ORACLE

データベース常駐接続プーリング（DRCP）を使用したOracle Database接続の卓越したスケーラビリティ

アプリケーションおよび中間層サービスでのOracle Databaseリソース使用率の最適化

2025年3月、バージョン3.3

免責事項

本文書には、ソフトウェアや印刷物など、いかなる形式のものも含め、オラクルの独占的な所有物である占有情報が含まれます。この機密文書へのアクセスと使用は、締結および遵守に同意したOracle Software License and Service Agreementの諸条件に従うものとします。本文書と本文書に含まれる情報は、オラクルの事前の書面による同意なしに、公開、複製、再作成、またはオラクルの外部に配布することはできません。本書は、ユーザーとのライセンス同意書の一部をなすものではなく、またオラクルやその子会社および関連会社とのいかなる契約上の合意事項にも含まれるものではありません。

本書は情報提供のみを目的としたものであり、ここで説明する製品の機能を実装およびアップグレードする際の資料として使用されることのみを意図しています。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するものでもありません。本書に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

画像一覧

表一覧

はじめに

データベース常駐接続プーリング（DRCP）は、データベース・リソース使用率が最適である複数の接続を必要とする環境向けに開発された、Oracle Databaseの機能です。DRCPは通常、アプリケーションがデータベース接続を取得し、それを使用して比較的短時間作業し、その後接続を解放するマイクロサービスやWebアプリケーションでの使用に適しています。DRCPは、同じまたは複数のアプリケーション層ホスト上で実行されている複数のアプリケーションで共有できる"専用"サーバー・プロセスのプール（**プール・サーバー**と呼ばれます）をデータベースに提供します。これらのプール・サーバーは、クライアント・アプリケーションとのデータベース接続/セッションを処理します。接続ブローカ・プロセスは、データベース・インスタンス・レベルでプール・サーバーを制御します。DRCPはアプリケーションの実行時に選択される構成可能な機能であるため、クライアント・アプリケーションは従来の接続アーキテクチャとDRCPベースの接続アーキテクチャの両方を同時に使用できます。

Oracle Databaseの従来の専用接続モデルでは、接続のオープン時またはクローズ時に、各プロセスがデータベース・サーバーを作成または破棄します。アイドル状態の専用接続を持つアプリケーションは、サーバー・プロセス、メモリ・ストレージなどのデータベース・リソースを保持します。

ゴル航空におけるDRCPの活用

「ゴル航空は、他のベンダーと比較し、セキュリティ、コスト、優れたパフォーマンスに基づいてOracle Cloud Infrastructureを選択しました。アプリケーションで同時接続数が増加した場合、ゴル航空のIT部門は自律型クラウド・データベースのデータベース常駐接続プールを使用します。」

GOL Linhas Aéreas、ブラジル
ブラジルの大手航空会社
オラクルのお客様の成功事例

DRCP実装では、複数のアプリケーション間で共有できるサーバー・プロセスのプールが、データベース・ホスト上に作成されます。DRCPプールは、データベース・サーバー・プロセスの数を減らしてクライアント接続を多重化することで、サーバーのメモリ消費量を大幅に削減します。これにより、データベース・サーバーの作成と破棄によるオーバーヘッドがなくなり、Oracle Databaseが関係するアプリケーション・デプロイメントのスケーラビリティが向上します。本書で後ほど説明するように、DRCPでアイドル接続を持つアプリケーションは、データベース・リソースを消費しません。

DRCPにより、データベースへの接続が最小限のコストで維持されるため、データベースとアプリケーション層のスケーラビリティが向上します。データベース・メモリは、プール・サーバーによってのみ使用されます。

DRCPは、Oracle Database 11g以降のすべてのエディションで利用でき、オンプレミスとOracle Cloudで使用できます。DRCPは、JDBC、ODP.NET、Oracle Call Interface（OCI[1]）ライブラリを実行する任意のアプリケーションで使用して、Oracle Databaseに接続することができます。Python、Node.js、PHP、Ruby、Go用のOracle Databaseドライバを使用するアプリケーションは、DRCPもサポートします。

本書では、DRCPの使用を開始して実行するためのアーキテクチャ、構成設定、コマンド、メリット、システム・ビュー、例について説明します。

DRCPの概要について詳しくは、最新のOracle Database管理者ガイド：DRCPのセクションを参照してください。

DRCPアーキテクチャの概要

DRCPを使用すると、アプリケーションを最大で数万の同時データベース接続までスケールアップすることができます。DRCPアーキテクチャは、このスケーラビリティの実現において重要な役割を果たします。

DRCPはプール・サーバー・プロセスを使用します。これは基本的には専用サーバー・プロセスとデータベース・セッションの組合せです。このモデルにより、短時間しかサーバーを必要としないすべてのクライアント接続に専用サーバーを割り当てるというオーバーヘッドが回避されます。

DRCPに接続をリクエストするクライアント・アプリケーション（このセクションでは以降、クライアントと呼びます）は、**接続ブローカ**と呼ばれるOracleバックグラウンド・プロセスと通信します。接続ブローカは、クライアントからのインバウンド接続リクエストのプール・サーバー・プロセスを多重化します。

[1] Oracle Databaseに接続するためのオラクルのネイティブCベース・ライブラリ

最初に、接続ブローカは、**認証サーバー**と呼ばれる、DRCPプロセスの予約セットを使用して、クライアントからの接続リクエストを認証します。通常、現在のプール・サーバーの約5 %が認証用に予約されています。

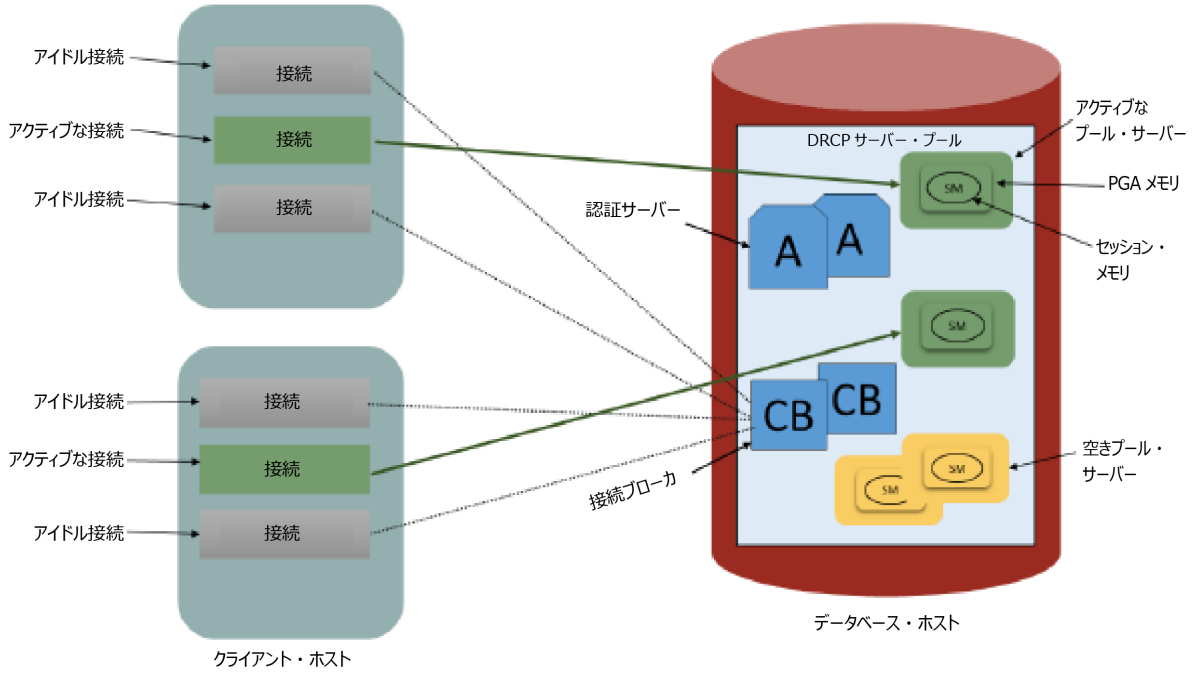

図1：クライアントとデータベース・ホスト間のDRCPアーキテクチャ

クライアント・プロセスがアプリケーション接続を取得するたびに、接続ブローカが空きプールからプール・サーバー・プロセスを選択し、それをクライアントに渡します。クライアントは、データベース・アクティビティが完了するまで、サーバー・プロセス（**'アクティブなプール・サーバー**と呼ばれます）に直接接続されます。

サーバーがデータベース・アクティビティを完了した後、クライアント・アプリケーションはアクティブなプール・サーバー・プロセスを解放して、DRCPプールに戻す必要があります。これにより、接続ブローカへのリンクが再確立されます。接続ブローカへのリンクは、クライアント・プロセスが実行を停止するか、クライアントがアプリケーション接続を取得するまで、開いたままになります。

また、DRCPの場合、セッション・メモリはデータベースのプログラム・グローバル領域（PGA）に保存されます。PGAはプロセス（ここではデータベース接続）にとって物理的にプライベートです。この実装により、クライアントは、データベース・アクティビティが必要な場合に接続を迅速に再確立できます。

DRCPのクイックスタート

プールは、PL/SQL DRCPパッケージ*DBMS_CONNECTION_POOL*内の適切なプロシージャを実行することで開始、構成、停止できます。SYSDBA権限を持つユーザー（すなわち*SYS*ユーザー）、またはSYSユーザーがPL/SQL DRCPパッケージに対するEXECUTEアクセス権を付与したユーザーのみが実行できます。

Oracle Cloud Autonomous Databaseでは、DRCPはデフォルトで開始されます。それ以外の場合は、必要な権限を持つユーザーとしてOracle Databaseにログインし、*dbms_connection_pool.start_pool()* PL/SQLプロシージャを実行します。

プロシージャが正常に実行されると、クライアント・アプリケーションは、*':pooled'*文字列を含むEasy Connect文字列構文、またはネットワーク接続記述子文字列での設定(`SERVER=POOLED`)を使用し、DRCPを介してOracle Databaseにアクセスできます。接続文字列を変更しないアプリケーションは、従来の専用サーバー・プロセスを引き続き使用します。

DRCPの構成およびDRCPを使用するその他のアプリケーションの詳細と例については、この技術概要の後の方のセクションで説明します。

DRCPの仕組み

DRCPでは、データベース・リスナーは最初に、クライアントからの新しい接続リクエストをDRCP接続ブローカ（CB）に渡します。これらの接続リクエストは、接続上でデータベース・トランザクションが実行される前に、最初に認証される必要があります。

ブローカは、プールの予約済み認証サーバーのいずれかを使用して認証を実行します。ログオン・トリガーは、認証ごとに1回、およびユーザー・セッションごとに（ユーザー・セッションの作成時に）1回起動されます。ログオフ・トリガーは、ログオフのたび、およびセッションが解放されるたびに起動されます。認証されると、クライアントが接続を閉じるまで、ブローカはクライアント接続を永続的に維持します。さらに、プール・サーバーのリクエストと解放は、この永続的な認証済みの接続上で行うことができます。これらのアクティビティはブローカによって調整されます。

アプリケーション接続リクエスト時に、接続ブローカはデータベース・サーバー・プロセスの空きプールからサーバーをクライアントに割り当て、認証済みのクライアント接続をこのサーバーに渡します。クライアントは、'ビジー'・サーバーと呼ばれるこの割り当てられたサーバー・プロセス上で、すべてのデータベース対話を保持します。クライアントがAPI呼出しを通じてセッションを明示的に閉じた後、またはクライアント・アプリケーションが終了すると、ビジー・サーバーはセッションとともに解放されて空きサーバー・プールに戻されます。その後、クライアント・アプリケーションがまだ実行中の場合、クライアントは接続ブローカへの接続をリストアします。ブローカは、後続のクライアント接続リクエストが受信されるか、クライアント・アプリケーションが接続を終了するか、またはクライアント・アプリケーションが終了するまで、この接続を保持します。

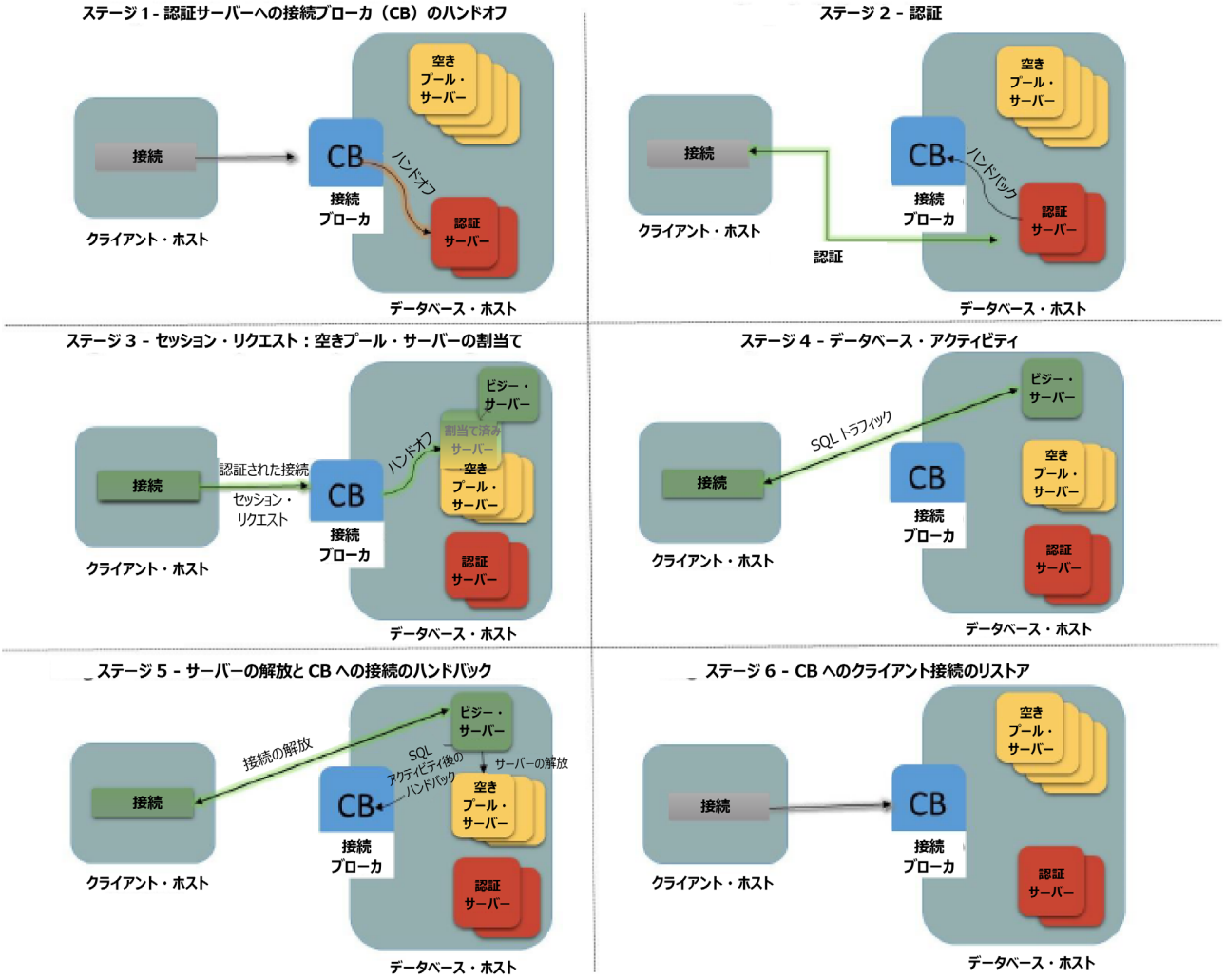

図2：DRCP操作の各ステージ

プールのサイズ、接続ブローカの数、認証サーバーの数は構成可能です。DRCPが有効な場合、データベース・インスタンスごとに少なくとも1つの接続ブローカが常に存在します。

DRCPを使用するケース

DRCPは一般に、次での使用が推奨されます。

- アプリケーション接続プールを使用できないアプリケーション・サーバー（PHPなど）があるアーキテクチャ
- データベース・アクセスとアプリケーション接続プールを必要とする複数のWebサーバー、マイクロサービス、またはアプリケーション・サーバーがある、大規模なWebデプロイメント
- データベース・ホストでのメモリ使用量を最小限に抑えながら、大量のクライアント接続通信量をサポートする必要があるWebアーキテクチャ
- 接続が短時間保持されるアプリケーション
- すべての接続に対して同じデータベース資格証明を主に使用するアプリケーション
- 日付書式やPL/SQLパッケージの状態など、同一の接続設定またはセッション設定を持つアプリケーション

これらのユースケースでは通常、データベースへの多数の接続を永続的に維持しながら、複数のホスト上でマルチプロセス・アプリケーションを実行しますが、接続がアクティブでないときにデータベース・サーバーのメモリを消費することは望ましくありません。データベースは、DRCPを使用することで、数万の同時接続までのスケーラビリティを持つようになります。

たとえば、プールのサイズが200の中間層接続プールに、データベースへの200の接続があるとします。データベースの側から見ると、これらの接続に関連付けられている200のサーバー・プロセスがあることになります。そこに、類似の中間層アプリケーションが30あるとします。データベースには、専用サーバー・モードで実行される、200 * 30 = 6,000の対応するサーバー・プロセスが存在する計算になります。これらの接続、その結果としてサーバー・プロセスが使用中であるのは常に5 %のみであると想定します。この場合、アクティブなサーバー・プロセスは300のみであり、データベース側では常に5,700のアイドル状態のサーバー・プロセスが無駄で未使用のリソースとして実行されています。

DRCPは、プール・サーバーの数を減らしてクライアント接続を多重化することによって、このリソースの浪費の問題を解決できます。たとえば、6,000のクライアント接続に必要なプール・サーバー・プロセスが100のみ（接続の保持時間によって異なります）である場合があり、データベースの最適なリソース使用率の実現とスケーラビリティの向上につながります。

DRCPは、ユーザーID/パスワードベースのデータベース認証を使用してTCP/IP経由で接続する場合に利用できます。オラクルのBequeath接続またはTCPS接続を使用しては利用できません。

最大限の効率とデータベースの最適なリソース消費のために、DRCPをアプリケーション接続プーリングと併用することをお勧めします。最良のDRCPパフォーマンスを実現するには、本書で後述するように、アプリケーションで接続クラスを明示的に指定する必要があります。

さらに、DRCPを使用する場合は、このリンク先に示した事項を考慮に入れてください。

DRCPと他のデータベース・サーバーのプロセス・モデルの比較

DRCPのプール・サーバーに加えて、Oracleアプリケーションは、専用サーバーと共有サーバーという、さらに2つのデータベース・サーバーのプロセス・モデルを使用してデータにアクセスできます。Oracle Databaseは、デフォルトで専用サーバーを提供します。

次の表は、専用サーバー、共有サーバー、プール・サーバーの違いを示しています。

専用サーバー	共有サーバー	プール・サーバー
接続が確立されると、専用サーバー・プロセスへのネットワーク接続と関連セッションが作成されます。	ディスパッチャ・プロセスへのネットワーク接続は、接続の作成時に確立されます。セッションはSGA[2]で作成されます。	ブローカへのネットワーク接続は、接続の作成時に確立され、認証されます。

[2] システム・グローバル領域 - Oracle Databaseインスタンス全体の共有メモリ構造のグループ

専用サーバーは、接続上のアクティビティを処理します	接続上の各アクション[3]はディスパッチャを経由し、ディスパッチャにより処理が共有サーバーに渡されます。	アプリケーションがセッションをリクエストすると、ブローカが起動し、セッションとともにネットワーク接続をプール・サーバーに渡します。 プール・サーバーは、専用サーバーと同様に、後続のデータベース・リクエストまたはアクティビティを直接処理します。
プログラムが実行されているものの、アイドル状態のクライアント接続がある場合、サーバー・プロセスへのリンクは維持され、セッション・リソースは保持されます。	プログラムが実行されているものの、アイドル状態のクライアント接続がある場合、セッション・リソースは保持されますが、サーバー・プロセスへのリンクは維持されません。	プログラムが実行されているものの、データベース・セッションをすでに解放したアイドル状態のクライアント接続がある場合、接続ブローカへのリンクは維持されます。
クライアント接続を閉じると、セッションが解放され、サーバー・プロセスが終了します。	クライアント接続を閉じると、セッションが解放され、クライアントはディスパッチャから切断されます。	クライアント接続を閉じると、プール・サーバーがセッションとともにプールに解放されます。接続ブローカへのネットワーク接続は保持されます。
メモリ使用量は、サーバー・プロセス数とセッション数に比例します。接続ごとに1つのサーバーと1つのセッションがあります。	メモリ使用量は、共有サーバー数とセッション数の合計に比例します。クライアント接続ごとに1つのセッションがあります。	メモリ使用量は、プール・サーバーのプロセス数とそのセッション数に比例します。プール・サーバーごとに1つのセッションがあります。

表1 - 専用サーバー、共有サーバー、プール・サーバーの違い

注：*DRCP*の場合、接続はアイドル状態になると解放されてアプリケーション接続プールに戻されます。

専用サーバー、共有サーバー、プール・サーバーのホスト・メモリ使用量の例

各セッションに必要なメモリが400 KBであるアプリケーションがあると考えてみましょう。64ビット・オペレーティング・システムでは、各サーバー・プロセスに必要なメモリは8 MBであり、DRCPが接続ごとに（主に接続ブローカで）使用するメモリは35 KBである可能性があります。プール・サーバーの数が100で構成されているとします。また、共有サーバーの数が100に設定され、デプロイされたアプリケーションが5,000のアプリケーション接続を作成すると想定します。

各サーバー・タイプで使用されるメモリの見積り値を次の表に示します。

	専用サーバー	共有サーバー	プール・サーバー
データベース・サーバー・メモリ	5000×8 MB	100×8 MB	100×8 MB
セッション・メモリ	5000×400 KB	5000×400 KB 注：共有サーバーの場合、セッション・メモリはSGAから割り当てられます	100×400 KB

[3] アクションは基本的にはSQLNetラウンドトリップであり、1つ以上のSQL文またはPL/SQL文の実行、トランザクションのコミットなどが関係している場合があります。

DRCP接続 ブローカ・オーバーヘッド	0（該当なし）	0（該当なし）	5000×35 KB
合計メモリ	**42 GB**	**2.8 GB**	**1 GB**

表2 - 専用サーバー、共有サーバー、プール・サーバーのサンプルDBメモリ使用量

ご覧のとおり、DRCPプール・サーバーは、データベース・ホストのメモリ使用量の点において、3つのオプションの中でもっとも優れています。

データベース常駐接続プーリングの構成

このセクションでは、サーバー側とクライアント側の両方でDRCPを構成して有効にする方法を説明します。

- サーバー側でのDRCPの有効化と構成
- DRCPを使用するためのアプリケーション・デプロイメント

注：*DRCP*は、*Oracle Cloud Autonomous Database*ではデフォルトですでに開始されています。

DRCPの設定は、DRCP構成オプションとDRCPデータベース初期化パラメータを使用して構成できます。これについてはこのセクションの後の方で詳しく説明します。

サーバー側でのDRCPの有効化と構成

Oracle Database 21c以降、データベース管理者（DBA）がDRCP構成で最初に行う必要がある選択は、PDBごとのDRCPまたはCDB DRCPのどちらにするかということです。CDB DRCPがデフォルトのDRCP構成であることに注意してください。

SYSDBA権限を持つDBA、または`DBMS_CONNECTION_POOL`パッケージに対するEXECUTE権限（`SYS`ユーザーによって付与されます）を持つPDB管理者のみが、プールを開始および停止できます。このセクションでは、SQL*Plusを使用してデータベースにDRCPを構成します。

CDB DRCPの場合、SYSDBA権限を持つデータベース・ユーザー（通常は`SYS`ユーザー）は、`DBMS_CONNECTION_POOL`パッケージに含まれる次のコマンドを使用してDRCPを管理できます。

***1.* プールを開始する：** `start_pool`プロシージャはDRCPを開始します。DRCPの開始時にプール名が指定されていない場合、Oracle Databaseは作成されるデフォルトの接続プールにSYS_DEFAULT_CONNECTION_POOLという名前を付けます。

```
sqlplus /nolog
SQL> connect / as sysdba
SQL> execute dbms_connection_pool.start_pool()
```

Oracle Database 23ai以降で複数プールDRCPを使用している場合、`start_pool`プロシージャでプール名を指定することもできます。

```
sqlplus /nolog
SQL> connect / as sysdba
SQL> execute dbms_connection_pool.start_pool('my_pool')
```

プールは一度開始されると、`stop_pool()`プロシージャで明示的に停止しない限り、インスタンスの再起動時に自動的に再起動されます。

2. プールを停止する： `stop_pool`プロシージャはDRCPを停止します。

デフォルトのDRCPプール（`SYS_DEFAULT_CONNECTION_POOL`）が実行中の場合、次のコマンドによって停止されます。

```
SQL> execute dbms_connection_pool.stop_pool()
```

Oracle Database 23ai以降では、次のようにして、複数プールDRCP内の特定の名前付きプールを停止できます。

```
SQL> execute dbms_connection_pool.stop_pool('my_pool')
```

Oracle Database 23aiでは、`stop_pool()`に新しいオプションのDRAINTIMEパラメータも提供されます。このパラメータを使用すると、指定した接続ドレイン時間（秒単位）の経過後にアクティブなDRCPプールを閉じるか、接続がアイドル状態になるのを待たずにすぐに閉じる（値0）ことができます。この機能により、DBAはDRCPの使用と構成をより適切に制御できるようになります。このパラメータは、デフォルトのDRCPプールおよび複数プールDRCP構成の名前付きプールで使用できます。

次に、いくつかの例を示します。

```
SQL> execute dbms_connection_pool.stop_pool(pool_name => '', draintime => 0)
```

この呼出しは、デフォルトのプールに含まれるすべてのプール・サーバーをすぐに強制終了し、デフォルトのプールを停止します。

```
SQL> execute dbms_connection_pool.stop_pool(pool_name => '', draintime => 30)
```

この呼出しは、30秒待機してからプール・サーバーを強制終了し、デフォルトのプールを停止します。

```
SQL> execute dbms_connection_pool.stop_pool(pool_name => 'my_pool', draintime => 0)
```

この呼出しは、'*my_pool*'という名前のプールに含まれるすべてのプール・サーバーをすぐに強制終了し、プールを停止します。

```
SQL> execute dbms_connection_pool.stop_pool(pool_name => 'my_pool', draintime =>
30)
```

この呼出しは、30秒待機してから'*my_pool*'という名前のプールに含まれるすべてのプール・サーバーを強制終了し、プールを停止します。

3. プールを構成する： `configure_pool`プロシージャは、追加のオプションを使用して、デフォルトまたは名前付きDRCPプールを構成します。たとえば、次のとおりです。

デフォルトのプール(`SYS_DEFAULT_CONNECTION_POOL`)を構成するには、次を実行します。

```
SQL> execute dbms_connection_pool.configure_pool( minsize => 4,
        maxsize => 40,
        incrsize => 2,
        session_cached_cursors => 20,
        inactivity_timeout => 300,
        max_think_time => 600,
        max_use_session => 500000,
        max_lifetime_session => 86400)
```

このプロシージャは、すべての接続プール・パラメータを変更する必要がある場合に使用されます。

*my_pool*という名前のプールを構成するには、次を実行します。

```
SQL> execute dbms_connection_pool.configure_pool(
        pool_name => 'my_pool',
        minsize => 4,
        maxsize => 40,
        incrsize => 2,
        session_cached_cursors => 20,
        inactivity_timeout => 300,
        max_think_time => 600,
        max_use_session => 500000,
        max_lifetime_session => 86400)
```

このプロシージャは、すべての接続プール・パラメータを変更する必要がある場合に使用されます。

4. パラメータを変更する： 別の方法として、メソッド`dbms_connection_pool.alter_param()`を使用し、DRCPプール内の単一のパラメータを設定することもできます。この場合、他のプール・パラメータには影響を与えません。

デフォルトのプール（`SYS_DEFAULT_CONNECTION_POOL`）の'MAX_THINK_TIME'パラメータ値を変更するには、次を実行します。

```
SQL> execute dbms_connection_pool.alter_param(
        param_name => 'MAX_THINK_TIME',
        param_value => '1200')
```

名前付きプール（例：*my_pool*）の'MAX_THINK_TIME'パラメータ値を変更するには、次を実行します。

```
SQL> execute dbms_connection_pool.alter_param(
        pool_name => 'my_pool',
        param_name => 'MAX_THINK_TIME',
        param_value => '1200')
```

オプション`alter_param`とオプション`configure_pool`の違いは、`alter_param`が単一のパラメータにのみ影響を与えるのに対し、`configure_pool`は呼出し時にすべてのパラメータ値を指定する必要があることです。

5. デフォルトをリストアする： `restore_defaults()`プロシージャは、DRCPプールのデフォルトの構成値をリセットします。

デフォルトのプール（`SYS_DEFAULT_CONNECTION_POOL`）のデフォルト値をリストアするには、次を実行します。

```
SQL> exec dbms_connection_pool.restore_defaults()
```

名前付きプール（例：*my_pool*）のデフォルト構成をリストアするには、次を実行します。

```
SQL> exec dbms_connection_pool.restore_defaults(`my_pool`)
```

DRCPがPDBレベル（PDBごとのDRCP）である場合、PDB管理者（このリンク先に示した権限が有効）は、対応するPDBに対して上記のコマンドを実行する必要があります。

DRCP構成設定

次の表は、*configure_pool*プロシージャおよび*alter_param*プロシージャで使用できる*DRCP*構成オプションのリストを示しています。

DRCPのオプション	説明
POOL_NAME	構成するプールの名前。Oracle Database 21cまで、サポートされる名前は、デフォルト値のSYS_DEFAULT_CONNECTION_POOLのみでした。Oracle Database 23ai以降は、新しい複数プール機能で他の名前を使用できます。
MINSIZE	プール内のプール・サーバーの最小数を設定します。DRCPがCDBレベルで構成されている場合のデフォルト値は4で、PDBごとのDRCPが有効な場合は0です。
MAXSIZE	プール内で許可されるプール・サーバーの最大数を設定します。この制限に達し、すべてのプール・サーバーがビジー状態の場合、接続リクエストはサーバーが空くまで待機します。デフォルトは40です。
INCRSIZE	サーバーが接続に使用できず、プールがまだ最大サイズに達していない場合の、プール・サーバーの増分数を設定します。デフォルトは2です。
SESSION_CACHED_CURSORS	すべてのプール接続に対して、データベース・パラメータSESSION_CACHED_CURSORSをオンにします。通常、この数値は、頻繁に使用される文のワーキング・セットのサイズに設定されます。キャッシュはサーバー上のカーソル・リソースを使用します。デフォルトは20です。*init.ora*パラメータは、データベース・インスタンス全体の値を設定するために使用することもできます。プール・オプションを使用すると、DRCPベースのアプリケーションでインスタンス設定をオーバーライドできます。
INACTIVITY_TIMEOUT	プール内の空きサーバーの存続時間（秒単位）。この時間が経過すると、空きサーバー・プロセスは終了します。このパラメータは、プールが最大容量まで使用されていない場合に、プールを縮小するのに役立ちます。プールのサイズがすでに*minsize*に設定されている場合、このパラメータは適用されません。デフォルトは300秒です。
MAX_THINK_TIME	クライアントがプール・サーバーに接続した後に非アクティブであることが許容される最大時間（秒単位）。 アプリケーション・コードまたはスクリプトがこの時間内にデータベース呼出しを発行しない場合、プール・サーバーは再利用のためにプールに戻される可能性があり、クライアント接続は終了します。アプリケーションが後で接続を使用しようとすると、ORA-3113エラーまたはORA-3135エラーが発生します。デフォルトは120秒です。
MAX_TXN_THINK_TIME	クライアントがプール・サーバーを使用してトランザクションを開始した後に非アクティブであることが許容される最大時間（秒単位）。クライアント・アプリケーションがプールからプール・サーバーを取得した後、*max_txn_think_time*で指定された時間枠内にデータベース呼出しを行わない場合、プール・サーバーは解放され、クライアント接続は終了します。

	このパラメータのデフォルト値は、*max_think_time*パラメータ値です。アプリケーションは、このパラメータ値に*max_think_time*値よりも大きい値を設定することで、オープンなトランザクションとの接続により多くの時間を割り当てることができます。アプリケーションが後で接続を使用しようとすると、ORA-3113エラーまたはORA-3135エラーが発生します。
MAX_USE_SESSION	サーバーに再起動のフラグが立てられるまでに、サーバーの取得とプールへの解放を繰り返すことができる最大回数。デフォルトは500000です。
MAX_LIFETIME_SESSION	プール・サーバーが再起動されるまでの存続時間（秒単位）。デフォルトは86400秒です。
NUM_CBROK	接続リクエストを処理するために作成された接続ブローカの数。このパラメータは*alter_param()*で設定できます。デフォルトは1です。 PDBごとのDRCPが有効である場合、このパラメータの設定に*alter_param()*は使用できません。ルートDBAのみがデータベース初期化パラメータCONNECTION_BROKERSを使用してそれらを設定できます。これについてはこちらで説明しています。PDB管理者は、このパラメータの値を変更することもできません。 CDBルートレベルのDRCPでは、このパラメータがCONNECTION_BROKERSを使用して設定されていない場合、ルートDBAは*alter_param()*プロシージャを使用してこのパラメータを設定できます。 このパラメータを設定するには、CONNECTION_BROKERSを使用することを推奨します。
MAXCONN_CBROK	各接続ブローカが処理できる接続の最大数を設定します。オペレーティング・システムのプロセスごとのファイル記述子の制限は、指定された接続数をサポートできるように、十分に大きな値に設定する必要があります。このパラメータは*alter_param()*でのみ設定できます。デフォルトは40000です。 PDBごとのDRCPが有効である場合、このパラメータの設定に*alter_param()*は使用できません。ルートDBAのみがデータベース初期化パラメータCONNECTION_BROKERSを使用してそれらを設定できます。これについてはこちらで説明しています。PDB管理者は、このパラメータの値を変更することもできません。 CDB DRCPでは、このパラメータがCONNECTION_BROKERSを使用して設定されていない場合、ルートDBAは*alter_param()*プロシージャを使用してこのパラメータを設定できます。 このパラメータを設定するには、CONNECTION_BROKERSを使用することを推奨します。

表3 - DRCP構成オプション

DRCPで追加の構成と最適化を行うために、データベース初期化パラメータを設定することもできます。

DRCPパラメータ	説明
ENABLE_PER_PDB_DRCP	Oracle Database 21c以降で利用可能です。このパラメータは、DRCPがCDBレベルでの構成であるか、PDBごとの構成であるかを指定します。デフォルト値はFALSEで、CDBレベルでDRCPを構成します。このパラメータがTRUEに設定されている場合、PDBごとに1つの分離された接続プールが作成され、CDBレベルでは接続プールは作成されません。

DRCP_DEDICATED_OPT	Oracle Database 19.11以降で利用可能です。このパラメータは、DRCPによる専用の最適化の使用を構成します。デフォルトは、Oracle Database 19cではYES、Oracle Database 21c以降ではNOです。このパラメータをYESに設定すると、専用の最適化が有効になります。専用の最適化により、DRCPブローカへの接続数がDRCPプールの最大サイズよりも少ない場合、DRCPは専用サーバーのように動作します。専用の最適化により、接続が非アクティブな場合でも、オープン・プール・サーバーの数を最大サイズまで増やすことができます。 ENABLE_PER_PDB_DRCPパラメータの値に応じて、CDBルート・ユーザーまたはPDB管理者ユーザーがこのパラメータを変更できます。
DRCP_CONNECTION_LIMIT	Oracle Database 21c以降で利用可能です。このパラメータは、PDBでのDRCP接続数の制限を設定します。PDBにセッション制限が設定されており、その後再起動される場合、デフォルトのセッション数は10*です。それ以外の場合は、0（DRCP接続は無制限）です。
MAX_AUTH_SERVERS	Oracle Database 19.10以降で利用可能です。このパラメータは、DRCP認証プール内のサーバー・プロセスの最大数を指定します。この値は、MIN_AUTH_SERVERSパラメータ値以上である必要があります。MIN_AUTH_SERVERS値が0の場合、この値は少なくとも1である必要があります。デフォルト値は25です。 ENABLE_PER_PDB_DRCPパラメータの値に応じて、CDBルート・ユーザーまたはPDB管理者ユーザーがこのパラメータを変更できます。
MIN_AUTH_SERVERS	Oracle Database 19.10以降で利用可能です。このパラメータは、DRCP認証プール内のサーバー・プロセスの最小数を指定します。この値は、MAX_AUTH_SERVERSとPROCESSESの両方のパラメータ値以下である必要があります。ENABLE_PER_PDB_DRCPパラメータの値に応じて、CDBルート・ユーザーまたはPDB管理者ユーザーがこのパラメータを変更できます。
CONNECTION_BROKERS	このパラメータは、接続ブローカのタイプ、各タイプの接続ブローカの数、およびブローカあたりの最大接続数を指定します。PDBごとのDRCPが有効である場合、PDB管理者ユーザーはPDBでこのパラメータを設定できません。

表4 - DRCPデータベース初期化パラメータ

ALTER SYSTEM SQLコマンドを使用すると、ENABLE_PER_PDB_DRCPを除く上記のすべてのパラメータを変更できます。除外されたパラメータは、データベース構成ファイルを介してのみ設定できます。

PDBごとのDRCPのブローカの構成

ブローカ・プロセスはすべてのPDB間で共有されるため、DBA_CPOOL_INFOのプール・パラメータ*num_cbrok*および*maxconn_cbrok*は無視され、PDB管理者がdbms_connection_pool.alter_param()を使用して変更することはできません。これらのパラメータは、データベース初期化パラメータCONNECTION_BROKERSを使用して設定でき、ROOTコンテナ内でのみ動的に変更できます。デフォルトでは、単一のブローカ・プロセスがブローカあたり最大40,000接続の制限で開始され、すべてのPDBで共有されます。ルートDBAは、ALTER SYSTEM SQLコマンドを使用し、次のようにしてCONNECTION_BROKERSパラメータを設定できます。

```
ALTER SYSTEM SET CONNECTION_BROKERS =
'((TYPE=POOLED)(BROKERS=2)(CONNECTIONS=40000))'
```

BROKERSオプションは接続ブローカの数を設定し、CONNECTIONSオプションはブローカごとの最大接続数を設定します。

Oracle Real Application Clusters（Oracle RAC）でのDRCPの使用

DRCPがOracle RAC[4]で使用される場合、各データベース・インスタンスは独自の接続ブローカとサーバーのプールを持つことになります。Oracle RAC環境でのDRCP構成は、すべてのデータベース・インスタンスに適用されます。したがって、各プールは同一の構成を持ちます。たとえば、すべてのプールは*minsize*サーバー・プロセスで開始されます。単一の*dbms_connection_pool*コマンドで、各インスタンスのプールが同時に変更されます。ただし、表6のデータベース初期化パラメータ（ENABLE_PER_PDB_DRCPパラメータとCONNECTION_BROKERパラメータを除く）は、インスタンスが異なる場合には異なる値に設定できます。

Oracle Cloud Autonomous Database（Oracle ADB）でのDRCPの使用

Oracle Cloud Autonomous Database（Oracle ADB）では、DRCPがデフォルトで有効になっています。次のサブセクションに示すとおり、クライアント・アプリケーションはDRCPの使用を選択できることに注意してください。Oracle ADBでは、ユーザーがDRCPを開始または停止することはできません。

クライアント・アプリケーションでのDRCPの操作

データベースでDRCPが有効になっていれば、アプリケーションは、データベースへの接続時にEasy Connect文字列に（以下の例のように）*':POOLED'*を指定するか、ネットワーク接続記述子文字列に(SERVER=POOLED)を指定することで、DRCPを使用できます。

Easy Connect文字列に':pooled'を指定してDRCPを有効にする

```
oraclehost.company.com:1521/booksdb.company.com:pooled
```

ネットワーク接続記述子文字列にSERVER=POOLEDを指定してDRCPを有効にする

```
BOOKSDB = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=oraclehost.company.com)
(PORT=1521))(CONNECT_DATA = (SERVICE_NAME=booksdb.company.com)(SERVER=POOLED)))
```

DRCPを介してOracle Databaseに接続するアプリケーションは、短いデータベース・アクティビティに接続を使用し、データベース・アクティビティが完了したらすぐに接続を閉じる必要があります。

[4] Real Application Clusters - 単一のデータベースが複数のノード上における複数のインスタンスによってホストされるデータベース・オプション

DRCP接続の管理

DRCPにより、あるデータベース・ユーザーが使用を開始したプール・サーバー内のセッションは、その同じユーザーIDによる接続でしか再利用できないことが保証されます。DRCPはまた、プールを論理グループまたは"接続クラス"へとさらにパーティション化します。プール・サーバーも、サービス名に基づいてパーティション化されます。DRCPを介してデータベースに接続するときに最高のパフォーマンスを実現するために、アプリケーションが接続クラスを提供することをお勧めします。

DRCPでは、アプリケーションがセッション純正値属性を設定して、プールされたセッションの再利用性を制御することもできます。

接続クラスとセッション純正値の設定は、複数のアプリケーション、Webアプリ、マイクロサービスがDRCPの可能性を最大限に生かし、エンドユーザーに最高のパフォーマンスを提供するのに役立ちます。

接続クラスとは

接続クラスは、アプリケーションが使用し、複数のアプリケーション・プロセスまたは他のアプリケーションとの間で共有する接続のタイプの論理名を定義します。適切な接続クラスのセットにより、接続が効果的にパーティション化され、接続間での不必要なセッション共有が防止されます。

デフォルトでは、DRCP接続は、複数のデータベース・ユーザーやサービス名の間で共有されることはありません。接続クラスにより、アプリケーションが同じユーザー名とサービス名を使用するときに維持できる共有境界のレベルがさらに追加されます。セッション内で異なる状態を必要とするアプリケーションは、異なるユーザー名や接続クラスを使用する必要があります。

指定された接続クラスのユーザーIDのリクエストに一致する空きプール・サーバーがなく、プールがすでに最大サイズに達している場合は、プール内にある別のクラスのアイドル状態のサーバーが使用され、そのサーバーに対して新しいセッションが作成されます。使用可能なプール・サーバーがない場合、接続リクエストは、プール・サーバーが使用可能になるまで待機します。この動作により、データベースは過負荷にならずに実行し続けることができます。

たとえば、同じユーザー名"Blake"で、*Sales*というグループ内の複数のアプリケーションはプール・サーバーを共有しようとしていているものの、*CRM*というアプリケーション・グループとは共有しないという場合があります。次の図では、グループ名"Sales"および"CRM"も、接続クラスに設定されている値です。

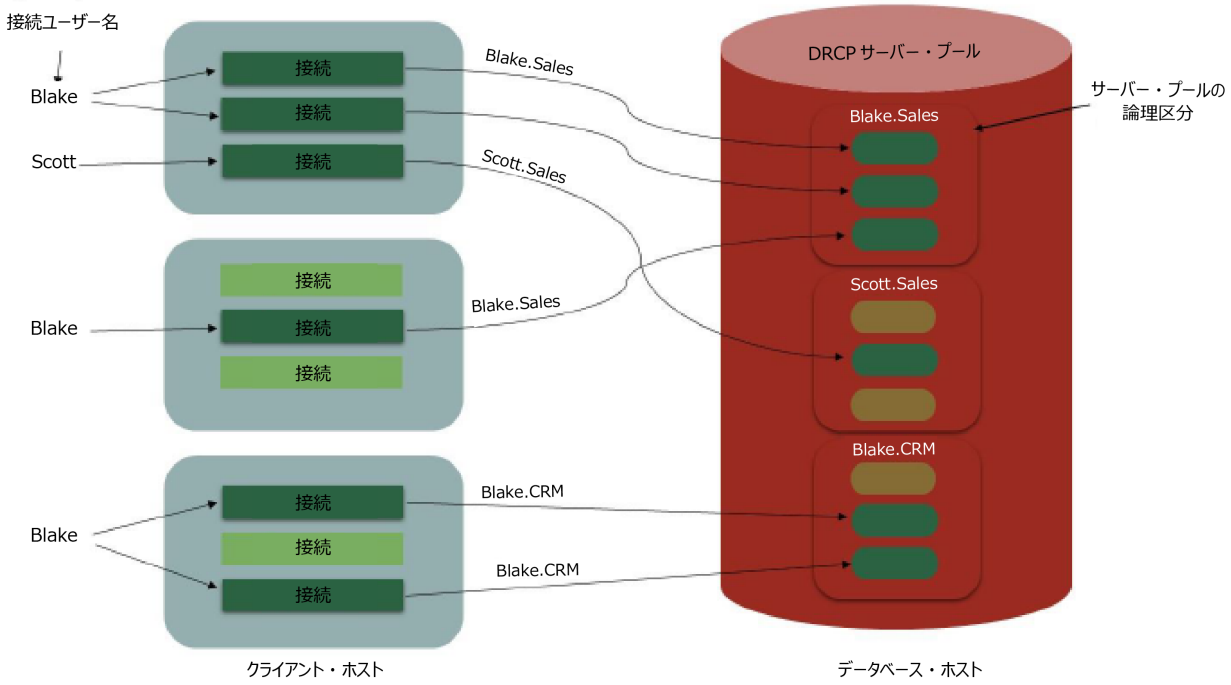

図3：複数のアプリケーションでのDRCPプール共有

セッション純正値とは

セッション純正値属性は、アプリケーションが"新しい"セッション（NEW）を必要とするか、アプリケーション・ロジックが"プール"セッション（SELF）を再利用するように設定されているかを指定します。この属性は、DRCPでのプール・セッションの再利用性を制御します。プール・セッションを再利用すると、接続パフォーマンスが向上します。

DRCPセッション純正値と接続クラスのデフォルト

DRCP接続の接続クラス属性と純度属性には、アプリケーションがローカル接続プールを使用しているかどうかに基づくデフォルト値があります。

DRCP接続設定	アプリケーション接続プールからの接続のデフォルト値	アプリケーション接続プールからではない接続のデフォルト値
純度	SELF	New
接続クラス	Oracle Call Interface（OCI）ライブラリを使用するアプリケーションの場合、アプリケーション・セッション・プールごとにランダムに生成された一意の名前が、セッション・プール内のすべての接続でデフォルトの接続クラスとして使用されます。 python-oracledb Thinモードでは、デフォルトで接頭辞"DPY"が付いた一意の接続クラス名が生成されます。 node-oracledb Thinモードでは、デフォルトで接頭辞"NJS"が付いた一意の接続クラス名が生成されます。 JDBC Thinの場合、デフォルトは、Universal Connection Pool（UCP）が設定されている場合に指定された接続プールの名前となります。UCPが設定されていない場合、クラスにはランダムな名前が付けられます。 マネージドODP.NETおよびODP.NET Coreでは、デフォルトはNULLです。	“SHARED”

表5 - セッション純正値と接続クラスのデフォルト

接続文字列でのセッション純正値と接続クラス

多くのデータベース・ドライバは、アプリケーションが接続クラスの値と純度の値を属性として設定するためのオプションを提供しています。ただし、アプリケーション・コードを介してこれらの属性値を設定するオプションが利用できない場合、またはこれらの値が最適でない場合は、接続文字列でパラメータPOOL_CONNECTION_CLASSおよびPOOL_PURITYを設定できます。SERVERがPOOLEDでない場合、これら2つのパラメータは無視されます。

接続文字列で指定されたPOOL_CONNECTION_CLASS属性とPOOL_PURITY属性は、もっとも高い優先順位を持ち、デフォルト値またはアプリケーション指定の値（Oracle Call Interface（C/C++）のOCI *OCIAttrSet*呼出しまたは*OCISessionGet*呼出し、あるいはPython、JDBC、ODP.NETシン・ドライバによって設定）をオーバーライドします。

POOL_PURITYの有効な値は、SELFとNEWです。この値では大文字と小文字は区別されません。

注：接続文字列でSELFを使用する場合、アプリケーションが*OCISessionRelease()*で*OCI_SESSRLS_DROPSESS*モードを渡した場合でも、アプリケーションの純度属性にNEWが指定されたセッション・リクエストは、アプリケーション・プールから削除されません。

POOL_CONNECTION_CLASSの値には、接続クラスのセマンティックに準拠する任意の文字列を指定できます。この値では大文字と小文字が区別されます。

サンプルの*Easy Connect*文字列：

```
oraclehost:1521/db_svc1:pooled?pool_purity=self&pool_connection_class=ccname
```

Easy Connect構文では、*pool_connection_class*属性と*pool_purity*属性を使用できます（Oracle Database 21c以降）。アプリケーションがOracle Clientライブラリを使用する場合、これらの属性はOracle Clientバージョン12以降のEasy Connect構文でサポートされます。

これらのDRCPパラメータの使用法の詳細については、Easy Connect構文に関する最新の技術概要を確認してください。

サンプルのネットワーク接続記述子文字列：

```
ServerPool =
  (DESCRIPTION =
          (ADDRESS=(PROTOCOL=tcp)(HOST=oraclehost)(PORT=1521))
          (CONNECT_DATA=(SERVICE_NAME=db_svc1)(SERVER=POOLED))
          (POOL_CONNECTION_CLASS=CCNAME)(POOL_PURITY=SELF))
```

PDBごとのDRCP

Oracle Database 12cで導入されたマルチテナント・オプションにより、コンテナ・データベース（CDB）モデルとプラガブル・データベース（PDB）モデルが導入されました。すべてのPDBで使用できるDRCPプールは1つだけであり、管理はSYSDBA権限を持つROOTユーザーによってCDBレベルで行われました。これは'CDB DRCP'と呼ばれます。

Oracle Database 21c以降、DRCPは、CDBレベル（**CDB DRCP**）またはPDBレベル（**PDBごとのDRCP**）のいずれかにすることができます。PDBごとのDRCPモードでは、PDB管理者ユーザー（例：*PDB1ADMIN*）は、そのPDBが所有するDRCPプールを構成、管理、監視できます。ブローカは引き続きROOTによって所有され、PDBごとのDRCPプールすべてによって共有されます。

PDBごとのDRCPの有効化

デフォルトでは、DRCPはCDBレベルです。CDB DRCPモードでは、CDBで実行されている単一のDRCPプールがすべてのPDBで共有されます。このモードでは、データベース初期化パラメータ`ENABLE_PER_PDB_DRCP`は`'FALSE'`になります。

`ENABLE_PER_PDB_DRCP`を`'TRUE'`に設定すると、PDBごとのDRCPを有効にすることができます。

*PDB1ADMIN*ユーザーが`DBMS_CONNECTION_POOL`パッケージにアクセスしてDRCP統計の問合せを実行するには、ROOTユーザー（*SYS*）が次の権限をPDB1ADMINに付与する必要があります。

```
GRANT CREATE SESSION, CREATE SYNONYM TO PDB1ADMIN;
GRANT EXECUTE ON DBMS_CONNECTION_POOL TO PDB1ADMIN;
GRANT SELECT ON V_$CPOOL_STATS TO PDB1ADMIN;
GRANT SELECT ON V_$CPOOL_CC_STATS TO PDB1ADMIN;
GRANT SELECT ON V_$CPOOL_CONN_INFO TO PDB1ADMIN;
GRANT SELECT ON V_$CPOOL_CC_INFO TO PDB1ADMIN;
GRANT SELECT ON V_$AUTHPOOL_STATS TO PDB1ADMIN;
```

DRCPプールの管理と監視を容易にするために、PDB管理者ユーザー（この場合は*PDB1ADMIN*）は、次のシノニムを作成できます。

```
CREATE SYNONYM DBMS_CONNECTION_POOL FOR SYS.DBMS_CONNECTION_POOL;
```

```
CREATE SYNONYM V$CPOOL_STATS FOR SYS.V_$CPOOL_STATS;
CREATE SYNONYM V$CPOOL_CC_STATS FOR SYS.V_$CPOOL_CC_STATS;
CREATE SYNONYM V$CPOOL_CONN_INFO FOR SYS.V_$CPOOL_CONN_INFO;
CREATE SYNONYM V$CPOOL_CC_INFO FOR SYS.V_$CPOOL_CC_INFO;
CREATE SYNONYM V$AUTHPOOL_STATS FOR SYS.V_$AUTHPOOL_STATS;
```

これが完了すると、各PDB管理者によるプール管理は、PDBレベルでのみ許可されます。

CDB DRCPとPDBごとのDRCPの比較

CDB DRCPでは、DBA ROOTユーザー（SYSなど）がCDB内のDRCPプールを管理します。すべてのPDBは、この単一のDRCPプールを共有します。

CDBからのプールの管理	PDBからのプールの管理	CDBからのプール統計の表示	PDBからのプール統計の表示
ROOTユーザーは、CDBに接続すると、`dbms_connection_pool`パッケージのすべてのプロシージャ（例：*start_pool()*、*stop_pool()*）を実行できます。 データベース・パラメータ*connection_brokers*が*init.ora*または*ALTER SYSTEM*によって設定されている場合、*alter_param()*プロシージャは、DRCP構成パラメータ*num_cbrok*および*maxconn_cbrok*を変更できません。	CDB DRCPは、どのユーザーもPDBから管理することはできません。	ROOTユーザーは、次のgv$表の問合せを実行できます。*gv$cpool_stats*、*gv$cpool_cc_stats*、*gv$cpool_conn_info*、*gv$authpool_stats*、*gv$cpool_cc_info*、それらに対応するv$表およびDBA_CPOOL_INFO。	PDBに接続しているROOTユーザーは、*gv$cpool_conn_info*および*gv$authpool_stats*とそれらに対応するv$表からの統計のみを表示できます。

表6 - CDB DRCPの動作

PDBごとのDRCPでは、PDB管理者ユーザーが個々のPDBのDRCPプールを管理します。

CDBからのプールの管理	PDBからのプールの管理	CDBからのプール統計の表示	PDBからのプール統計の表示
CDBに接続している場合、ROOTユーザーまたはPDB管理者ユーザーは、`dbms_connection_pool`パッケージのプロシージャを実行できません。 ROOTユーザーは、データベース・パラメータ*connection_brokers*を使用して、*num_cbrok*および*maxconn_cbrok*の値を変更できます。	PDBに接続している場合、PDB管理者ユーザーのみが、`dbms_connection_pool`パッケージのすべてのプロシージャ（例：*start_pool()*、*stop_pool()*）を実行できます。 *alter_param()*プロシージャは、DRCP構成パラメータ*num_cbrok*および*maxconn_cbrok*を変更できません。PDB管理者ユーザーは、データベース・パラメータ*connection_brokers*を変更できません。	ROOTユーザーは、次のgv$表と、CDBに接続済みの対応するv$表の問合せを実行できます。*gv$cpool_stats*、*gv$cpool_cc_stats*、*gv$cpool_conn_info*、*gv$authpool_stats*、*gv$cpool_cc_info* 結果には、すべてのPDBに関するDRCP情報が含まれます。	PDB管理者ユーザーまたはROOTユーザーは、PDBから次のgv$表の問合せを実行できます。*gv$cpool_stats*、*gv$cpool_cc_stats*、*gv$cpool_conn_info*、*gv$authpool_stats*、*gv$cpool_cc_info*、それらに対応するv$表および*DBA_CPOOL_INFO*。 結果には、特定のPDBに関するDRCP情報が含まれます。

表7 - PDBごとのDRCPの動作

DRCPでの暗黙的接続プーリング

Oracle Database 23aiでは、暗黙的接続プーリングが導入されました。これをDRCPとともに使用すると、データベースがSQLまたはPL/SQLトランザクションの特定の境界要件に基づいて接続/セッションを自動的に解放できるようになり、アプリケーションでのプール管理の責任を軽減できます。暗黙的接続プーリングは、PDBごとのDRCPとCDB DRCPの両方で機能します。

DRCPの暗黙的接続プーリングは、データベース接続/セッションがステートレスである（オープン・カーソル、一時LOB、一時表、またはアクティブなトランザクションがない）ことを検出し、データベース接続上で'暗黙的解放'を実行します。

'暗黙的解放'プロセスには、次の2つのステップが関係します。

- 接続を接続ブローカにハンドバックする
- プール・サーバーをセッションとともにDRCPの'空きサーバー'プールに戻す

'暗黙的解放'プロセスは、アプリケーションに認識されずに実行されます。接続上の後続のデータベース呼出しは、暗黙的にDRCPプールからセッションを取得します。

DRCPでの暗黙的接続プーリングは、次のようなアプリケーションに適しています。

- データベース作業を実行していないときにも接続を保持するアプリケーション
- カスタム・プーリングを使用するアプリケーション、またはプーリングをまったく使用しないアプリケーション
- スパースであるが反復的なワークロード処理を行うアプリケーション
- OracleプーリングAPIへの移行が難しい、レガシーまたはサード・パーティのコードを含むアプリケーション

詳細な考察については、暗黙的接続プーリングのブログをご覧ください。

暗黙的接続プーリングの有効化

アプリケーションが暗黙的接続プーリングを使用できるようにするには、まずデータベース・サーバーでDRCPを構成します。

アプリケーション側では、接続文字列にPOOL_BOUNDARYオプションを設定して、アプリケーションが暗黙的接続プーリングを使用できるようにする必要があります。POOL_BOUNDARYオプションには、次の2つの値を指定できます。

- STATEMENT - データベース・セッションがステートレスであるとき、DRCPは'暗黙的解放'を実行します。
- TRANSACTION - コミット/ロールバック時、またはデータベース・セッションがステートレスであるとき、DRCPは'暗黙的解放'を実行します。この解放では、コミット/ロールバックの場合、アクティブなカーソル、一時表、一時LOBがすべて閉じられます。

暗黙的接続プーリングを使用した、サンプルの*Easy Connect*文字列：

```
oraclehost:1521/db_svc_name:pooled?pool_boundary=statement
```

暗黙的接続プーリングを使用した、サンプルのネットワーク接続記述子文字列：

```
DBServerPool =
  (DESCRIPTION =
         (ADDRESS=(PROTOCOL=tcp)(HOST=oraclehost)(PORT=1521))
```

```
        (CONNECT_DATA=(SERVICE_NAME=db_svc_name)(SERVER=POOLED)
        (POOL_BOUNDARY=STATEMENT))
)
```

暗黙的接続プーリングでの接続において、セッション純正値値のデフォルトは'SELF'です。暗黙的接続プーリングでは、アプリケーションにその他の変更を加える必要はありません。

暗黙的接続プーリングのメリット

DRCPでの暗黙的接続プーリングにより、アプリケーションによる明示的な接続呼出しの開始や終了に依存せず、データベース接続の多重化が強化されます。この暗黙的接続プーリングにより、アプリケーションは接続を長時間開いたままとして、データベース・サーバー・プロセスとセッション・メモリを共有できるようになります。これにより、データベース・ホストの負荷が軽減され、システム全体のスケーラビリティが向上します。

要約すると、DRCPでの暗黙的接続プーリングは、アプリケーションに次のメリットをもたらします。

- 多重化の強化によりアプリケーションのスケーラビリティが向上
- アプリケーション側で必要なプール処理を削減
- 最適なデータベース・リソース使用率により、中間層のより高い同時実行性をサポート

複数プールDRCP（名前付きプール）

Oracle Database 23aiでは、複数プールDRCPが導入され、異なる構成で複数の名前付きプールを作成できるようになりました。この機能を使用すると、データベース管理者はDRCPプールを追加または削除できます。複数プールDRCPは、CDBレベルとPDBレベルの両方で構成できます。アプリケーションが特定のDRCPプールにアクセスするには、接続文字列でプール名を指定する必要があります。

複数プールDRCPは、データベース管理者（DBA）に構成の柔軟性を提供できます。さらに、受信するアプリケーション・リクエストのタイプに基づいてデータベース接続を編成するのに役立ちます。

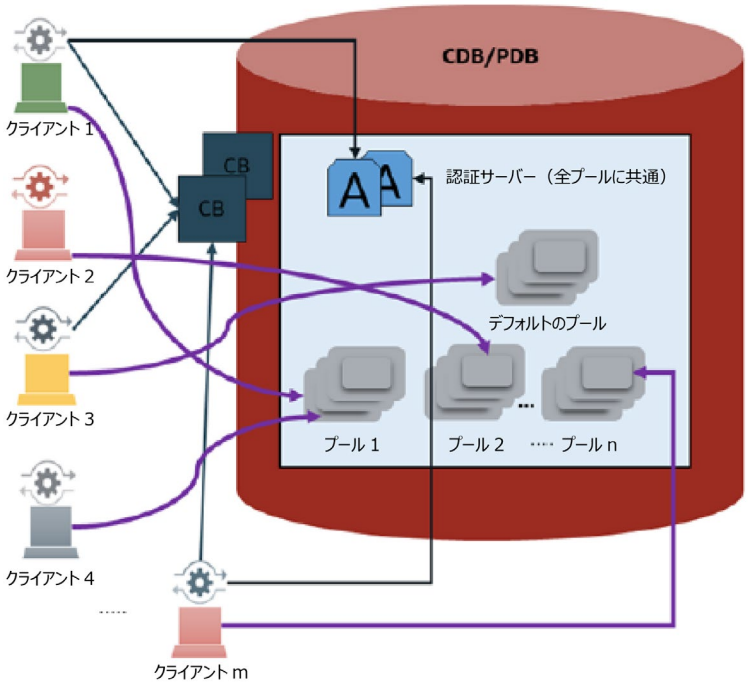

図4：複数プールDRCPアーキテクチャ

名前付きプールの追加および削除

新しいPL/SQLプロシージャdbms_connection_pool.add_pool()によって新しいプールが追加されます。

デフォルトのプール・パラメータを使用して'my_pool'という名前の新しいプールを追加するには、次を実行します。

```
SQL> execute dbms_connection_pool.add_pool('my_pool')
```

もう1つの新しいPL/SQLプロシージャであるdbms_connection_pool.remove_pool()は、プールを削除します。

'my_pool'を削除するには、次を実行します。

```
SQL> execute dbms_connection_pool.remove_pool('my_pool')
```

複数プールDRCPの構成

複数プールDRCPでは、DRCPを有効にすること以外に、データベース・サーバーで追加の構成は必要ありません。

アプリケーションは、DRCPが適切なプールに対するクライアント接続を選択するために、接続文字列で(POOL_NAME=<*pool_name*>)と(SERVER=POOLED)を指定する必要があります。たとえば、次のとおりです。

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_name)(PORT=port_number))
(CONNECT_DATA=(SERVICE_NAME=db_service.company.com>)(SERVER=POOLED)
(POOL_NAME=my_pool)))
```

Easy Connect文字列を使用している場合は、次のようにしてプール名を指定できます。

```
host_name:port_number/db_service.company.com:pooled?pool_name=my_pool
```

本書の「サーバー側でのDRCPの有効化と構成」セクションで説明しているように、DBMS_CONNECTION_POOLパッケージのPL/SQLプロシージャを使用して、デフォルトのDRCPプールと類似の名前付きプールを構成できます。DRCPが有効になっている場合でも、SYS_DEFAULT_CONNECTION_POOLは自動的に作成され、デフォルトのDRCPプールのままであることに注意してください。

複数プールDRCPの構成と操作の詳細については、ブログ「Multi-pool Database Resident Connection Pooling（DRCP）in Oracle Database 23ai」を参照してください。

DRCPの監視

Oracle Databaseでは、DRCPのパフォーマンスを監視するために、組込みのデータ・ディクショナリ・ビューと動的パフォーマンス・ビューを使用できます。データベース管理者は、ビジー・サーバーと空きサーバーの数、クライアント・リクエストの総数に対するプール内のヒットとミスの数などの統計を確認できます。

DRCP統計を参照するためにOracle Databaseで使用できる組込みビューは次のとおりです。

```
DBA_CPOOL_INFO
V$CPOOL_STATS
V$CPOOL_CC_STATS
V$CPOOL_CONN_INFO
V$CPOOL_CC_INFO
V$AUTHPOOL_STATS
```

次のサブセクションでは、SQL*Plusを使用し、データ・ディクショナリ・ビューに対してDRCPに関する問合せを実行します。

DBA_CPOOL_INFO

DBA_CPOOL_INFOビューには、プールのステータス、接続の最大数と最小数など、接続プールに関する構成情報が表示されます。

次の例では、プールが開始されているかどうか（*ACTIVE*ステータス）を確認し、許可されているプール・サーバーの最大数を見つけます。

```
SQL> SELECT connection_pool, status, maxsize FROM dba_cpool_info;
CONNECTION_POOL                 STATUS          MAXSIZE
------------------------------  ----------      ----------
SYS DEFAULT CONNECTION POOL     ACTIVE            40
```

V$CPOOL_STATSビュー

V$CPOOL_STATSビューには、データベース・インスタンスのDRCP統計に関する情報が表示されます。V$CPOOL_STATSビューでは、接続プール設定の効率を評価できます。

次の例の問合せは、アプリケーションがプールを効果的に使用していることを示しています。ミスの数が少ないということは、サーバーとセッションが共有アプリケーションによって再利用されており、その純度が'SELF'であることを示しています。待機数は、プール・サーバーが使用可能になるまでに、10 %をやや超える数のリクエストが待機する必要があったことを示しています。

```
SQL> SELECT num_requests, num_hits, num_misses, num_waits FROM v$cpool_stats;

NUM_REQUESTS      NUM_HITS      NUM_MISSES       NUM WAITS
------------    ----------      ----------      ----------

       10031          99990              40            1055
```

接続クラスが設定されている場合（プール・サーバーとセッションの再利用が許可されている場合）、NUM_MISSESは小さくなります。プールの*maxsize*値が接続負荷に対して小さすぎる場合、NUM_WAITSは大きくなります。

CDBレベルのDRCPが有効な場合、このビューはCDBルート（SYSユーザー）から問合せが実行された場合にのみデータを返し、PDBから問合せが実行された場合は0行を返します。PDBごとのDRCPを使用すると、このビューはCDBルート（SYSユーザー）およびPDB（PDBADMINユーザー）から問合せが実行された場合にデータを返します。

注：SQL*PlusはデフォルトではPURITYを設定しないため、DRCPセッションを再利用しません。

V$CPOOL_CC_STATSビュー

ビューV$CPOOL_CC_STATSには、インスタンスごとのプールの接続クラス・レベルの統計が含まれています。たとえば、次のとおりです

```
SQL> SELECT cclass_name, num_requests, num_hits, num_misses
     FROM v$cpool_cc_stats;
CCLASS_NAME                      NUM_REQUESTS         NUM_HITS NUM_MISSES
-------------------------------  --------------------  -------- ----------
```

。

```
HR.MYCLASS                      100031          99993       38
```

CDBレベルのDRCPが有効な場合、このビューはCDBルート（SYSユーザー）から問合せが実行された場合にのみデータを返し、PDBから問合せが実行された場合は0行を返します。PDBごとのDRCPの場合、このビューはCDB（ルート・ユーザー）およびPDB（PDBADMINユーザー）から問合せが実行されたときにデータを返します。

Oracle Database 23aiでは、新しい*POOL_NAME*列がV$CPOOL_CC_STATSビューに追加されており、使用可能な場合に、名前付きプールの接続クラス統計が維持されます。

V$CPOOL_CONN_INFOビュー

ビューV$CPOOL_CONN_INFOを監視して、接続クラスが正しく設定されていないなど、不適切に構成されたマシンを特定できます。このビューには、接続ブローカへの各接続の接続情報が表示されます。次の例の問合せは、マシン名をクラス名にマップします。

```
SQL> SELECT cclass_name, machine FROM v$cpool_conn_info;

CCLASS_NAME                                             MACHINE
------------------------------------                    ------------

GK.OCI:SP:wshbIFDtb7rgQwMyuYvodA                        gklinux
```

この例では、Linuxマシン（*gklinux*）上のアプリケーションを調べて、*cclass*が設定されていることを確認します。V$CPOOL_CONN_INFOビューのその他の使用例については、こちらをご覧ください。

Oracle Database 23aiでは、新しい*POOL_NAME*列がこのビューに追加されており、使用可能な場合に、名前付きプールの接続プール情報が維持されます。

V$CPOOL_CC_INFOビュー

V$CPOOL_CC_INFOは、各データベース・インスタンスのDRCPプールにおける、プールと接続クラスのマッピングに関する情報を保持します。次の例の問合せでは、データベース・インスタンス内のすべての接続クラスを識別します。

```
SQL> SELECT pool_name, cclass_name FROM v$cpool_cc_info;

POOL_NAME                               CCLASS_NAME                     CON_ID
--------------------------------------- ------------------------------- ------
SYS_DEFAULT_CONNECTION_POOL             HR.MYCLASS                         3
```

この例では、ユーザーは*HR*で、接続クラスは*MYCLASS*です。

V$AUTHPOOL_STATSビュー

V$AUTHPOOL_STATSは、DRCP認証サーバーの統計を示します。このビューは、Oracle Database 21c以降で利用可能です。次の例の問合せでは、認証サーバーの統計を参照します。

```
SQL> select num_srvs, num_busy, num_free, num_waiters from v$authpool_stats;

     NUM_SRVS     NUM_BUSY    NUM_FREE    NUM_WAITERS
---------- ---------- ---------- -----------

         3          0          3           0
```

上記の例は、3つの認証サーバー・プロセスが解放されており、接続認証リクエストを受信できる状態であることを示しています。

さまざまな言語でのDRCPの例

アプリケーションでDRCPを有効にして使用するには、次のようにする必要があります。

1. データベースでDRCPを構成して有効にする
2. DRCP接続を使用するようにアプリケーションを構成する
3. アプリケーションをデプロイする

DRCP用にデータベースを構成せずに次のコード・スニペットを実行すると、接続は成功せず、アプリケーションにエラーが返されます。

PythonでのDRCP

Oracle Database向けの最新のPythonインタフェースである*python-oracledb*（パッケージ名：*oracledb*）は、DRCPをサポートしています。

DRCPを使用するためのアプリケーション・デプロイメント

データベースに対してDRCPプール・サーバーを使用するようにリクエストするためには、次の構文のいずれかに似た特定の接続文字列を*oracledb.create_pool()*または*oracledb.connect()*で使用します。

オラクルのEasy Connect構文を使用すると、接続パラメータは次のようになります。

```
import oracledb
connection = oracledb.connect(user="hr", password=userpwd,
                              dsn="dbhost.example.com/orcl:pooled",
                              cclass="MYAPP")
```

または、*customerdb*という名前のtnsnames.oraエイリアスを使用して接続する場合は、次のようになります。

```
connection = oracledb.connect(user="hr", password=userpwd,
                              dsn="customerdb")
```

この場合、Oracleネットワーク構成ファイル*tnsnames.ora*のみを次のように変更する必要があります。

```
customerdb = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=dbhost.example.com)
             (PORT=1521))(CONNECT_DATA=(SERVICE_NAME=CUSTOMER)(SERVER=POOLED)))
```

スタンドアロン接続またはpython-oracledb接続プールを作成するときに、*server_type*パラメータを設定することで、DRCPプール・サーバーを使用するように指定することもできます。

たとえば、次のとおりです。

```
pool = oracledb.create_pool(user="hr", password=userpwd,
                            dsn="dbhost.example.com/orclpdb", min=2, max=5,
                            increment=1, server_type="pooled")
```

接続クラス属性と純度属性の設定

ユーザーが選択したこの名前により、DRCPセッション・メモリがある程度パーティション化されます。したがって、再利用は類似のアプリケーションに限定されます。複数のアプリケーション・プロセスが開始される場合、最大のプール共有が提供されます。

接続クラス名（*cclass*属性）を使用してDRCPサーバーをリクエストするアプリケーション接続プールを作成し、接続を取得するには、次のようにします。

```
pool = oracledb.create_pool(user="hr", password=userpwd,
                            dsn="dbhost.example.com/orclpdb:pooled",
                            min=2, max=5, increment=1,
                            cclass="MYAPP")
connection = pool.acquire()
```

プール内におけるすべての接続の純度は、デフォルトでSELF（python-oracledbの*PURITY_SELF*値）に設定されます。これは推奨されるベスト・プラクティスでもあります。

python-oracledb接続プールのサイズは、DRCPプールのサイズと一致する必要はありません。DRCPプールのサイズによって、全体的な実行並列度の制限が決まります。

接続クラス名は、次のようにして*acquire()*関数に渡すこともできます。

```
connection = pool.acquire(cclass="OTHERAPP")
```

プール接続の純度を*NEW*に変更するには、*acquire()*関数で純度属性をPURITY_NEWに設定します。

```
connection = pool.acquire(cclass="MYAPP", purity=oracledb.PURITY_NEW)
```

この接続オブジェクトを使用して、任意のデータベース・トランザクションを実行できます。

```
with connection.cursor() as cursor:
    print("Performing query using DRCP...")
    for row in cursor.execute("select sysdate from dual"):
        print(row)
```

このコード・スニペットは、データベース・ホストの現在のシステム日付を出力します。

*cclass*パラメータとSELF純度が設定されていない場合、プール・サーバー・セッションは最適に再利用されることはなく、DRCP統計ビューでNUM_MISSESに大きな値が記録される可能性があります。

DRCPを使用すると、接続がプールから取得されるたびに、接続のセッション・メモリを再利用またはクリーンアップできます。プールまたは接続の作成では、純度パラメータ値は、PURITY_NEW、PURITY_SELF、またはPURITY_DEFAULTになります。
デフォルトでは、python-oracledbのプール接続はPURITY_SELFを使用し、スタンドアロン接続はPURITY_NEWを使用します。

接続文字列での接続クラスと純度の設定

python-oracledb Thinモードの場合、バージョン21c以降のOracle DatabaseのEasy Connect文字列で接続クラスと純度を指定できます。これにより、DRCPを使用したいときに既存のアプリケーションを変更する必要がなくなります。

```
dsn = "localhost/orclpdb:pooled?pool_connection_class=MYAPP&pool_purity=self"
```

Node.jsでのDRCP

node-oracledbでDRCPを使用するには、Oracle Database用のNode.jsドライバで次のようにします。

*oracledb.createPool()*または*oracledb.getConnection()*のプロパティconnectString（またはそのエイリアスの*connectionString*）では、*myhost/sales:POOLED*のようなEasy Connect構文によって、または*(SERVER=POOLED)*を含むネットワーク接続記述子文字列のtnsnames.oraエイリアスを使用して、プール・サーバーを使用することを指定する必要があります。

効率のために、DRCP接続は、node-oracledbのローカル接続プールで使用することをお勧めします。

例：

```
Easy Connect string: dbhost.us.oracle.com:2222/dbsvc.company.com:POOLED
```

または

```
tnsnames.ora:
customerpool = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=dbhost.example.com)
(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=CUSTOMER)(SERVER=POOLED)))
```

node-oracledbのローカル接続プールを使用している場合、次のコード・スニペットが機能します。

*DRCP*を指す*Easy Connect*文字列

```
const oracledb = require('oracledb');
const pool = await oracledb.createPool({
  user: "scott",
  password: "tiger",
  connectString: "dbhost.us.oracle.com:2222/dbsvc.company.com:POOLED",
  poolMax:1,
  poolMin:1,
  poolPingInterval:0,
});
const connection = await pool.getConnection();
```

または

*DRCP*を指す*tnsnames.ora*エイリアス

```
const oracledb = require('oracledb');
const pool = await oracledb.createPool({
  user: "scott", password: "tiger",
  connectString: "customerpool",
  poolMax:1, poolMin:1, poolPingInterval:0
});
const connection = await pool.getConnection();
```

スタンドアロン接続の場合、次のコード・スニペットが機能します。

*DRCP*を指す*Easy Connect*文字列

```
const connection = await oracledb.getConnection({
  user: "scott",
  password: "tiger",
  connectString: "dbhost.us.oracle.com:2222/dbsvc.company.com:POOLED"
});
```

または

*DRCP*を指す*tnsnames.ora*エイリアス

```
const connection = await oracledb.getConnection({
  user: "scott",
  password: "tiger",
  connectString: "customerpool"
});
```

上記のコード・スニペットはすべて*Async*関数内に記述する必要があることに注意してください。これらのコード・スニペットをすべて*try-catch-finally*ブロックに埋め込むことをお勧めします。

この接続オブジェクトを使用して、任意のデータベース・トランザクションを実行できます。

```
console.log("System Date:");
const result = await connection.execute(
  `SELECT sysdate
   FROM dual`
);
const ts = result.rows[0][0];
console.log(ts);
if (connection) await connection.close();
```

このプログラムはデータベース・ホストの現在のシステム日付を出力し、接続を閉じます。

接続クラスと純度の設定

node-oracledbは、接続クラス名を設定するための*connectionClass*属性を提供します。

```
oracledb.connectionClass = "NodePool";
```

'純度'の値は、node-oracledbでのDRCP接続の場合には常にSELFです。これにより、プール・サーバー・プロセスとセッション・メモリの両方を再利用できるようになり、DRCPから最大限のメリットが得られます。node-oracledbには、接続の純度を設定するための別個のパラメータや関数はありません。

接続クラスと純度の値は、「PythonでのDRCP」サブセクションに示されているEasy Connect構文を使用して設定することもできます。

JDBCでのDRCP

Oracle JDBCドライバはDRCPをサポートします。

DRCP実装では、サーバー側にプールが作成され、複数のクライアント・プール間で共有されます。JDBCアプリケーションは、アプリケーション接続プーリングにOracle Universal Connection Pool（Oracle UCP）を使用します。Oracle UCPは（サーバー・プロセスの数が減るため）メモリ消費量を大幅に削減し、データベース・レイヤーのスケーラビリティを向上させます。

Javaアプリケーションは、サーバー側の接続のチェックイン操作とチェックアウト操作を追跡するために、Oracle UCP for JDBCなどのアプリケーション接続プールまたはサード・パーティのJava接続プールを使用する必要があります。サード・パーティのクライアント・プール上でOracle UCPを使用するメリットは、Oracle UCPがサーバー接続のアタッチとデタッチを透過的に処理することです。

何らかの理由でOracle UCPが使用されない場合、接続では、*attachServerConnection()*関数と*detachServerConnection()*関数を使用して、それぞれで接続ブローカへの接続のアタッチとデタッチを行う必要があります。

クライアント側でDRCPを有効にするには、次を実行する必要があります。

• NULL以外の、空でない文字列値を、DRCP接続クラス・プロパティに渡します。

```
oracle.jdbc.DRCPConnectionClass
```

• ネットワーク接続記述子文字列で(SERVER=POOLED)を渡します。

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=<port>))(CONNECT_DATA=
(SERVICE_NAME=<service name>)(SERVER=POOLED)))
```

次のように短縮URL形式で(SERVER=POOLED)を指定することもできます。

```
jdbc:oracle:thin:@//<host>:<port>/<service_name>[:POOLED]
```

たとえば、次のとおりです。

```
jdbc:oracle:thin:@//localhost:5221/orclpdb:POOLED
```

接続プロパティ*oracle.jdbc.DRCPConnectionClass*を使用して、すべてのプール・サーバー・プロセスに同じDRCP接続クラス名を設定すると、サーバー上のプール・サーバー・プロセスを複数の接続プール間で共有できます。

DRCPでは、特定の接続にタグを適用し、後でそのタグ付き接続を簡単に取得することもできます。

Oracle Universal Connection Pool（Oracle UCP）を使用してクライアント側でDRCPを有効にする

*oracle.ucp.jdbc*パッケージの*PoolDataSource*オブジェクトは、UCPの作成に使用されます。

```
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;
import java.util.Scanner;
import oracle.ucp.jdbc.PoolDataSource;
```

```
import oracle.ucp.jdbc.PoolDataSourceFactory;
public class DRCPSamplewithUCP{
final static String url = "jdbc:oracle:thin:@//localhost:1522/orclpdb:POOLED";

  static public void main(String args[]) throws SQLException {
    PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
    pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
    Scanner sc = new Scanner(System.in);

    // DataSourceプロパティの設定 - DB資格証明を入力として取得します
    System.out.print("Enter the DB User name: ");
    String dbUser = sc.nextLine();
    System.out.print("Enter the DB password: ");
    String dbPassword = sc.nextLine();
    System.out.println ("Connecting to " + url);
    pds.setUser(dbUser);
    pds.setPassword(dbPassword);
    pds.setURL(url);

    // UCPプロパティを設定します
    pds.setInitialPoolSize(1);
    pds.setMinPoolSize(4);
    pds.setMaxPoolSize(20);

    // Universal Connection Poolからデータベース接続を取得します
    try (Connection conn = pds.getConnection()) {
      System.out.println("\nConnection obtained from UniversalConnectionPool");
      // データベース操作を実行します
      doSQLWork(conn);
      System.out.println("Connection returned to the UniversalConnectionPool");
    }
  }

  // システム日付（sysdate）を表示します
  public static void doSQLWork(Connection connection) throws SQLException {
    // StatementとResultSetはこの構文により自動クローズ可能です
```

```
    try (Statement statement = connection.createStatement()) {
      try (ResultSet resultSet = statement
        .executeQuery("select SYSDATE from DUAL")) {
          while (resultSet.next())
            System.out.print("Today's date is " + resultSet.getString(1) + " ");
      }
    }
    System.out.println("\n");
  }
}
```

このコードは、DRCPを介したUCP JDBC接続を使用して、システム日付を出力します。この例では、Easy Connect構文を使用しました。

JDBC接続文字列は、TNS URL形式もサポートします。

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=<protocol>)
(HOST=<dbhost>)(PORT=<dbport>)) (CONNECT_DATA=(SERVICE_NAME=<service-name>))
```

接続クラス属性と純度属性の設定

接続クラスは、*oracle.jdbc.DRCPConnectionClass*プロパティを通じて設定されます。Oracle UCPが使用されている場合、接続クラスのデフォルトは、UCPプールの名前が設定されているときにはその名前になり、設定されていないときにはランダムに生成された名前になります。

Javaの*Properties*オブジェクトを介して接続クラス名を設定し、それをOracle UCPのConnectionプロパティに追加できます。

```
Properties prop = new Properties();
prop.put("oracle.jdbc.DRCPConnectionClass", "MyConClass");
pds.setConnectionProperties(prop);
```

DRCP接続の純度は、必要に応じて、*oracle.jdbc.DRCPConnectionPurity*プロパティ（デフォルト：SELF）を介して設定されます。

```
prop.put("oracle.jdbc.DRCPConnectionPurity", "NEW");
```

Oracle Call Interface（OCI）でのDRCP

Oracle Call Interface（OCI）ライブラリは、Oracle Databaseにアクセスして操作するためのC言語APIを提供します。DRCPの場合、最適なパフォーマンスを実現するには、OCIセッション・プールAPIである*OCISessionPoolCreate()*、*OCISessionGet()*、*OCISessionRelease()*を使用する必要があります。

OCIアプリケーションは、アプリケーションおよびDRCPプール設定に対して適切に設定された*sessMin*、*sessMax*、*sessIncr*の各パラメータを使用して*OCISessionPoolCreate()*を呼び出すことにより、OCIセッション・プールの環境を初期化します。DRCPを使用するシングルスレッド・アプリケーションでは、*sessMin*を0または1に、*sessMax*を1に設定します。DRCPとアプリケーション側接続プーリングを使用するマルチスレッド・アプリケーションでは、スレッド要件に基づいて*sessMin*パラメータと*sessMax*パラメータ（1以上）を設定します。

OCIアプリケーションがDRCPのOCIセッション・プールからセッションを取得するには、modeパラメータに*OCI_SESSGET_SPOOL*を指定して、*OCISessionGet()*を呼び出します。アプリケーションがセッションを解放してDRCPのOCIセッション・プールに戻すには、*OCISessionRelease()*を呼び出します。

OCIセッション・プールは、接続ブローカへの接続を透過的にキャッシュして、パフォーマンスを向上させることができます。OCIアプリケーションは、*OCISessionGet()*を呼び出す前に、*OCIAuthInfo*ハンドルを使用して、*OCIAttrSet()*関数の接続クラスに*OCI_ATTR_CONNECTION_CLASS*属性を設定することで、同様の状態のセッションを再利用できます。

セッション純正値は、OCIアプリケーションがプール・セッションを再利用できるか（*OCI_SESSGET_PURITY_SELF*）、新しいセッションを利用できるか（*OCI_SESSGET_PURITY_NEW*）を指定します。

*OCISessionGet()*は、*OCI_SESSGET_PURITY_NEW*または*OCI_SESSGET_PURITY_SELF*のセッション純正値設定値を取り込むことができます。あるいは、アプリケーションは、*OCISessionGet()*を呼び出す前に、*OCIAuthInfo*ハンドルに*OCI_ATTR_PURITY_NEW*または*OCI_ATTR_PURITY_SELF*を設定できます。これらのメソッドはどちらも同等です。OCIセッション・プールのデフォルトの純度値はSELFで、スタンドアロン接続はNEWです。

DRCPを使用したOCI関数のセッション純正値と接続クラスの動作

次の表は、Oracle Call Interface（OCI)ライブラリを使用するアプリケーションでのセッション純正値属性と接続クラス属性の動作を示しています。

アプリケーションの純度	接続文字列の純度	アプリケーションのセッション解放モード	セッション解放動作	セッション取得動作
未設定、NEW、またはSELF	New	デフォルト / OCI_SESSRLS_DROPSESS（セッションを削除するためのOCI属性）	セッションは削除されます	新規セッション
New	SELF	デフォルト	セッションは保持されます	一致するセッションが見つかった場合は既存のセッションを取得し、それ以外の場合は新規セッションとなります
New	SELF	OCI_SESSRLS_DROPSESS	セッションは保持されます	一致するセッションが見つかった場合は既存のセッションを取得し、それ以外の場合は新規セッションとなります
SELF	SELF	デフォルト	セッションは保持されます	一致するセッションが見つかった場合は既存のセッションを取得し、それ以外の場合は新規セッションとなります
SELF	SELF	OCI_SESSRLS_DROPSESS	セッションは削除されます	一致するセッションが見つかった場合は既存のセッションを取得し、それ以外の場合は新規セッションとなります

表8 - DRCPを使用したOCIアプリケーションのセッション純正値と接続クラスの動作

デフォルトの純度は、非セッション・プール・アプリケーションの場合にはNEW、セッション・プール・アプリケーションの場合にはSELFです。

接続文字列にPOOL_PURITY=SELFが含まれている場合、セッションの再利用が必要になります。純度がNEWであり、*OCISessionRelease()*で*OCI_SESSRLS_DROPSESS*が指定されたセッションでは、セッションの削除が必須となり、セッションの再利用が防止されます。

接続文字列を介してPOOL_PURITY=SELFが指定されたアプリケーションと、純度がNEWであり、*OCISessionRelease()*で*OCI_SESRLS_DROPSESS*が指定されたセッションの場合、セッションの再利用機能がより重要であると認識されます。したがって、*OCI_SESRLS_DROPSESS*はサーバーによって無視され、セッションは削除されません。

セッションの再利用ではなく、*OCISessionRelease()*での*OCI_SESRLS_DROPSESS*を優先するアプリケーションでは、接続文字列でPOOL_PURITY=SELFを使用しないでください。

新しいセッションの接続クラス属性とセッション純正値属性の設定

次のコード・スニペットは、接続プーリングOCIアプリケーションが新しいDRCPセッションを設定する方法を示しています。

```
#include <oci.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

OraText userName[129];
OraText userPassword[129];


/* "プールされた"接続をリクエストします */
const OraText connectString[] = "localhost:1522/orclpdb.domain.com:pooled";

/* DRCP接続クラス名 */
const OraText connectionClassName[] = "OCIConnectionPool";

int main(int argc, char** argv)
{
  OCIEnv *envhp = NULL;
  OCIError *errhp = NULL;
  OCIAuthInfo *authInfop = NULL;
  OCISvcCtx *svchp = NULL;
  OraText *poolName = NULL;
  ub4 poolNameLen = 0;
  OCISPool *spoolhp = NULL;
  int rc;

  /* 必要に応じてエラー処理やその他の変数を追加します */

  // DBコンテキストを初期化します
  rc = OCIEnvNlsCreate(&envhp, OCI_DEFAULT, 0, NULL, NULL, NULL, 0, NULL, 0, 0);
  /* 一貫性を保つためにエラー処理コードを以下に追加します... */

  // すべてのハンドル（エラー、認証、セッション・プール）を初期化します
  rc = OCIHandleAlloc(envhp, (void **)&errhp, OCI_HTYPE_ERROR, 0, NULL);
  /* 以下にエラー処理コードを追加します... */
```

```
  rc = OCIHandleAlloc(envhp, (void **)&authInfop, OCI_HTYPE_AUTHINFO, 0, NULL);
  /* 以下にエラー処理コードを追加します... */

  rc = OCIHandleAlloc(envhp, (void **)&spoolhp, OCI_HTYPE_SPOOL, 0, NULL);
  /* 以下にエラー処理コードを追加します... */

  // ユーザー入力を介してDB資格証明を取得します（推奨）
  printf("Enter the DB username: ");
  scanf("%s", userName);
  printf("Enter the DB password: ");
  scanf("%s", userPassword);

  // セッション・プールを作成します
  rc = OCISessionPoolCreate(envhp, errhp, spoolhp, &poolName, &poolNameLen,
    connectString, strlen((char *)connectString), 0, UB4MAXVAL, 1,
    (OraText *)userName, strlen((char *)userName), (OraText *)userPassword,
    strlen((char *)userPassword), OCI_SPC_NO_RLB | OCI_SPC_HOMOGENEOUS);

  // 接続クラス名を設定するOCIAttrSetメソッド
  OCIAttrSet(authInfop, OCI_HTYPE_AUTHINFO, (dvoid *)connectionClassName, (ub4)
    strlen((char *)connectionClassName), OCI_ATTR_CONNECTION_CLASS, errhp);

  // OCISessionGetモードのメソッド
  OCISessionGet (envhp, errhp, &svchp, authInfop, poolName, poolNameLen, NULL, 0,
    NULL, NULL, NULL, OCI_SESSGET_SPOOL);

  /* 以下にDB問合せコードを追加します... */

  // セッション・プールを破棄してハンドルを解放します
  OCISessionPoolDestroy(spoolhp, errhp, OCI_DEFAULT);
  OCIHandleFree((dvoid *)spoolhp, OCI_HTYPE_SPOOL);
  OCIHandleFree((dvoid *)authInfop, OCI_HTYPE_AUTHINFO);
  OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
  OCIHandleFree((dvoid *)envhp, OCI_HTYPE_ENV);
}
```

接続の純度値を設定するには、*OCIAttrSet()*関数を使用します。

```
ub4 purity = OCI_ATTR_PURITY_NEW;

OCIAttrSet(authInfop, OCI_HTYPE_AUTHINFO, &purity, (ub4)sizeof(purity),
 OCI_ATTR_PURITY, errhp);
```

Oracle Call C++ Interface（OCCI）でのDRCP

Oracle Call C++ Interface（OCCI)は、Oracle Databaseにアクセスして操作するためのC++ APIを提供します。現在、OCCIライブラリはOracle Call Interface（OCI）ライブラリの上に構築されています。したがって、DRCP、接続文字列、アプリケーション・プールに関連する基本的な動作はOCIと同じままです。

DRCPの場合、最適なパフォーマンスを実現するには、OCCI環境ハンドル・クラスのOCCIセッション・プール・メソッド*createStatelessConnectionPool*を使用する必要があります。このメソッドは、*StatelessConnectionPool*オブジェクトを作成します。

一般的なOCCIアプリケーションは、*createEnvironment*静的メソッドを使用して、セッション・プールの環境ハンドルを初期化します。環境ハンドルは、その*createStatelessConnectionPool*メソッドを呼び出して、*maxConn*、*minConn*、*incrConn*などの特定のプール属性とpoolType（*HOMEOGENEOUS*または*HETEROGENEOUS*）を持つ*StatelessConnectionPool*オブジェクトを作成します。*HETEROGENEOUS*は、OCCIでのデフォルトのプール・タイプです。

アプリケーションがStatelessConnectionPoolオブジェクトから新しい接続を取得するには、渡された接続クラスと純度のパラメータで（*Connection::SELF*を使用して）*getConnection*メソッドを呼び出し、必要なDRCP設定が指定された新しい接続オブジェクトを取得します。接続オブジェクトを使用すると、文オブジェクトと結果セットを使用してデータベース文を実行できます。

最後に、アプリケーションは、接続を解放してプールに戻し（*releaseConnection*）、プールを閉じ（*terminateStatelessConnectionPool*）、環境ハンドルを閉じる（*terminateEnvironment*）必要があります。

次のコード・スニペットは、接続プーリングOCCIアプリケーションが上記のようにDRCPセッションを設定する方法を示しています。

```
#include <iostream>
#include <occi.h>
using namespace oracle::occi;
using namespace std;


#include <stdlib.h>


main(int argc, char *argv[])
{
  int i=0;
  if (argc !=2) {cout << "Usage: con <ntimes>\n"; exit(1);}
  Environment *env = Environment::createEnvironment();
  try
  {
    StatelessConnectionPool *scp = env->createStatelessConnectionPool( "
        db_user","db_pwd"," localhost:1522/orclpdb.domain.com:pooled",
        10,0,1,StatelessConnectionPool::HOMOGENEOUS);


    Connection *conn;
    Statement *stmt;
    ResultSet *rs;
```

```
    for (i=1;i<=atoi(argv[1]);i++) // これは通常はスレッド関数です
    {
      string my_tname;


      // DRCPの接続クラスと純度のパラメータを設定します
      // "ABCAPP"はここでは接続クラスです
      conn = scp->getConnection("ABCAPP",Connection::SELF);


      // DB文を実行します
      stmt = conn->createStatement("select tname from tab where rownum < 5");
      rs = stmt->executeQuery();
      while (rs->next())
      {
        // 行データを出力します
        my_tname = rs->getString(1);
        cout << my_tname << endl;
      }
      stmt->closeResultSet(rs);
      conn->terminateStatement(stmt);


      // 接続を解放して接続プールに戻します
      scp->releaseConnection(conn);
    } // ループの終わり


    // 接続プールを閉じて削除します
    env->terminateStatelessConnectionPool(scp);
  }
  catch (SQLException e)
  {
    cout << e.what() << endl;
  }

  Environment::terminateEnvironment(env);

};
```

ODP.NETでのDRCP

ODP.NET[5]には、ODP.NET Core、マネージドODP.NET、およびアンマネージドODP.NETという3つのドライバ・タイプがあります。アンマネージドODP.NETは、OCI（Oracle Call Interface）ライブラリを使用し、.NET Framework上で実行されます。マネージドODP.NETおよびODP.NET Coreには、Oracle Databaseと直接連携し、それぞれ.NET Frameworkおよび.NET Core上で実行される、100 %マネージド・コードが含まれています。

マネージドODP.NET/ODP.NET Coreドライバ・コードを使用したサンプル・コードを次に示します。

```
// このアプリケーションは次のネットワーク接続記述子を使用します
// oracle =
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=<port>))(CONNECT_DATA=
(SERVICE_NAME=<service name>)(SERVER=POOLED)))


using System;
using Oracle.ManagedDataAccess.Client;


class DRCP
{
    static void Main()
      {
            string dbconfig = "user id=hr;password=hr;data source=oracle";
            OracleConnection con = new OracleConnection(dbconfig);
            con.DRCPConnectionClass = "GroupA";
            con.Open();
            con.Dispose();
      }

}
```

*dbconfig*変数に保存されたデータベース構成は、ODP.NETでデータベース接続を確立するために使用されます。*dbconfig*変数には、ユーザー名（*user*属性）、パスワード（*password*属性）、およびネットワーク接続記述子文字列（*data source*属性）が含まれます。

また、（DRCPを有効にして）*data source*属性を指定するために、Easy Connect構文を次のように使用することもできます。

```
data source=//<hostname>:<port>/<service_name>:pooled
```

Easy Connect構文を使用した、DRCP接続の有効なデータベース構成（*dbconfig*）は、次のようになります。

```
"user id=hr;password=hr;data source=//localhost:1522/orclpdb.us.acme.com:pooled"
```

アンマネージドODP.NETでDRCPを有効にするには、次のような追加の構成が必要です。

- ODP.NET構成ファイルの設定*CPVersion*を2.0に設定します。または
- *CPVersion*構成オプションが設定されていない場合は、アプリケーションによって使用されるネットワーク接続記述子文字列に*(SERVER=POOLED)*を含めるか、Easy Connect文字列に*':pooled'*を含めます。

[5] Oracle Data Provider for .NET - オラクルによる、Oracle Database用のADO .NETデータ・プロバイダの実装

ODP.NETの接続クラス・プロパティとセッション純正値プロパティの設定

DRCP接続を複数のODP.NET接続プール間で共有するには、ODP.NET接続を開く前に、*OracleConnection.DRCPConnectionClass*プロパティを文字列値に設定します。このプロパティは、その接続の接続クラスを設定します。ODP.NETは、最初に、DRCP接続クラスのプロパティ値が同じであるアイドル接続を取得しようとします。それが見つからない場合は、新しい接続を確立します。

たとえば、前述のODP.NETコード・サンプルにある次の行では、接続クラスを設定しています。

```
con.DRCPConnectionClass = "GroupA";
```

このプロパティは、接続クラス名の設定と取得に使用できます。その値は、各アプリケーション接続プールで一意です。このプロパティのデフォルト値は*NULL*です。文字数制限は、1024からユーザーIDの文字数を引いた数となります。このプロパティを使用する場合は、接続を開く前に設定する必要があります。

ODP.NETでのデフォルトの純度値はSELF（ODP.NETの用語では*Pooled*）であり、これが推奨値です。

DRCP純度属性をNEWに設定するには、次のようにして、前述のODP.NETコード・サンプルにある*OracleConnection.DRCPPurity*プロパティを使用します。

```
con.DRCPPurity = = OracleConnection.OracleDRCPPurity.New;
```

PHPでのDRCP

PHP用のOCI8拡張は、Oracle Clientライブラリのバージョン9.2以降で使用できます。ただし、DRCP機能は、PHPがOracle 11gのクライアント・ライブラリとリンクされ、Oracle Database 11gに接続されている場合にのみ使用できます。

インストールしたら、PHPの*phpinfo()*関数を使用して、OCI8がロードされたことを確認します。

DRCPを使用する前に、新しい*php.ini*パラメータ*oci8.connection_class*を設定して、PHPアプリケーションによるプール・サーバーへのすべてのリクエストで使用される接続クラスを指定する必要があります。

```
oci8.connection_class = MYPHPAPP
```

このパラメータは、*php.ini*、*.htaccess*、または*httpd.conf*の各ファイルで設定できます。PHP関数の*ini_set()*および*ini_get()*を使用して、プログラム的にこのパラメータを設定したり取得したりすることもできます。

DRCPを使用するためのPHPアプリケーション・デプロイメント

PHPアプリケーションがDRCPを使用するには、接続文字列でサーバー・タイプPOOLEDを指定する必要があります。オラクルのEasy Connect構文を使用すると、*myhost*上の販売データベースに接続するためのPHP呼出しは、次のようになります。

```
$c = oci_pconnect('myuser', 'mypassword', 'myhost/sales:pooled');
```

PHPがOracleネットワーク・エイリアスを使用する場合は、次のようになります。

```
$c = oci_pconnect('myuser', 'mypassword', 'salespool');
```

次に、Oracleネットワーク構成ファイル*tnsnames.ora*のみを次のように変更する必要があります。

```
salespool=(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=myhost.domain.com)(PORT=1521))
```

```
(CONNECT_DATA= (SERVICE_NAME=sales)(SERVER=POOLED)))
```

*oci_close($c)*は、スクリプトの最後に実行される暗黙的な接続終了まで待たず、データベース作業の完了後すぐに呼び出すことをお勧めします。

注：*oci_pconnect()*はSELF純度を使用しますが、PHPにおけるその他の接続関数、*oci_connect()*および*oci_new_connect()*はNEW純度を使用します。

DRCPに関してよくある質問

Q1：接続ブローカの数を確認および調整するにはどうすればよいですか。

接続ブローカの数は、次のSQL文を使用して確認および調整することができます。

```
SQL> select num_cbrok from DBA_CPOOL_INFO;

NUM_CBROK
----------
         1
```

接続ブローカを設定するには、表-3の説明に従って、NUM_CBROK DRCP構成オプションを設定してください。

```
SQL> select num_cbrok from DBA_CPOOL_INFO;

NUM_CBROK

----------
         2
```

同時接続リクエストの数が多い場合は、単一の接続ブローカが過負荷になる可能性があるため（ブローカのCPU使用率が高くなることで示されます）、接続ブローカの数を増やすことをお勧めします。

Q2：複数の接続ブローカがある場合、接続ブローカ・プロセス間の接続分散負荷を確認できる方法はありますか。

接続分散負荷に関しては、次のようにしてv$cpool_conn_infoで確認できます。

```
SQL> select cmon_addr, count(*) from v$cpool_conn_info group by cmon_addr;

CMON ADDR           COUNT(*)
----------------- -----------
000000014BE63E40         500
000000014BE64198         500
```

上記の例では、各接続ブローカ・プロセスに対して500のクライアント接続が確立されます。合計1,000の接続がDRCPに対して行われます。

Q3：アプリケーションから接続ブローカへの接続があるときに*DRCP*プールを停止できますか。

Oracle Database 23aiでは、*dbms_connection_pool.stop_pool()*にDRAINTIMEパラメータが導入されています。これは、すべてのプール・サーバーをすぐに強制終了してプールを停止するように構成できます。

これより前のリリースでは、クライアントからブローカへの接続があるときにプールを停止することはできません。

Q4：接続ブローカの数を変更するために、DRCPプールを再起動する必要がありますか。

ブローカの数を増やす場合、プールを再起動する必要はありません。ブローカの数を減らす場合は、クライアント/アプリケーションが切断されるまで待機してください。

接続ブローカは、接続のチェックイン/チェックアウトを管理し、ブローカが複数ある場合は負荷を共有させ、CPU使用率が増加しないようにします。接続ブローカ・プロセスは、クライアントが接続するときか、プール・サーバーをアクティブにリクエストして解放するときにのみCPUを使用します。

Q5：接続ブローカの数の制限とは何ですか。同時にいくつの接続リクエストを処理できますか。

DRCP接続ブローカの数はワークロードに完全に依存するため、その数にハードコードされた制限はありません。オラクルの自動ワークロード・リポジトリ（AWR）レポートは、接続ブローカの負荷に関する洞察を提供します。これが、接続ブローカ・プロセスの数を増やす必要があるかどうかを示している場合があります。

Q6：DRCPはTCPS接続をサポートしていますか。

いいえ。本書の公開時点では、TCPS接続はサポートされていません。

Q7：DRCPのMAX_AUTH_SERVERSパラメータの値を大きくした後に認証サーバー（num_auth）セッションの数を検証する（SQL問合せによる認証のメリットの向上を追跡する）にはどうすればよいですか。

十分な数の認証サーバーがある場合、ネットワーク（TNS）エラーは表示されないはずです。

存在する認証サーバーの数を確認したい場合はいつでも、ルートDBAユーザーとしてOracle Database 19cで問合せを実行できます。

```
SQL> select kmpcpname as pool, kmpcpnsrv as num_srvs, kmpcpbsrv as num_busy,
kmpcpfsrv as num_free, (kmpcpawait+kmpcpswait) as num_waiters from x$kmpcp where
kmpcpstate != 0;


POOL NUM_SRVS NUM_BUSY NUM_FREE NUM_WAITERS
------------------------------ ---------- ---------- ---------- ------------

SYS_AUTH_POOL 1 0 1 0
SYS_DEFAULT_CONNECTION_POOL 4 1 3 0
```

Oracle Database 21c以降のDBリリースでは、同じ目的でV$AUTHPOOL_STATSビューが提供されます。

Q8：DRCPで許可される認証サーバー・プロセスの数を増やすにはどうすればよいですか。

認証サーバーの数は、データベース初期化パラメータMAX_AUTH_SERVERSによって制限されます。このパラメータのデフォルト値は40です。

DRCPで認証サーバー・プロセスの数を増やすには、次のようにして、必要な権限を持つユーザー（場合によってはルートDBAまたはPDB管理者ユーザー）としてデータベースのMAX_AUTH_SERVERS値を大きくする必要があります。

```
SQL> ALTER SYSTEM SET MAX_AUTH_SERVERS = 100
```

このパラメータに設定できる値に、ハードコードされた制限はありません。

Q9：接続ブローカと認証サーバーの数が多い場合、インフラストラクチャのどの部分が影響を受けますか。

データベース・ホストのCPUは、接続ブローカと認証サーバーの数の増加によって影響を受けます。

まとめ

DRCPにより、アプリケーションは、複数のアプリケーション・サーバー間と中間層デプロイメント間で共有される、データベース内の接続プールを使用できます。これらのアプリケーションがDRCPを効果的に使用するためには、セッションを取得または解放する呼出しでデータベース・アクティビティをアクティブにラップする必要があります。このようなアプリケーションは接続を迅速に確立でき、多数の接続に対して最小限のデータベース・リソースを使用します。

オラクルのデータベース・プロキシ・ソリューションである、Traffic DirectorモードのOracle Connection Manager（CMAN-TDM）には、DRCPと同じように機能する独自のプーリング機能であるプロキシ常駐接続プーリング（PRCP）があります。あるアプリケーションがDRCPで適切に機能する場合、そのアプリケーションはPRCPでも同様に機能します。PRCPに（アプリケーション側で）必要な唯一の変更は、tnsnames.oraエイリアスまたはEasy Connect文字列がデータベース/DRCPサーバーではなくPRCPサーバーを指す必要があることです。

まとめると、DRCPのメリットは次のとおりです。

- DRCPによって、複数のクライアント・アプリケーションとアプリケーション・サーバー間でリソースを共有できるようになります。
- DRCPは、データベース・ホストのリソース使用率を削減することにより、データベースとアプリケーションのスケーラビリティを向上させます。

詳細情報

詳しくは、次のリンクとドキュメントを参照してください。

- DRCPについて：Oracle Database管理者ガイド
- データベース常駐接続プールの使用：Oracle Database開発者ガイド
- データベース常駐接続プーリング：Oracle Database概要
- データベース常駐接続プーリング：Oracle Call Interfaceプログラマーズ・ガイド
- データベース常駐接続プーリング：Oracle Data Provider for .NET開発者ガイド
- データベース常駐接続プーリング：Oracle JDBC開発者ガイド
- DRCPの新規パラメータ：Oracle Database新機能ガイド
- CMAN-TDM - スケーラブルな高可用性アプリケーションのためのOracle Database接続プロキシ：CMAN-TDM技術概要
- Oracle DatabaseのEasy Connect Plus：Easy ConnectおよびEasy Connect Plusの技術概要
- プーリングとキャッシュを使用したアプリケーション・プログラミング：Oracle Databaseリソースのプーリングとキャッシュに関する技術概要
- ゴル航空はOracle Cloud Infrastructureを使用してチケット購入を60秒以内に追跡：オラクルのお客様事例
- 複数プールDRCPに関するブログ：「Multi-pool Database Resident Connection Pooling（DRCP）in Oracle Database 23ai」
- 暗黙的な接続に関するブログ：「Implicit Connection Pooling when connections overload your database」

Connect with us

+1.800.ORACLE1までご連絡いただくか、**oracle.com**をご覧ください。北米以外の地域では、**oracle.com/contact**で最寄りの営業所をご確認いただけます。

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

ORACLE