があります。たとえば、データベース・サーバーの現在の天気予報が変更される際にこれを実行します。

データベース変更通知を受信するには、データベース管理者がアプリケーション・ユーザーに CHANGE NOTIFICATION 権限を付与する必要があります。データベースの接続後、.NET ユーザーは、変更を通知する特定の問合せを登録できます。開発者は、.NET クライアント側のイベント・ハンドラを作成して、データベース変更通知を受信する際のアプリケーションの実行処理を指示します。通常、イベント・ハンドラは、データベース・サーバーの問合せを再実行し、キャッシュをリフレッシュします。

変更通知アプリケーションを構築する場合、次の ODP.NET クラスが使用されます。

- OracleDependency - アプリケーションと Oracle データベース間の依存性を作成します。データの変更（UPDATE 文など）、スキーマの変更（ALTER TABLE など）、またはグローバルなイベント（データベースの停止など）の通知をアプリケーションで受信できます。このクラスの OnChange イベント・ハンドラは、通知の受信後に実行する処理のクライアント・ロジックを提供します。
- OracleNotificationEventArgs - 変更通知が発生した場合にすべてのイベントの詳細を提供します。
- OracleNotificationRequest - 通知登録のタイムアウト値など、通知リクエストの特性とその通知を指定します。

Oracle データベースの変更通知には、SQL Server では使用できない多くの機能があります。Oracle データベースはすべてのタイプの結合をサポートしますが、SQL Server は外部結合または自己結合を含む問合せをサポートしません。SQL Server ではビューを使用した文の通知をサポートしませんが、Oracle データベースの変更通知では固定ビュー（V$表など）とマテリアライズド・ビュー以外のビューをすべてサポートします。SQL Server の通知では、明示的な列の参照も必要になります。Oracle Database の通知では、SELECT *と明示的な列の参照の両方をサポートします。

SQL Server の通知ハンドラは永続的ではありません。SQL Server の通知が発行されると、データベースから通知ハンドラが削除されます。通知ハンドラが必要な場合、アプリケーションは新しい通知ハンドラを登録する必要があります。Oracle データベースの変更通知は、変更が繰り返されても登録を保持するオプションを提供します。OracleNotificationRequest.IsNotifiedOnce を false に設定すると、これが可能です。

Oracle Real Application Clusters（Oracle RAC)

ODP.NET には、Oracle RAC データベースの 2 つの接続プーリング機能があります。1) リアルタイムのクラスタ・メトリックに基づいて、ODP.NET 接続の既存のノードのロードバランシングが自動的に実行されます。また、2) 切断されたデータベース接続が接続プールから自動的に削除されます。

Oracle RAC は、従来のシェアード・ナッシングおよびフェデレーテッド・データベース・アプローチの制約を克服する共有キャッシュ・アーキテクチャを備えたクラスタ・データベースです。単一のコンピュータ・サーバーよりも優れたスケーラビリティおよび可用性を実現する複数のコンピューティング・ノードに Oracle RAC クラスタ・データベースがホストされます。特殊なハードウェアおよび.NET アプリケーションを変更する必要がないので、コードを変更することなく既存のアプリケーションのバックエンドとして、汎用ハードウェアを使用して Oracle RAC を構築できます。

ODP.NET は、Oracle RAC への透過的なデータ・アクセスを常にサポートします。リアルタイムのデータベース・ワークロード情報に基づいて ODP.NET 接続管理を改善するため、オラクルでは ODP.NET 10.2 に 2 つの接続プール・プロパティを導入しました。

１つ目の機能のランタイム接続ロードバランシングでは、とくにクラスタからノードを追加または削除したあと、Oracle RAC インスタンス間でワークロードのロードバランシングを改善します。2 つ目の機能の高速接続フェイルオーバーでは、切断された Oracle RAC 接続を接続プールから自動的に削除します。

ランタイム接続ロードバランシングを使用した ODP.NET 接続の割当て方法は、データベースのロードバランシング・アドバイザリと実行時のサービスの目標に基づいています。ロードバランシングは、使用できるすべての Oracle RAC データベース・インスタンスの処理を分散します。

通常、少ない頻度で接続が作成され、長期間接続が維持されます。処理は、頻繁にシステムに組み込まれ、プールのこうした接続を使用し、比較的短期間存在します。ロードバランシング・アドバイザリは、最適なサービス品質を実現するために ODP.NET が受け取った処理を割り当てる Oracle RAC インスタンスを指示します。ロードバランシングは、あとで異なるインスタンスに処理を再配置する必要性を最小限に抑えて、既存のジョブの迅速な実行を保証します。

インスタンス間でこの処理を分散するメトリックは、サービスの目標によって決定されます。データベース管理者は、サービス時間またはスループットのサービスの目標を設定します。

サービス時間メトリックは、データベースによるタスクの実行速度（応答時間）に基づいています。ロードバランシング・アドバイザリ・データは、サービスで実行された処理の経過時間とサービスの使用可能な帯域幅に基づいています。インターネット・ショッピング・システムなどの異なる速度で処理が実行されるデータベース・アプリケーションには、サービス時間がもっとも役立ちます。スループット・メトリックでもデータベース・タスクの実行速度を追跡し、タスクが最速で完了したノードに追加の処理を割り当てます。サービス時間とは異なり、スループット・メトリックは、各インスタンスの効率性の測定を目的とします。追加のタスクを実行する場合に使用できるプロセッサ・リソースの量を測定します。新しい操作を受け取った場合、もっともプロセッサ時間に余裕のあるノードを指示します。一様なバッチ・プロセスの実行など、比較的均一なタスクにこのメトリックを使用すると最適です。このようなサービスの目標を使用することによっ

て、フィードバックがシステムに組み込まれます。全体に最適なサービス時間を提供するために処理がルーティングされます。ルーティングは、変化するシステム条件に正しく対応します。クラスタの Oracle RAC ノードの追加または削除を実行する場合、ロードバランシングを実行して、システムの変更に関連するすべてのノードに処理を迅速に分散できます。エンドユーザーがサービスの停止や遅延に直面することを減少させます。安定した状態で、システムは、すべての Oracle RAC インスタンス間の向上したスループットで均衡を保ちます。

ODP.NET のこの機能を使用するには、アプリケーションで接続プーリングを有効にし、Load Balancing 接続プール・パラメータを true に設定する必要があります。

Oracle RAC 接続プーリングの 2 つ目の構成可能な機能である高速接続フェイルオーバーによって、停止した Oracle RAC サービス、サービス・メンバー、またはノードによる接続の切断に関連したリソースを ODP.NET が自動的に解放できます。この機能がないと、Oracle RAC ノードで障害が発生した場合は、無効になった接続リソースが接続プールに保持されます。エンドユーザーは、プールのこのような切断された接続を使用する場合があります。これらの接続を識別する方法がない場合、管理者は、Oracle RAC クラスタの一部で障害が発生するたびにプールのすべての ODP.NET 接続を再設定する必要があります。

これらの切断された ODP.NET 接続は、実行時に管理者が介入することなく自動的にクリーンアップされます。この ODP.NET 機能を有効にするには、接続プーリングを使用し、HA Events 接続プール・パラメータを true に設定します。

XML 機能

ODP.NET は、Oracle XML DB のサポートおよび System.XML との相互運用を提供します。ODP.NET は、OracleXMLType および OracleXMLStream を使用したネイティブな Oracle XML データ型をサポートするスキーマとスキーマ・ベースではない XML に対応できます。

XML はデータ統合および Web サービスの一般的な言語なので、多くの.NET プログラマーは、アプリケーションの不可欠な要素として XML を組み込みます。XML は、Oracle データベースおよび.NET Framework の主要な部分です。ODP.NET によって、開発者は 2 つのテクノロジー（Oracle XML DB および.NET System.XML サービス）を同時に利用できます。

Oracle XML DB は、データベース・サーバー内で使用できる Oracle の高パフォーマンスでネイティブな XML の格納および検索テクノロジーです。標準 W3C XML データ・モデルの構造化データと非構造化データを格納および管理する一意な機能を提供します。Oracle XML DB は、XML および SQL メタファの完全な透過性および互換性を提供します。ODP.NET は、Oracle XML DB の機能を.NET クライアントに公開します。これによって、開発者はデータベースと.NET 間の XML の共有および変更を実行できます。このサポートがスキーマおよびスキーマ・ベースではない XML に拡張され、異なるアプリケーション要件を適用する柔軟性が提供されます。また、ODP.NET は、2 つのネイティブな XML データ型（OracleXMLType および OracleXMLStream）を使用して、クライアントの XML データ管理を容易にします。このような機能を使用して、Oracle XML DB および Microsoft の System.XML サービスの XML 管理を簡素化できます。

System.XML は、.NET データ・プロバイダから XML データセットを操作する一連のインタフェースです。ODP.NET は、ODP.NET DataAdapter インタフェースを通じてデータを供給する System.XML プログラミング・インタフェースと相互運用します。Oracle XML DB と System.XML のおもな違いは、Oracle XML DB がデー

タベース・サーバーのデータの存在する場所に XML サービスを提供するのに対して、System.XML はクライアント側の XML を操作することです。このため、ODP.NET を使用すると、プログラマーはプロジェクト要件に最適な XML テクノロジーを幅広く選択できます。

ODP.NET を使用する場合、Oracle データベースのリレーショナル・データおよびオブジェクト・リレーショナル・データに XML としてアクセスし、Microsoft .NET 環境に提供できます。クライアントの XML 変更を実行し、XML またはリレーショナル・データとしてサーバーに保存できます。ODP.NET には、スキーマ・ベースの OracleXMLType とスキーマ・ベースではない XML のサポートが含まれます。

ADO.NET 2.0

ODP.NET は、プロバイダ・ファクトリ、接続文字列ビルダー、スキーマ検出 API、DataAdapter バッチ更新を含む ADO.NET 2.0 の機能をサポートします。

バージョン 10.2.0.2 から、ODP.NET は ADO.NET 2.0 をサポートします。ADO.NET 2.0 では、データ・アクセス・レイヤー（DAL）の新しいレベルの抽象化が導入されます。汎用インタフェースを実装するプロバイダ固有のクラスを使用する代わりに、ADO.NET 2.0 は、System.Data.Common 名前空間から継承される DbCommon クラスを提供し、開発者はファクトリ・クラスを使用できます。データベース・プロバイダ・ファクトリ・クラスを使用すると、データベースへの一連の汎用データ・アクセス・コードを簡単に作成できます。

OracleClientFactory クラスは、System.Data.Common のすべてのクラスの作成をサポートします。これらの具象クラスは DbCommon 抽象基本クラスから継承されるので、開発者は、DbCommon 基本クラスのオブジェクト名を使用して、汎用 DAL コードを記述できます。接続文字列、SQL、ストアド・プロシージャ・コールを含むデータソース固有の DAL の領域があります。

ADO.NET 2.0 のおもな追加機能は、向上した接続文字列管理です。DbConnection StringBuilder クラスには、汎用パラメータ名とプロバイダ固有のパラメータ名をマップするディクショナリがあります。DbConnectionStringBuilder 汎用クラスから継承および拡張された OracleConnectionStringBuilder クラスは、Oracle 固有の接続文字列プロパティを公開します。プログラマーは、OracleConnectionStringBuilder を動的に使用して、実行時に接続文字列パラメータを設定するか、app.config ファイルから接続文字列パラメータを取得するか、あるいはその両方が可能です。この機能によって、とくに多くの属性を使用する接続文字列の管理が容易になります。

ADO.NET 2.0 のスキーマ検出 API は、データソースからメタデータを取得する汎用的な方法を提供します。開発者は、OracleConnection.GetSchema メソッドの呼出しを使用して、Oracle メタデータを取得できます。5 つのタイプ（MetaDataCollections、Restrictions、DataSourceInformation、DataTypes、ReservedWords）の一般的なメタデータが公開されます。また、取得できる追加の ODP.NET 固有のデータソース情報があります。開発者は、スキーマ検出を使用して、Oracle データソースからメタデータを簡単に取得できます。

大きい.NET DataSet を使用する場合、開発者は、更新に必要なデータベース・ラウンドトリップの数を最小限に抑えようとします。プログラマーは、OracleDataAdapter.UpdateBatchSize プロパティを使用して、ラウンドトリップごとに更新する行数を指定できます。1 回のラウンドトリップで変更されたすべての行を更新する場合、UpdateBatchSize をゼロに設定できます。このバッチ処理サポートによっ

て、DataSet 更新をさらに迅速に実行できます。

ネイティブな Oracle 型

ODP.NET は、.NET 環境内で REF Cursor などのネイティブな Oracle データベース型をサポートします。これらのネイティブなデータ型によって、データ取得の優れたパフォーマンスと柔軟性が実現します。

Microsoft は、異なる.NET プログラミング言語に統一された一連のデータ型を導入しました。.NET プログラマーは、ODP.NET を使用して、.NET データ型と Oracle データ型にアクセスできます。.NET アプリケーション内で Oracle データ型を完全に操作して、.NET データ型と相互運用できます。ネイティブな Oracle 型は、XML、結果セット、イメージ、テキスト、Microsoft Office ドキュメントなど、データベースのデータ構造を格納および操作する高度な機能を提供します。.NET の小数型に相当する OracleDecimal などのスカラー型を使用する場合でも、Oracle 型は追加の機能を提供します。OracleDecimal の例において、このデータ型は、28 桁の精度の.NET 小数よりも高水準の 38 桁の精度を提供します。

ODP.NET は、REF Cursor、XMLType、LOB（CLOB、ＢLOB、NCLOB）、BFILE、LONG、RAW、LONG RAW、N データ型を含むあらゆる高度な Oracle 型を.NET 環境内でサポートします。サード・パーティの.NET データ・プロバイダを使用する場合の制限の 1 つは、ユーザーが Oracle データ型機能を制限されることです。たとえば、ODP.NET で、ストアド・プロシージャから REF Cursor 出力パラメータとして返される複数の結果セットに任意の方法でアクセスできます。最初の結果を取得せずに 2 つ目の REF Cursor の結果を読み取ることができます。ほかの.NET プロバイダを使用する場合、次のデータにアクセスする前に最初の結果セットのデータを取得する線形の方法でデータにアクセスする必要があります。これは、パフォーマンスに悪影響を与えます。

ADO.NET 2.0 から、Oracle データ型を.NET DataSet 内にネイティブに格納できます。以前は、DataSet で.NET データ型だけを使用できました。OracleDataAdapter.ReturnProviderSpecificTypes を true に設定することで、OracleDataApdater.Fill が呼び出される場合に DataSet に ODP.NET 固有のデータ型が移入されます。.NET によるデータ型変換が省略されるため、この機能によってパフォーマンスが向上されます。

ほかの主要な機能

ほかの主要な ODP.NET 機能には、すべてのタイプの PL/SQL、ローカルおよび分散トランザクション、Unicode を使用した国際化アプリケーションの使用が含まれます。

ODP.NET は、PL/SQL、トランザクション、Unicode サポートを含む多くのほかの Oracle データベース機能を公開しています。

ODP.NET ユーザーは、データベースの PL/SQL ストアド・プロシージャおよびストアド・ファンクションを完全に実行できます。PL/SQL をパッケージ化したりパッケージ化を解除したりできます。また、.NET 内の匿名の PL/SQL としても使用されます。ODP.NET では、匿名の PL/SQL を採用して、一連の SQL 文のバッチ処理をおこない、1 回のデータベース・ラウンドトリップで文を実行します。文のバッチ処理は、有益なパフォーマンス最適化技術です。

ODP.NET は、リソース・マネージャとして Oracle データベースを使用したトランザクション・アプリケーションに参加できます。ODP.NET は、Microsoft Distributed Transaction Coordinator（DTC）を採用して、Windows 環境のトランザクションを管理します。Oracle Services for Microsoft Transaction Server（OraMTS）は、ODP.NET、DTC、および Oracle データベースのプロキシの役割を果たして、これらのトラン

ザクションを調整します。OraMTS は、トランザクション・アプリケーションの高い可用性とスケーラビリティを維持する ODP.NET プログラマー向けの強力なアーキテクチャを提供します。.NET Framework 2.0 以降で、ODP.NET 10.2.0.3 は、System.Transactions 名前空間を通じてローカルおよび分散トランザクションをサポートします。

ODP.NET は、完全に Unicode をサポートします。.NET ユーザーは、さまざまな言語を使用して簡単にアプリケーションのグローバル対応を実行できます。このグローバリゼーション・サポートによって、開発者はさまざまな文化/言語設定の一連のコードを作成できます。ODP.NET グローバリゼーションでは、クライアント・コンピュータの言語設定を抽出して、ロケール固有の形式で情報を表示します。たとえば、日本語に設定されているブラウザには、通貨に円が表示されます。追加のコーディングをおこなわずに、同じアプリケーションをドイツに配置して通貨にユーロを表示できます。したがって、追加のコーディングをおこなうことなく複数のロケールのアプリケーションを簡単かつ迅速に開発できます。

結論

Oracle Database 11*g*のODP.NETは、既存のパフォーマンス、スケーラビリティ、使いやすさ、およびセキュリティ機能を基盤とするOracleデータベース開発者向けの新しいパフォーマンス機能を提供します。Oracleデータベースと.NET Frameworkの新しい機能が導入されており、ODP.NETはこうした新機能を引き続きサポートしていきます。

ODP.NET および Oracle on .NET の概要について、詳しくは以下の Web サイトを参照してください。

OTN ODP.NET Product Center：
http://www.oracle.com/technology/tech/windows/odpnet/（US OTN、英語）

OTN NET Developer Center：
http://www.oracle.com/technology/global/jp/tech/dotnet/index.html

Oracle Data Provider For .NET 11*g*
2007 年 10 月
著者：Alex Keh

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

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
www.oracle.com