Oracle® Database
Oracle Sharding の使用
19c

F16095-08(原本部品番号:E87088-11)

2023年2月

タイトルおよび著作権情報

Oracle Database Oracle Shardingの使用 19c

F16095-08

原本著者: Virginia Beecher、Roopam Jain

原本協力者: Shailesh Dwivedi、Jean- Francois Verrier、Prakash Jashnani、Pankaj Chandiramani、Mark Dilman、Nourdine Benadjaoud、Rennie Sreekumar Ranjit Kumar、David Colello、Steve Ball、Abhishek Srivastava、Sebastian Binek、Shahab Hamid

目次

はじめに

次のことを行うには、次のトピックを確認してください。

- Oracle Shardingについて学習するために、このドキュメントの使用方法を調べます
- このドキュメントのアクセシビリティ情報を参照します
- Oracle Sharding環境の設計、開発、デプロイおよび管理に役立つ関連ドキュメントのリストを参照します
- このドキュメントで使用されている表記規則について学習します。

対象読者

このドキュメントは、対象読者を広範に想定して記述されています。システム・アーキテクトとアプリケーション・アーキテクトは、目的の要件に対するOracle Shardingの適性を評価するために使用できます。ITマネージャは、概念実証と本番環境のデプロイに向けてOracle Shardingを実装するために必要な作業を精査できます。データベース管理者は、シャード・データベースのデプロイとメンテナンスに役立つ情報を調べられます。アプリケーション開発者は、Oracle Shardingを使用する場合のコード変更について学習できます。最後に、ビジネス・アナリストは、このドキュメントをガイドとして使用して、Oracle Shardingの実装についての費用を算定できます。

ドキュメントのアクセシビリティについて

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracleサポートへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

詳細は、Oracle Databaseのドキュメント・セットを参照してください。特に関連性の高いドキュメントを次に示します。

- Oracle Database Global Data Services概要および管理ガイド
- Oracle Database管理者ガイド
- Oracle Data Guard概要および管理
- Oracle Data Guard Broker
- Oracle Fusion Middleware Oracle GoldenGate Microservices Architectureの使用
- Oracle Database JDBC開発者ガイド
- Oracle Universal Connection Pool開発者ガイド
- Oracle Data Provider for .NETの開発者ガイドfor Microsoft Windows

- Oracle Call Interfaceプログラマーズ・ガイド

表記規則

このドキュメントでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダー変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

Oracle Shardingのこのリリースの変更点

「はじめに」の内容は次のとおりです。

Oracle Database 19cでの変更点

次に、Oracle Database 19cのOracle Shardingの変更点を示します。

新機能

このリリースの新機能は次のとおりです。

システム管理のシャーディングのための複数の表ファミリのサポート

Oracle Database 18cのOracle Sharding機能では、シャード・データベースごとに1つの表ファミリ(同じシャーディング・キーを共有する関連表のセット)のみがサポートされていました。Oracle Database 19cのOracle Shardingでは、異なる表ファミリのすべてのデータが同じチャンクに存在する、複数の表ファミリをサポートしています。この機能は、システム管理のシャード・データベースにのみ適用されます。異なる表ファミリにアクセスする異なるアプリケーションが、1つのシャード・データベースでホストできるようになりました。

1つの新しいGDSCTLコマンドCONFIG TABLE FAMILYがあり、この機能をサポートするために、ADD SERVICE、MODIFY SERVICE、CONFIG SERVICE、CONFIG CHUNKS、STATUS ROUTING、VALIDATE CATALOGといったその他のコマンドが拡張されています。

この機能で導入された新しいSQLキーワードまたは文はありませんが、CREATE SHARDED TABLEおよびTABLESPACE SETの使用について、一部の制限事項が変更されています。

関連項目

- シャード表ファミリ
- Oracle Database Global Data Services概念および管理ガイド
- Oracle Database SQL言語リファレンス

同一のCDBでの複数のPDBシャードのサポート

Oracle Database 18cのOracle Shardingでは、CDBで1つのPDBをシャードまたはシャード・カタログ・データベースとして使用する機能が導入されました。Oracle Database 19cのOracle Shardingでは、CDBで複数のPDBをシャードまたはシャード・カタログ・データベースとして使用できますが、特定の制限事項があります。たとえば、この機能によって、CDBには異なるシャード・データベース(SDB)からのシャードPDBを含めることができ、それぞれに個別のカタログ・データベースがあります。

シャードPDBを19cに移行する方法の詳細は、Oracle Database 18cからの互換性と移行を参照してください。

シャード間の一意の順序番号の生成

Oracle Database 19cより前では、シャード間で一意の番号が必要な場合、それを自分で管理する必要がありました。

Oracle Database 19cでは、Oracle Shardingを使用して、シャードごとに順序番号を個別に生成できます。順序番号はすべてのシャード間で一意です。

この機能をサポートするために、新しいSEQUENCEオブジェクト句、SHARDおよびNOSHARDがSEQUENCEオブジェクトのDDL構文に含まれます。

関連項目

- シャード間の一意の順序番号の生成
- Oracle Database SQL言語リファレンス

シャード・カタログ・スタンバイでのマルチシャード問合せコーディネータのサポート

Oracle Database 19cより前は、マルチシャード問合せコーディネータとして使用できるのはプライマリ・シャード・カタログ・データベースのみでした。Oracle Database 19cでは、シャード・カタログ・データベースのOracle Active Data Guardスタンバイでマルチシャード問合せコーディネータを有効にすることもできます。これにより、マルチシャード問合せワークロードのスケーラビリティおよび可用性が向上します。

関連項目

- 問合せ処理と問合せコーディネータ

シャード間のパラメータ設定の伝播

Oracle Database 19cより前は、データベース管理者はシャード・データベースの各シャードでALTER SYSTEMパラメータ設定を構成する必要がありました。この機能により、管理者はパラメータ設定を集中的に管理し、シャード・カタログからすべてのデータベース・シャードに伝播できるため、管理性が向上します。シャード・カタログで設定を構成すると、シャード・データベースのすべてのシャードに自動的に伝播されます。

シャード間のパラメータ設定の伝播を参照してください。

非推奨およびサポート終了

このリリースで非推奨となった機能またはサポートされなくなった機能は次のとおりです。

GDSCTLコマンドラインでのパスワードの設定のサポート終了

セキュリティを強化するために、Oracle Database 19c以降、オペレーティング・システム・プロンプトからコールされたときにGlobal Data Services制御ユーティリティ(GDSCTL)のコマンドラインからパスワードを指定する機能はサポートされなくなりました。

このサポート終了は、ユーザー・コマンドライン・プロンプトからGDSCTLがコールされた場合のパスワード変更にのみ適用されます。たとえば、次のコマンドはサポートされなくなりました。

```
$ gdsctl add database -connect inst1 -pwd gsm_password
```

GDSCTLユーティリティ自体からのパスワードの指定は、引き続き有効です。たとえば、次のコマンドは有効です。

```
GDSCTL> add database -connect inst1 -pwd gsm_password
```

この非推奨では、オペレーティング・システム・プロンプトからコールされるGDSCTLコマンドでパスワードを指定する際のセキュリティの脆弱性に対処します。

1 Oracle Shardingの概要

ここでは、Oracle Shardingで実行できることについて大まかな概念について学習します。

次の項目では、Oracle Shardingの機能および利点について説明します。

シャーディングとは

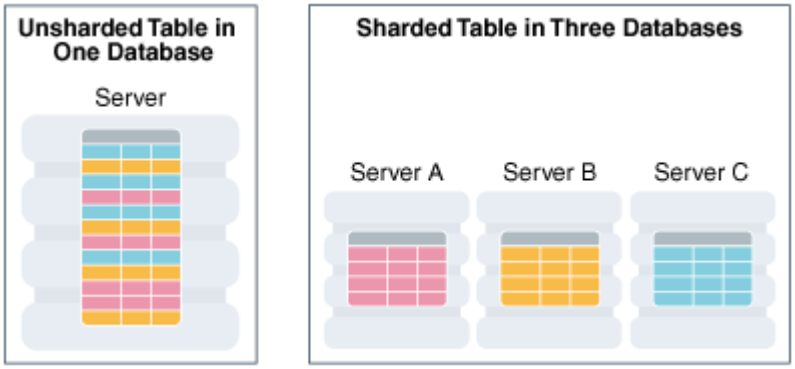
ハイパースケール・コンピューティングは、システムに対する需要の増加にあわせて素早くスケール・アップまたはスケール・ダウンできるコンピューティング・アーキテクチャです。このアーキテクチャの革新は、分散サイトを運営しているインターネット巨大企業によって主導されたもので、大規模なクラウド・プロバイダに採用されています。

多くの場合、企業はデータベース・シャーディングと呼ばれるテクノロジを使用してハイパースケール・コンピューティングを実現します。このテクノロジでは、データ・セットのセグメントが分散され、シャードは多数の異なるコンピュータに多数のデータベースを分散します。

シャーディングには、何も共有しないアーキテクチャを使用します。このアーキテクチャでは、シャードはハードウェアもソフトウェアも共有しません。すべてのシャードが一体になって、シャード・データベースと呼ばれる単一の論理データベースを形成します。

アプリケーションの観点からは、シャード・データベースは単一のデータベースのように見えます。シャードの数とそれらのシャード間でのデータの分散は、データベース・アプリケーションに対して完全に透過的です。データベース管理者の立場では、シャード・データベースは、まとめて管理できる複数のデータベースで構成されています。

図1-1 データベース・シャード間の表の分散



Oracle Shardingについて

Oracle Shardingは、ハードウェアまたはソフトウェアを共有しないOracleデータベースのプール間でデータを自動的に分散およびレプリケートできる、Oracle Databaseの機能です。Oracle Shardingは、ここで説明するように、成熟したRDBMSおよびNoSQLデータベースの最高の機能と性能を備えています。

- オブジェクトの作成、厳格なデータ整合性、複雑な結合、ACIDトランザクションのプロパティ、分散トランザクション、リレーショナル・データ・ストア、セキュリティ、暗号化、堅牢なパフォーマンス・オプティマイザ、バックアップとリカバリ、およびOracle Databaseでのパッチ適用に使用されるSQL言語

- Oracleのイノベーションとエンタープライズ・レベルの機能: Advanced Security、Automatic Storage Management (ASM)、Advanced Compression、パーティション化、高性能ストレージ・エンジン、SMPスケーラビリティ、Oracle RAC、Exadata、インメモリー列、オンライン再定義、JSONドキュメント・ストアなど
- シャーディング対応のOracle Databaseツール: シャード・データベース・アプリケーションの開発と管理のためのSQL Developer、Enterprise Manager Cloud Control、Recovery Manager (RMAN)、およびData Pump
- プログラム・インタフェース: シャード・アプリケーション開発用の拡張機能を含むJava Database Connectivity (JDBC)、Oracle Call Interface (OCI)、Universal Connection Pool (UCP)、Oracle Data Provider for .NET (ODP.NET)、PL/SQLなど
- Oracle Data GuardおよびActive Data Guardによる究極の可用性。

  ノート:

  Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。
- 複数モデルのデータ(リレーショナル、テキスト、JSONなど)のサポート
- 社内およびワールド・ワイドのOracleデータベース管理者スキル・セットの活用して、既存のライフサイクル管理および運用プロセスの維持が可能
- エンタープライズ・レベルのサポート
- NoSQLデータベースの最大限のスケーラビリティと可用性

分散パーティション化としてのOracle Sharding

シャーディングは、データを複数の独立した物理データベース間に水平にパーティション化するデータベースのスケール・テクニックです。このような構成内の各物理データベースをシャードと呼びます。

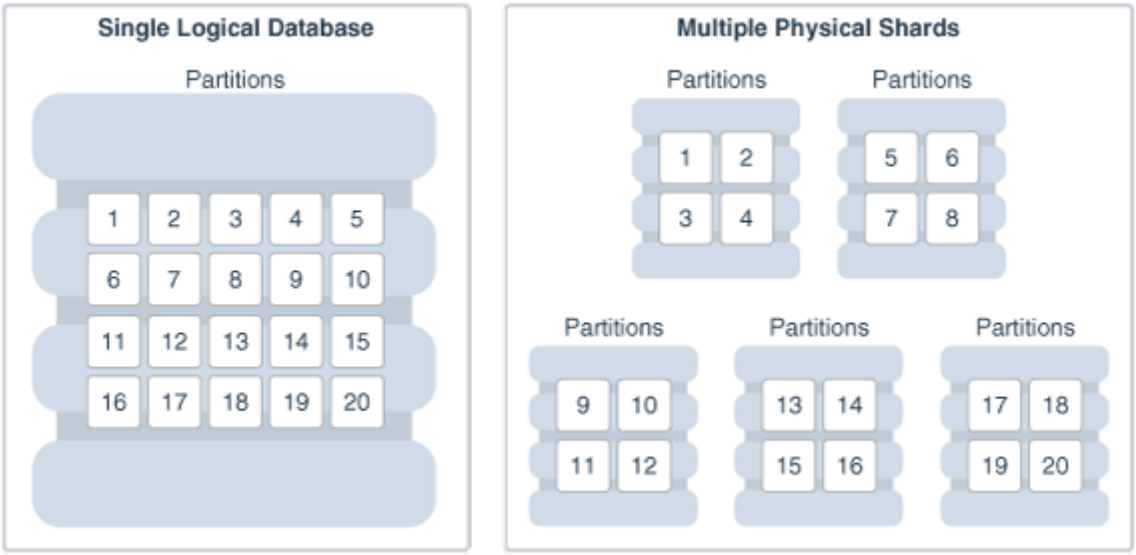
アプリケーションの視点からすると、Oracle Shardingのシャード・データベースは単一のデータベースのように見えます。シャードの数とそのシャード間におけるデータ分散は、アプリケーションに対して完全に透過的です。

シャード・データベースはアプリケーションおよびアプリケーション開発者には単一のデータベースのように見えますが、データベース管理者の視点では、シャード・データベースは、それぞれが単一のシャードであり、一括して管理できる一連の別個のOracleデータベースで構成されています。

シャード表は、シャード・データベースのすべてのシャード間にパーティション化されます。各シャードの表のパーティションは、シャーディングされていないOracleデータベースで使用できるパーティションと同様です。

次の図は、単一の論理データベースのパーティション化と複数シャード間に分散されたパーティションの違いを示しています。

図1-2 分散パーティション化としてのシャーディング



CREATE SHARDED TABLE文を実行すると、Oracle Shardingは自動的にシャード間にパーティションを分散します。パーティションの分散はアプリケーションに対して透過的です。前述の図は、シャード表の論理ビューとその物理的な実装を示しています。

Oracle Shardingの利点

Oracle Shardingは、最も負荷の高いアプリケーションに対する線形の拡張性を備え、完全な障害分離およびグローバルなデータ分散を実現します。

Oracle Shardingには、次のような主な利点があります。

- 線形のスケーラビリティ

  Oracle Shardingの何も共有しないアーキテクチャにより、パフォーマンスのボトルネックを排除して、制限のないスケーラビリティが得られます。Oracle Shardingでは、1,000シャードまでのスケーリングがサポートされています。

- 最大限の可用性と障害の分離

  シャードはソフトウェア、CPU、メモリー、ストレージ・デバイスなどのリソースを共有しないため、単一障害点が排除されます。あるシャードの障害や処理速度の低下が別のシャードのパフォーマンスや可用性に影響しません。

  シャードは、Oracle Data GuardやOracle RACなどのOracle MAAベスト・プラクティス・ソリューションによって保護されます。

  シャードの計画外停止または計画メンテナンスはそのシャードのデータの可用性にしか影響しないため、フェイルオーバー・ブラウンアウト時などにそのごく一部のデータのユーザーのみが影響を受けます。

- データの地理的分散

  シャーディングにより、単一の論理データベースが複数の地理的位置に分散されるグローバル・データベースを使用できます。これにより、データ・プライバシ規則の要件(データ主権)を満たせるようになり、特定のデータを利用者の近くに保存できるようにもなります(データ近接性)。

データベース・シャーディングを使用するアプリケーションの例

Oracle Shardingは、様々なユース・ケースでメリットが得られます。

リアル・タイムOLTP

リアル・タイムOLTPアプリケーションは、非常に高いトランザクション処理スループット、大量のユーザー人口、膨大なデータ量を扱い、大規模に厳格なデータ整合性と管理を必要とします。たとえば、インターネットに接続する消費者アプリケーション、金融アプリケーション(モバイル決済など)、大規模SaaSアプリケーション(課金や診察アプリケーションなど)が挙げられます。こうしたアプリケーションにOracle Shardingを使用すると、次のようなメリットが得られます。

- 1秒当たりのトランザクションの線形スケーラビリティ。増大したデータ量をサポートするために新しいシャードを追加したときにも応報時間が一定に保たれます
- アプリケーションSLAの向上。特定のシャードに対する計画停止および計画外停止は、別のシャードに保存されたデータに影響することなく利用できます
- トランザクション・アプリケーションに対応する厳格なデータ整合性
- 複数のシャードに及ぶトランザクション
- 複雑な結合、トリガー、およびストアド・プロシージャのサポート
- 大規模な管理の簡易化

グローバル・アプリケーション

多くのエンタープライズ・アプリケーションは本質的にグローバルなもので、同じアプリケーションが複数の地理的位置にいる顧客に使用されます。一般に、こうしたアプリケーションは、複数の地域で共有される単一の論理グローバル・データベースを使用します。共有グローバル・データベースには、次のようなメリットがあります。

- データ主権の厳格な施行。データ・プライバシ規制により、特定の地理的場所、地域、国、または地方にデータを保留することが必要になる場合があります。
- 各場所に渡るデータ・レプリケーションの削減
- アプリケーションSLAの向上。ある地域での計画停止および計画外停止は別の地域に影響しません

IoT (Internet of Things)とデータ・ストリーミング・アプリケーション

一般に、こうしたアプリケーションは、大量のデータを収集して、高速でストリーミングします。Oracle Shardingには、最適化されたデータ・ストリーム・ライブラリが用意されています。このライブラリでは、Oracle Databaseダイレクト・パスI/Oテクノロジを使用して、シャード・データベースにデータを高速にロードします。このようなアプリケーションのデータ・ロードの要件は、1秒当たりのレコード数が数億になることもあります。データがデータベースに直接ロードされると、高度な問合せ処理と分析機能によってすぐに処理できるようになります。

機械学習

機械学習アプリケーションの多くは、リアルタイムでのモデルのトレーニングおよびスコアリングを必要とします。異常検出などのアルゴリズムを使用するアプリケーションに対応するモデルのトレーニングとスコアリングや、クラスタリングは、特定のエンティティに固有のものです(たとえば、一定の日数における特定のユーザーの金融取引パターンやデバイスのメトリックなど)。この種のデータは、ユーザーやデバイスに固有のシャーディング・キーを使用することで簡単に共有できます。さらに、Oracle Database Machine Learningのアルゴリズムは、データベースで直接適用できるため、個別のデータ・パイプラインや機械学習処理インフラストラクチャが不要になります。

ビッグ・データの分析

TB単位のデータがあるときには、シャーディングによって、分析のためにデータをウェアハウスに格納するが必要なくなります。Oracle Shardingには最大1000シャードの収容力があるため、リレーショナル・データベースをウェアハウス規模のデータ・スト

アに転用できます。フェデレーテッド・シャーディング・ソリューションを使用すると、同じアプリケーションを実行する様々な場所にある複数のデータベース・インストールをフェデレーテッド・シャード・データベースに変換できるため、データを移動せずにデータ分析を実行できます。

NoSQLの代替

NoSQLソリューションには、リレーショナル・スキーマ、SQL、複合データ型、オンラインのスキーマ変更、マルチコアのスケーラビリティ、セキュリティ、ACIDプロパティ、単一シャード操作のためのCRなど、主要なRDBMS機能がありません。Oracle Shardingでは、NoSQLで得られるほとんど無制限のスケーリングとシャーディングに加えて、Oracle Databaseのすべての機能と利点も得られます。

柔軟なデプロイメント・モデル

Oracle Shardingの何も共有しないアーキテクチャにより、データはオンプレミス、クラウド、またはクラウドとオンプレミス・システムのハイブリッドに保持できます。データベース・シャードでは一切のリソースを共有しないため、シャードはオンプレミスおよびクラウドの様々なシステムのどこにでも存在できます。

すべてのシャードをオンプレミスにデプロイすることも、すべてをクラウドに配置することも、ニーズにあわせてクラウドとオンプレミスに分割することもできます。

シャードは、あらゆるデータベース・デプロイメント・モデル(単一インスタンス、Exadata、Oracle RACなど)にデプロイできます。

Oracle Shardingの高可用性

Oracle Shardingは、高可用性および障害時リカバリを提供するためにOracle Data Guardと緊密に統合されています。レプリケーションはシャード・データベースが作成されると自動的に構成およびデプロイされます。

Oracle Data Guardのレプリケーションは、高可用性およびデータ保護のために、シャード(プライマリ)の1つ以上の同期されたコピー(スタンバイ)を維持します。スタンバイはローカルまたはリモートにデプロイでき、Oracle Active Data Guardを使用する場合は、読取り専用アクセスでオープンすることもできます。このオプションは、アプリケーションに厳格なデータ整合性とゼロ・データ損失が必要な場合に使用してください。

Oracle GoldenGateは、ファイングレイン・アクティブ-アクティブ・レプリケーションに使用します。ただし、アプリケーションは、潜在的なフェイルオーバー発生時の競合とデータ損失に対処できる必要があります。

ノート:

Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。

オプションで、シャードレベルの高可用性のために、レプリケーションによって補完されるOracle RACを使用すると、クラスタが停止した場合にシャードレベルのデータの可用性を維持できます。各シャードは、Oracle RACクラスタにデプロイすることで、すぐにノード障害からの保護が可能です。たとえば、各シャードを2ノードのOracle RACクラスタにすることができます。

シャーディング方法

Oracle Shardingは表のパーティション化に基づくため、Oracle Databaseで提供されるサブパーティション方法はすべてOracle Shardingでもサポートされます。データ・シャーディング方法は、シャードへのデータの配置を制御します。Oracle Shardingでは、システム管理、ユーザー定義またはコンポジットの各シャーディング方法をサポートしています。

- システム管理のシャーディングでは、データをシャードにマップする必要はありません。コンシステント・ハッシュによるパーティション化を使用して、データが自動的にシャード間に分散されます。パーティション化アルゴリズムにより、データがシャード全体に、均一かつランダムに分散されます。
- ユーザー定義のシャーディングでは、個々のシャードへのデータのマッピングをユーザーが明示的に指定できます。これは、パフォーマンスや規制などの理由で、特定のデータを特定のシャードに格納する必要があり、管理者がシャード間のデータの移動を完全に制御する必要がある場合に使用します。
- コンポジット・シャーディングでは、2つのレベルのシャーディングを使用できます。まずデータが範囲またはリストによってシャードされ、次にコンシステント・ハッシュによってさらにシャードされます。

  多くの(特に、データの主権とデータの近接性の要件に関する)ユース・ケースで、コンポジット・シャーディング方法は、システム管理とユーザー定義の両方のシャーディング方法の最良点を提供し、必要な自動化とデータ配置の制御を可能にします。

クライアント・リクエストのルーティング

Oracle Shardingは、アプリケーションからシャードへの直接的なキーベースのルーティング、シャード・カタログを使用したプロキシによるルーティングおよびシャードでアフィニティ化された中間層(アプリケーション・コンテナ、Webコンテナなど)へのルーティングをサポートしています。Oracle Databaseクライアント・ドライバおよび接続プールはシャーディング対応です。

- キー・ベースのルーティング。Oracleクライアント側ドライバ(JDBC、OCI、UCP、ODP.NET)は、高パフォーマンスのデータ依存ルーティングのために接続文字列に指定されたシャーディング・キーを認識できます。接続レイヤーのシャード・ルーティング・キャッシュは、データが常駐するシャードにデータベース・リクエストを直接ルーティングするために使用されます。
- プロキシによるルーティング。Oracle Shardingは、シャーディング・キーを指定していない問合せのルーティングをサポートしています。これにより、あらゆるデータベース・アプリケーションは、問合せを実行するシャードを指定することなくSQL文を実行するという柔軟性が得られます。プロキシ・ルーティングでは、単一シャードの問合せとマルチシャードの問合せを処理できます。
- 中間層ルーティング。データ層のシャーディングに加えて、Web層とアプリケーション層のシャーディングも可能です。こうした中間層のシャードを分散することで、データベース・シャードの特定のセットを処理して、スイム・レーンというパターンを作成します。スマート・ルーターは、クライアントのリクエストを特定のシャーディング・キーに基づいて適切なスイム・レーンにルーティングできます。このスイム・レーンにより、シャードのサブセットに対する接続が確立されます。

問合せの実行

Oracle Shardingをサポートするために、問合せやDML文に変更を加える必要はありません。ほとんどの既存のDDL文は、非シャードOracle Databaseと同じ構文およびセマンティクスを使用して、シャード・データベースで同様に機能します。

構成内のすべてのシャードでDDL文を実行するのと同じ方法で、特定のOracle提供のPL/SQLプロシージャも実行できます。

また、Oracle Shardingには、シャード・データベースに対してのみ実行できるSQL DDL文の独自のキーワードがあります。

高速データ収集

SQL*Loaderにより、高速データ収集の際にはデータをデータベース・シャードに直接ロードできます。

SQL*Loaderは、外部ファイルからOracleデータベースにデータを移動する際に使用するバルク・ローダー・ユーティリティです。その構文はDB2ローダー・ユーティリティの構文と同様ですが、より多くのオプションを使用できます。SQL*Loaderは、様々なロード形式、選択的ロード、および複数表のロードをサポートしています。その他の利点は、次のとおりです。

- ストリーミング機能により、大規模なクライアントのグループからのデータをブロックなしで受信できます
- ネイティブUCPを使用するOracle RACシャード・アフィニティに従ったレコードのグループ化
- レコード処理をI/Oから切り離した上でのCPU割当ての最適化
- ダイレクト・パス・インサート(SQLをバイパスしてデータベース・ファイルに直接書き込むこと)による、Oracle Databaseの挿入メソッドの高速化

デプロイの自動化

シャード・データベースのデプロイメントは、スクリプト(Terraform、Kubernetes、およびAnsible)によって高度に自動化されます。

このデプロイメント用スクリプトは、目的のデプロイメント・トポロジについて記述したシンプルな入力ファイルを受け取って、単一のホストから実行することですべてのシャード・データベース・ホストにシャードをデプロイします。スクリプトには、エラー発生時のための一時停止操作、再開操作、およびクリーン・アップ操作が含まれています。

データ・ポンプの移行

Oracle Data Pumpはシャーディング対応で、非シャードOracleデータベースからシャードOracleデータベースにデータを移行するために使用します。

各シャードでOracle Data Pumpを実行することで、データをシャードに直接ロードできます。この方法は非常に高速です。これは、データ・ロード操作全体が、データ・セット全体の最大サブセットを使用してシャードをロードするために必要な時間内に完了できるためです。

シャードのライフサイクル管理

Oracle Shardingコマンドライン・インタフェースとOracle Enterprise Managerは、シャード・データベースの管理に役立ちます。

この提供されたツールを使用すると、次の作業を実行できます。

- スクリプトによる新しいシャード・データベースのプロビジョニング
- オンラインでのシャードの追加と自動リバランスの活用による必要に応じたスケール・アウト
- 負荷が低いときのデータの移動とハードウェアの集約によるスケール・イン
- Enterprise Managerを使用したパフォーマンス統計情報の監視

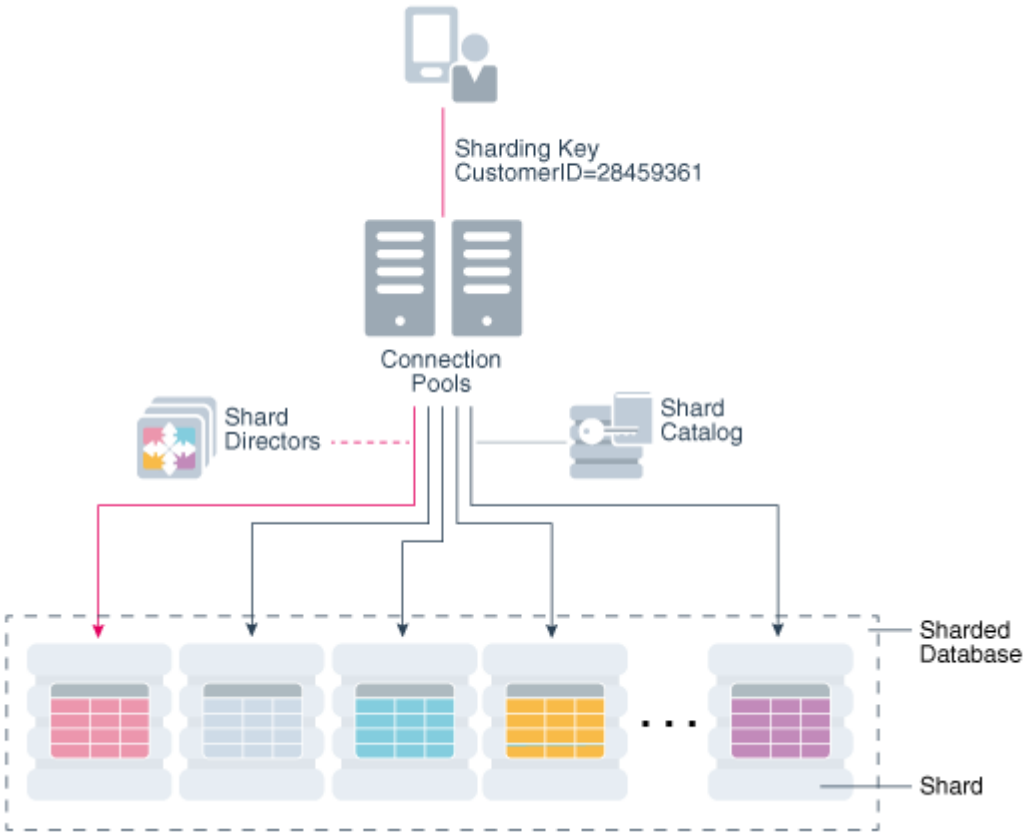
- Cloudバックアップ・サービス、RMAN、およびZero Data Loss Recovery Applianceを使用した障害回復のためのバックアップ作成
- ローリング・モードのoPatchAutoによって自動化されるパッチ適用とアップグレード

2 Oracle Shardingのアーキテクチャおよび概念

Oracle Shardingアーキテクチャのコンポーネント

次の図は、次のトピックで説明するOracle Shardingの主要なアーキテクチャ・コンポーネントを示しています。

図2-1 Oracle Shardingのアーキテクチャ



シャード・データベースとシャード

シャード・データベースは**シャード**の集まりです。

シャード・データベースとは、ハードウェアまたはソフトウェアを共有しない物理Oracleデータベース(シャード)のプールで水平にパーティション化された単一の論理Oracle Databaseです。

シャード・データベース内の各シャードは、シャード・データベースのデータのサブセットをホストする独立したOracle Databaseインスタンスです。シャード間で共有記憶域は必要ありません。

シャードは、Oracleデータベースをホストできる場所であればどこでもホストできます。Oracle Shardingは、オンプレミス、任意のクラウド・プラットフォーム、Oracle Exadata Database Machine、仮想マシンなど、単一インスタンスまたはクラスタ化されたOracle Databaseで予想されるシャードのすべてのデプロイメント選択をサポートしています。

各シャードは、1つのリージョンに配置するか、別々のリージョンに配置できます。Oracle Shardingのコンテキストでのリージョンは、1つのデータ・センターまたはネットワーク上で近接している複数のデータ・センターを表します。

シャードは、Oracle Data Guardを使用した高可用性および障害時リカバリのためにレプリケートされます。高可用性のために

は、プライマリ・シャードが配置されているリージョンと同じリージョンに、Data Guardスタンバイ・シャードを配置できます。障害時リカバリのためには、スタンバイ・シャードを別のリージョンに配置できます。

ノート:

Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。

シャード・カタログ

シャード・カタログは、自動的なシャードのデプロイメント、シャード・データベースの集中管理、およびマルチシャード問合せをサポートするOracle Databaseです。

シャード・カタログは次の目的を果たします。

- 共有データベース全体の管理サーバーとして機能します。
- データベース・スキーマのゴールド・コピーを格納します
- マルチシャード問合せコーディネータを使用したマルチシャード問合せの管理
- 重複表データのゴールド・コピーを格納します

シャード・カタログは特殊な目的のOracle Databaseであり、シャード・データベース構成データの永続的なストアとなり、シャード・データベースの集中管理で主要な役割を果たします。すべての構成の変更(シャードやグローバル・サービスの追加、削除など)は、シャード・カタログで開始されます。シャード・データベースのすべてのDDLは、シャード・カタログに接続することによって実行されます。

シャード・カタログには、シャード・データベースのすべての重複表のマスター・コピーも含まれています。シャード・カタログは、マテリアライズド・ビューを使用して、すべてのシャードの重複表に変更を自動的にレプリケートします。シャード・カタログは、マルチシャード問合せ、およびシャーディング・キーを指定しない問合せを処理するための問合せコーディネータとしても機能します。

複数のシャード・カタログを高可用性のためにデプロイできます。シャード・カタログの高可用性のためにOracle Data Guardを使用することをベスト・プラクティスとしてお薦めします。

実行時に、アプリケーションがキーベースの問合せを使用しないかぎり、シャードに問合せを送信するにはシャード・カタログが必要です。シャーディング・キーベースのトランザクションは引き続きシャード・データベースにルーティングされて実行され、カタログの停止の影響を受けません。

スタンバイ・シャード・カタログへの自動フェイルオーバーを完了するために必要な短期の停止時間は、メンテナンス操作の実行、スキーマの変更、重複表の更新、マルチシャード問合せの実行、トポロジの変更を引き起こすシャードの追加やチャンクの移動などのその他の操作の実行に影響します。

シャード・ディレクタ

シャード・ディレクタは、シャーディング・キーに基づいて高パフォーマンスな接続ルーティングを可能にするネットワーク・リスナーです。

Oracle Database 12cでは、グローバル・サービス・マネージャが導入され、データベース・ロール、負荷、レプリケーション・ラグ

およびローカル性に基づいて接続がルーティングされます。Oracle Shardingをサポートするために、グローバル・サービス・マネージャはデータの場所に基づいた接続のルーティングをサポートします。グローバル・サービス・マネージャは、Oracle Shardingのコンテキストではシャード・ディレクタと呼ばれます。

シャード・ディレクタは、グローバル・サービス・マネージャの特殊な実装であり、シャード・データベースに接続するクライアントのリージョナル・リスナーとして機能します。シャード・ディレクタはシャード・データベースの現在のトポロジ・マップを維持します。接続リクエスト中に渡されたシャーディング・キーに基づいて、シャード・ディレクタは適切なシャードに接続をルーティングします。

一般的なシャード・データベースの場合、一連のシャード・ディレクタが各リージョンの市販のローエンドの専用サーバーにインストールされます。高可用性とスケーラビリティを実現するには、複数のシャード・ディレクタをデプロイします。特定の領域に最大5つのシャード・ディレクタをデプロイできます。

シャード・ディレクタの主な機能を次に示します。

- シャード・データベース構成に関する実行時データおよびシャードの可用性の保守
- 配下のリージョンと他のリージョンの間のネットワーク待機時間の測定
- クライアントがシャード・データベースへの接続に使用するためのリージョナル・リスナーとして機能します
- グローバル・サービスの管理
- 接続ロード・バランシングの実行

グローバル・サービス

グローバル・サービスは、シャード・データベースのデータへのアクセスに使用されるデータベース・サービスです。

グローバル・サービスは、従来のデータベース・サービスの概念を拡張したものです。グローバル・サービスでは、従来のデータベース・サービスのすべてのプロパティがサポートされます。シャード・データベースでは、グローバル・サービス用の追加のプロパティが設定されます。たとえば、データベース・ロール、レプリケーション・ラグの許容値、クライアントとシャード間のリージョン・アフィニティなどがあります。読取り/書込みトランザクション・ワークロードの場合は、シャード・データベースのプライマリ・シャードのデータにアクセスするために、単一のグローバル・サービスが作成されます。Active Data Guardを使用する高可用性シャードの場合は、別個の読取り専用グローバル・サービスを作成できます。

パーティション、表領域およびチャンク

異なるシャードに存在する表領域にパーティションを作成することで、シャード間にパーティションを分散します。

シャード表の各パーティションが個別の表領域に格納され、表領域がSDBのデータ分散の単位になります。

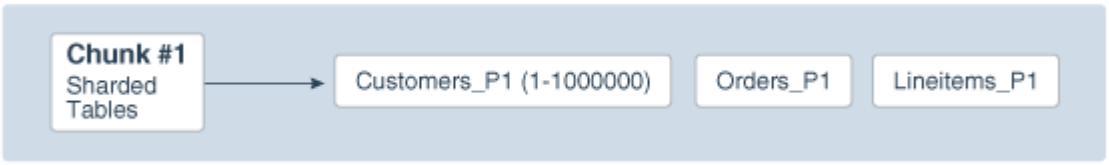
シャード表ファミリで説明するように、マルチシャード結合の数を最小にするために、表ファミリのすべての表の対応するパーティションが常に同じシャードに格納されます。構文の例で表領域セットts1がすべての表に使用されているように、1つの表ファミリの表が分散された表領域の同じセットに作成されるときに、これが保証されます。

ただし、1つの表ファミリの異なる表を異なる表領域のセットに作成することも可能です。たとえば、表領域セットts1にCustomers表を作成し、表領域セットts2にOrdersを作成できます。この場合、Customersのパーティション1が格納される表領域が常に、Ordersのパーティション1が格納される表領域と同じシャードに存在することが保証される必要があります。

この機能をサポートするために、表ファミリのすべての表の対応するパーティションのセット(チャンクと呼ばれる)が作成されます。チャンクには表ファミリの各表の1つのパーティションが含まれます。これより、様々なシャード表の関連するデータが一緒に移動されることが保証されます。つまり、チャンクはシャード間のデータ移行の単位です。システム管理シャーディングおよび複合シャーディングでは、シャード・データベースの作成時に各シャード内のチャンク数が指定されます。ユーザー定義シャードでは、チャンクの合計数はパーティションの数と同じです。

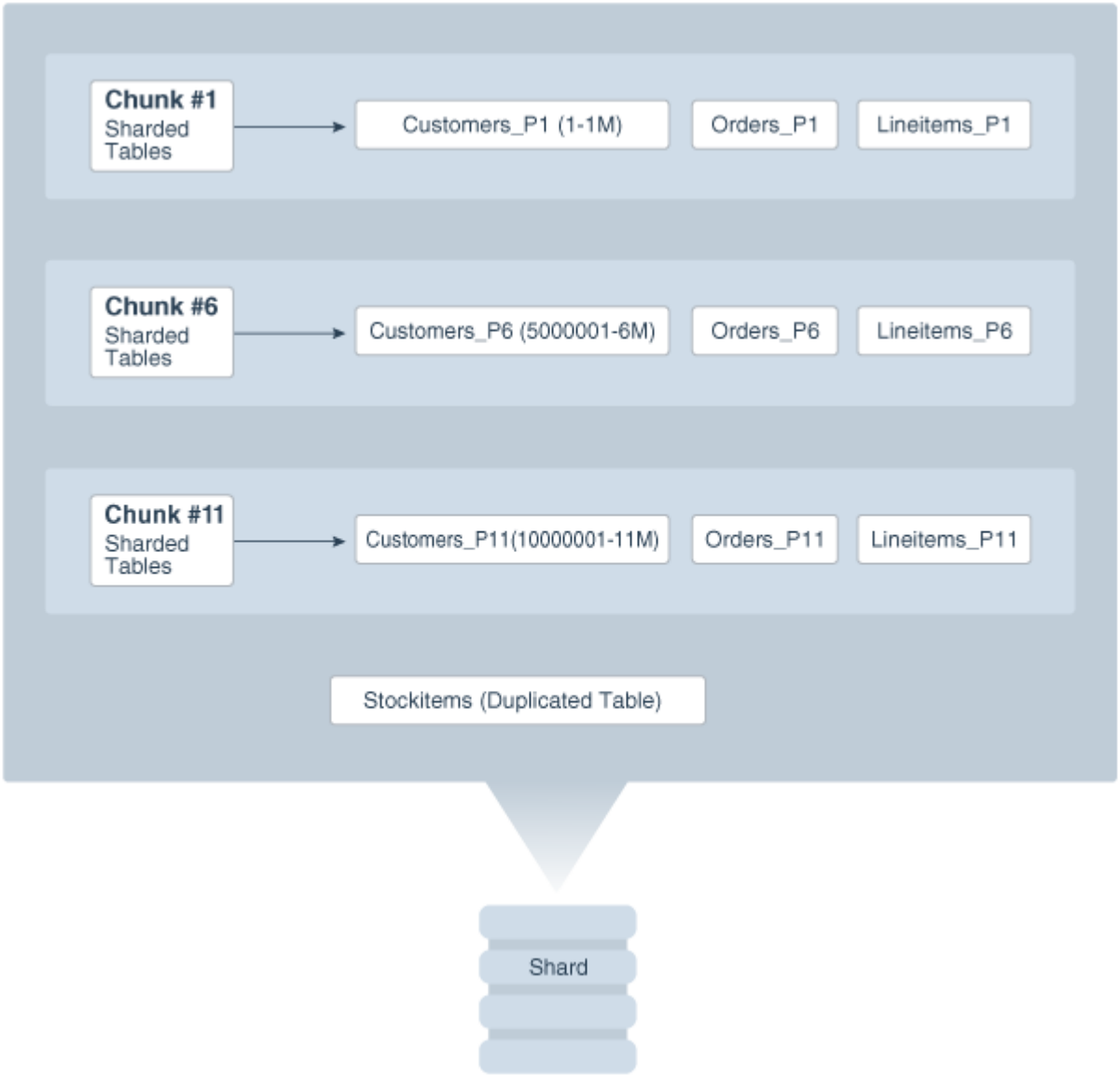
次の図に、Cutomers-Orders-LineItemsスキーマの表の対応するパーティションを含むチャンクを示します。

図2-2 パーティションのセットとしてのチャンク



次の図のように、各シャードに複数のチャンクが含まれます。

図2-3 シャードの内容



シャード表に加えて、シャードには1つ以上の重複表も含めることができます。重複表は、シャード表に使用される表領域には格

納できません。

表領域セット

Oracle Shardingは、TABLESPACE SETと呼ばれる単位として表領域を作成および管理します。

システム管理シャーディングと複合シャーディングではTABLESPACE SETを使用し、ユーザー定義シャーディングでは通常の表領域を使用します。

表領域は、シャード・データベースでのデータ分散の論理単位です。異なるシャードに存在する表領域にパーティションを自動的に作成することで、シャード間にパーティションを分散します。マルチシャード結合の数を最小にするために、関連する表の対応するパーティションは常に同じシャードに格納されます。シャード表の各パーティションが個別の表領域に格納されます。

PARTITIONS AUTO句は、パーティション数を自動的に判別することを指定します。このハッシュ・タイプによってシャード間のデータ移行でさらに柔軟性および効率性がもたらされ、これは弾力的な拡張性にとって重要です。

表領域セットごとに作成される表領域の数は、デプロイメント時にシャード領域に定義されたチャンクの数に基づいて決定されます。

ノート:

表領域セットでは、Oracle Managed Files のみがサポートされます。

個々の表領域を表領域セット全体とは別個に削除または変更することはできません。

TABLESPACE SET をユーザー定義のシャーディング方法とともに使用することはできません。

シャーディング方法

次のトピックでは、Oracle Shardingでサポートされるシャーディング方法、方法の選択およびサブパーティションの使用方法について説明します。

システム管理のシャーディング

システム管理のシャーディングでは、シャードへのデータのマッピングをユーザーが指定する必要はありません。コンシステント・ハッシュによるパーティション化を使用して、データが自動的にシャード間に分散されます。パーティション化アルゴリズムにより、データがシャード間に均一およびランダムに分散されます。

システム管理のシャーディングで使用される分散はホット・スポットを回避し、シャード間で均一なパフォーマンスを実現するためのものです。Oracle Shardingは、SDBへのシャードの追加または削除の際に、バランスの取れたチャンク分散を自動的に維持します。

コンシステント・ハッシュはスケーラブルな分散システムでよく使用されるパーティション化の方法です。従来のハッシュ・パーティション化とは異なります。従来のハッシュの場合はバケット数がHF（key） % Nとして計算されます。ここで、HFはハッシュ関数、Nはバ

ケット数です。この方法はNが一定の場合はうまく機能しますが、Nが変化するとすべてのデータの再シャッフルが必要になります。

線形ハッシュ法などのより高度なアルゴリズムの場合、ハッシュ・バケットを追加するために表全体を再ハッシュする必要はありませんが、バケット数が2のべき乗のみ、およびバケットを分割できるオーダーのみなど、バケット数に制限があります。

Oracle Shardingで使用するコンシステント・ハッシュの実装では、このような制限を回避するために、次の図のように、ハッシュ関数の値の取りうる範囲(たとえば、0から2^{32})をN個の隣接する間隔のセットに分割し、それぞれの間隔をチャンクに割り当てます。この例では、SDBに1024個のチャンクが含まれ、各チャンクに2^{22}のハッシュ値の範囲が割り当てられます。したがって、コンシステント・ハッシュによるパーティション化は、本質的にはハッシュ値の範囲によるパーティション化です。

図2-4 チャンクに割り当てられるハッシュ値の範囲

すべてのシャードの計算能力が同じだとすると、SDBの各シャードに等しい数のチャンクが割り当てられます。たとえば、16個のシャードを含むSDB内に1024個のチャンクが作成されると、各シャードに64個のチャンクが含まれます。

SDBへのシャードの追加や削除の際の再シャーディングの場合、いくつかのチャンクがシャード間を移動して、シャード間のチャンクの均一な分散が維持されます。このプロセスでチャンクの内容は変化せず、再ハッシュは行われません。

1つのチャンクが分割されるときに、ハッシュ値の範囲が2つの範囲に分割されますが、他のチャンクに対して何かを行う必要はありません。任意のチャンクをいつでも個別に分割できます。

接続リクエストのシャードへの送信にかかわるSDBのすべてのコンポーネントに、各シャードでホストされるチャンクのリスト、および各チャンクに関連付けられたハッシュ値の範囲を含むルーティング表が保持されます。特定のデータベース・リクエストのルーティング先を決定するために、ルーティング・アルゴリズムによって、シャーディング・キーで指定された値にハッシュ関数が適用され、計算されたハッシュ値が適切なチャンクにマップされ、次にチャンクを含むシャードにマップされます。

システム管理のシャーディングのSDB内のチャンク数は、GDSCTLコマンド、CREATE SHARDCATALOGで指定できます。指定しない場合は、デフォルト値としてシャード当たり120個のチャンクが使用されます。SDBのデプロイ後にチャンク数を変更する方法はチャンクの分割のみです。

コンシステント・ハッシュによってパーティション化されたシャード表を作成する前に、表のパーティションを格納するために表領域のセット(チャンク当たり1つの表領域)を作成する必要があります。表領域はCREATE TABLESPACE SETのSQL文を実行すると自動的に作成します。

表領域セットのすべての表領域が同じ物理属性を持ち、Oracle Managed Files (OMF)のみを格納できます。最も簡単な形式の場合、CREATE TABLESPACE SET文は、たとえば次のように表領域セットの名前という1つのパラメータのみを持ちます。

```
CREATE TABLESPACE SET ts1;
```

この場合、セットの各表領域にデフォルトの属性を持つ1つのOMFファイルが含まれます。表領域の属性をカスタマイズするには、文にUSING TEMPLATE句(次の例を参照)を追加します。USING TEMPLATE句は、セットの各表領域に適用される属性を指定

します。

```
CREATE TABLESPACE SET ts1
USING TEMPLATE
(
 DATAFILE SIZE 10M
 EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K
 SEGMENT SPACE MANAGEMENT AUTO
 ONLINE
)
;
```

表領域セットが作成されたら、コンシステント・ハッシュでパーティション化される表を作成して、セットに属する表領域にパーティションを格納できます。CREATE TABLE文は、たとえば次のようになります。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3)
, signup      DATE
, CONSTRAINT cust_pk PRIMARY KEY(cust_id)
)
PARTITION BY CONSISTENT HASH (cust_id)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

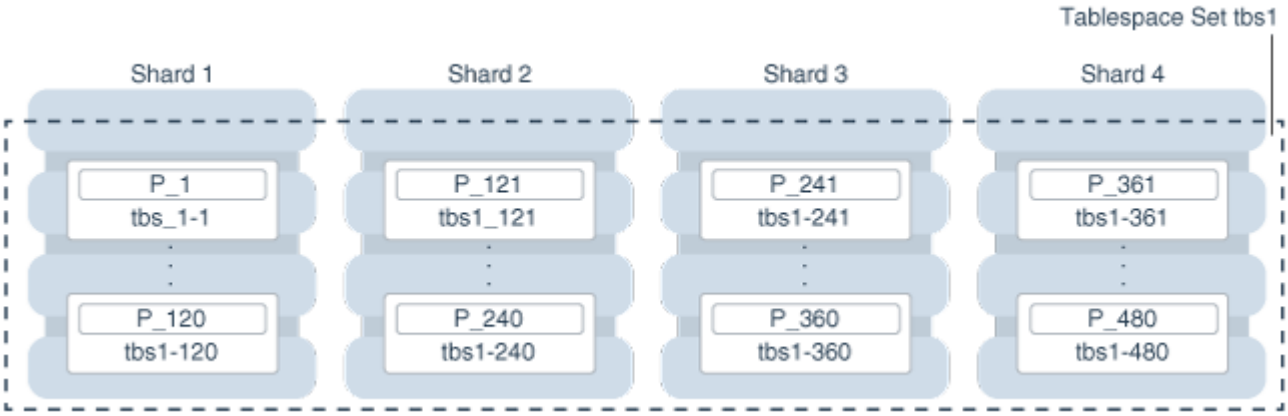
この文のPARTITIONS AUTOは、パーティション数が自動的に表領域セットts1の表領域の数(チャンク数と等しい)に設定され、各パーティションが個別の表領域に格納されることを意味します。

表領域セットの各表領域が別個のチャンクに属します。つまり、1つのチャンクは所定の表領域セットの1つの表領域のみを含むことができます。ただし、同じ表ファミリに属する複数の表に同じ表領域セットを使用できます。この場合、セットの各表領域に複数のパーティション(各表から1つ)が格納されます。

または、表ファミリの各表を個別の表領域セットに格納できます。この場合、1つのチャンクに複数の表領域(各表領域セットから1つ)が含まれ、各表領域に1つのパーティションが格納されます。

次の図は、単一シャード表のユースケースの場合のパーティション、表領域およびシャード間の関係を示しています。この場合、各チャンクに1つの表領域が含まれ、各表領域に1つのパーティションが格納されます。

図2-5 システム管理のシャーディング



ノート:

シャーディング方法は GDSCTL CREATE SHARDCATALOG コマンドで指定し、後から変更することはできません。

ユーザー定義のシャーディング

ユーザー定義のシャーディングでは、個々のシャードへのデータのマッピングをユーザーが明示的に指定できます。これは、パフォーマンスや規制などの理由で、特定のデータを特定のシャードに格納する必要があり、管理者がシャード間のデータの移動を完全に制御する必要がある場合に使用します。

ユーザー定義のシャード・データベースでは、Oracle Data GuardとOracle Active Data Guardの2つのレプリケーション・スキーマがサポートされます。レプリケーション方法としてOracle GoldenGateが使用されている場合、ユーザー定義のシャーディングはサポートされません。

ユーザー定義のシャーディングのもう1つのメリットとして、シャードの計画停止または計画外停止の場合に、どのデータが使用できないかをユーザーが正確に知ることができます。ユーザー定義のシャーディングのデメリットは、データベース管理者が、シャード間のデータおよびワークロードの分散のバランスを監視して維持する必要があることです。

ユーザー定義のシャーディングの場合、範囲またはリストによってシャード表をパーティション化できます。シャード表のCREATE TABLE構文は、各パーティションを個別の表領域に格納する必要があることを除き、通常の表のための構文とそれほど違いはありません。

```
 CREATE SHARDED TABLE accounts
( id             NUMBER
, account_number NUMBER
, customer_id    NUMBER
, branch_id      NUMBER
, state          VARCHAR(2) NOT NULL
, status         VARCHAR2(1)
)
PARTITION BY LIST (state)
( PARTITION p_northwest VALUES ('OR', 'WA') TABLESPACE ts1
, PARTITION p_southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE ts2
, PARTITION p_northcentral VALUES ('SD', 'WI') TABLESPACE ts3
, PARTITION p_southcentral VALUES ('OK', 'TX') TABLESPACE ts4
, PARTITION p_northeast VALUES ('NY', 'VM', 'NJ') TABLESPACE ts5
, PARTITION p_southeast VALUES ('FL', 'GA') TABLESPACE ts6
)
;
```

ユーザー定義のシャーディングには表領域セットはありません。各表領域を個別に作成し、シャード領域と明示的に関連付ける必要があります。シャード領域はシャードのセットで、キーの値の範囲またはリストに対応するデータが格納されます。

ユーザー定義のシャーディングでは、シャード領域が1つのシャード、または完全にレプリケートされたシャードのセットから構成されます。ユーザー定義のシャーディングでのレプリケーションの詳細は、シャード・レベルの高可用性を参照してください。わかりやすいように、各シャード領域が1つのシャードから構成されると仮定します。

次の構文を使用して、前述の例に示したaccounts表の表領域を作成します。

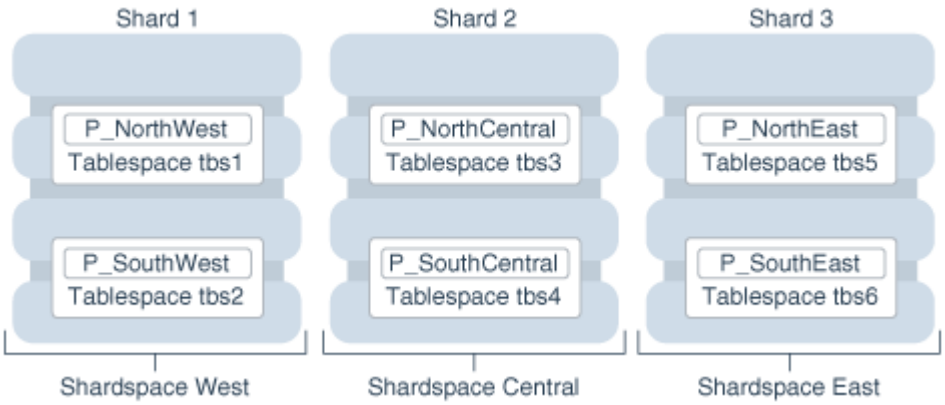
```
CREATE TABLESPACE tbs1 IN SHARDSPACE west;
CREATE TABLESPACE tbs2 IN SHARDSPACE west;
CREATE TABLESPACE tbs3 IN SHARDSPACE central;
CREATE TABLESPACE tbs4 IN SHARDSPACE central;
CREATE TABLESPACE tbs5 IN SHARDSPACE east;
CREATE TABLESPACE tbs6 IN SHARDSPACE east;
```

CREATE TABLESPACE文を実行する前に、シャード領域を作成し、シャードを移入する必要があります。たとえば、次のGDSCTLコマンドを使用できます。

```
ADD SHARDSPACE -SHARDSPACE east
ADD SHARDSPACE -SHARDSPACE central
ADD SHARDSPACE -SHARDSPACE west
ADD SHARD -CONNECT shard-1 -SHARDSPACE west;
ADD SHARD -CONNECT shard-2 -SHARDSPACE central;
ADD SHARD -CONNECT shard-3 -SHARDSPACE east;
```

次の図は、前述の例に示したaccounts表について、表領域へのパーティションのマッピング、およびシャードへの表領域のマッピングを示しています。

図2-6 ユーザー定義のシャーディング



システム管理のシャーディングと同様に、ユーザー定義のシャーディングで作成される表領域もチャンクに割り当てられます。ただし、SDBにシャードを追加したときに、チャンクの移行は自動的には開始しません。移行する必要がある各チャンクに対して、ユーザーがGDSCTLコマンドMOVE CHUNKを実行する必要があります。

GDSCTLコマンドSPLIT CHUNKはシステム管理のシャーディングでは、ハッシュ範囲の中央でチャンクを分割するために使用されますが、ユーザー定義のシャーディングではサポートされません。ALTER TABLE SPLIT PARTITION文を使用してチャンクを分割する必要があります。

ノート:

シャーディング方法は GDSCTL CREATE SHARDCATALOG コマンドで指定し、後から変更することはできません。

コンポジット・シャーディング

コンポジット・シャーディング方法では、コンシステント・ハッシュによりパーティション化された表のデータの異なるサブセットについて、複数のシャード領域を作成できます。シャード領域はシャードのセットで、キーの値の範囲またはリストに対応するデータが格納されます。

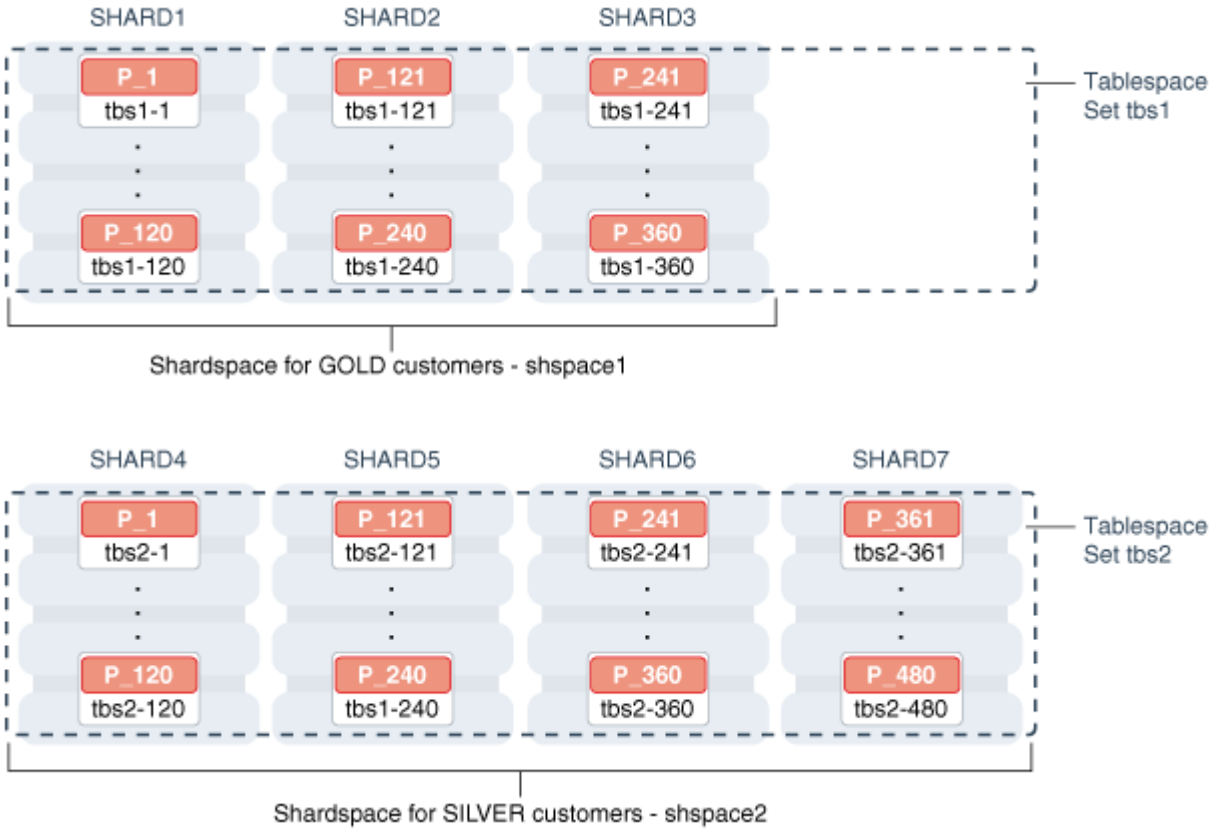
システム管理のシャーディングでは、コンシステント・ハッシュによるパーティション化を使用して、シャード間にデータをランダムに分散します。これは、範囲またはリストによるパーティション化を使用するユーザー定義のシャーディングと比べて、よりよいロード・バランスを実現します。ただし、システム管理のシャーディングでは、データのシャードへの割当てをユーザーが制御できません。

主キーのコンシステント・ハッシュによるシャーディングの際には、データのサブセットを異なる地理的場所に格納する、データのサブセットに異なるハードウェア・リソースを割り当てる、または高可用性と障害回復を異なる構成にするために、SDB内のデータのサブセットを区別する必要が生じることがよくあります。通常は、この区別は顧客の場所やサービスのクラスなど、別の(主キーでない)列の値に基づいて行われます。

コンポジット・シャーディングは、ユーザー定義のシャーディングとシステム管理のシャーディングの組合せで、必要に応じて両方の利点を活用できます。コンポジット・シャーディングでは、最初にデータがリストまたは範囲により複数のシャード領域にパーティション化され、次にコンシステント・ハッシュにより各シャード領域の複数のシャード間にさらにパーティション化されます。2つのレベルのシャーディングにより、各シャード領域のシャード間のバランスのとれたデータ分散を自動的に維持し、同時にシャード領域間にデータをパーティション化できます。

たとえば、高速なサーバーでホストされる3つのシャードをゴールド顧客に割り当て、低速のマシンでホストされる4つのシャードをシルバー顧客に割り当てるとします。顧客IDのコンシステント・ハッシュによるパーティション化を使用して、シャードの各セット内に顧客を分散する必要があります。

図2-7 コンポジット・シャーディング



そのような構成には2つのシャード領域を作成する必要があります。たとえば、次のGDSCTLコマンドを使用できます。

```
ADD SHARDSPACE -SHARDSPACE shspace1;
ADD SHARDSPACE -SHARDSPACE shspace2;
ADD SHARD -CONNECT shard1 -SHARDSPACE shspace1;
ADD SHARD -CONNECT shard2 -SHARDSPACE shspace1;
ADD SHARD -CONNECT shard3 -SHARDSPACE shspace1;
ADD SHARD -CONNECT shard4 -SHARDSPACE shspace2;
ADD SHARD -CONNECT shard5 -SHARDSPACE shspace2;
ADD SHARD -CONNECT shard6 -SHARDSPACE shspace2;
ADD SHARD -CONNECT shard7 -SHARDSPACE shspace2;
```

コンポジット・シャーディングでは、他のシャーディング方法と同様に、表領域を使用してシャードへのパーティションのマッピングを指定します。シャード表のデータのサブセットを異なるシャード領域に割り当てるために、次の例に示すように、各シャード領域に個別の表領域のセットを作成する必要があります。

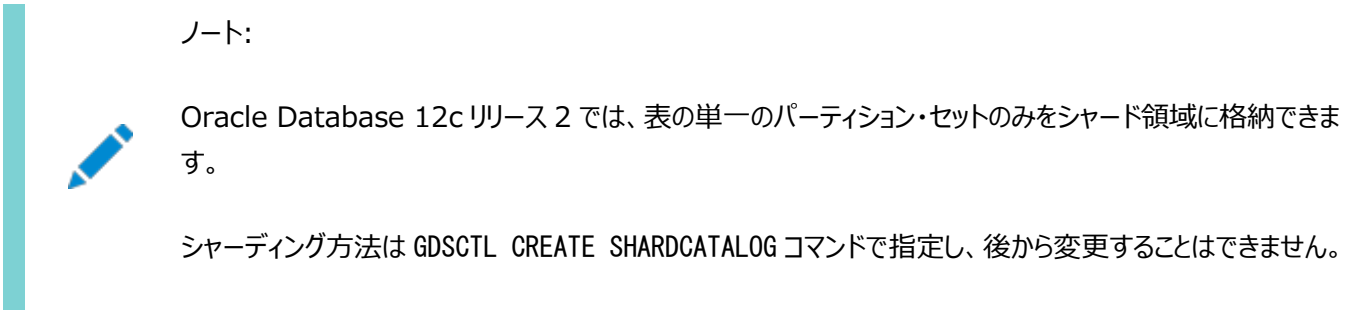
```
CREATE TABLESPACE SET tbs1 IN SHARDSPACE shspace1;
CREATE TABLESPACE SET tbs2 IN SHARDSPACE shspace2;
```

ユーザー定義のデータのサブセットを異なる表領域に格納するために、Oracle Shardingには、パーティションをセットにグループ化し、パーティションの各セットを表領域のセットと関連付ける構文が用意されています。パーティション・セットのサポートは、論理的には、コンシステント・ハッシュによるパーティション化の上位に実装される高レベルのパーティション化に相当すると考えることができます。

次の例に示す文では、サービスのクラスに基づいて、シャード表をgoldとsilverという2つのパーティション・セットにパーティション化しています。各パーティション・セットが個別の表領域に格納されます。次に、各パーティション・セットのデータが、顧客IDのコンシステント・ハッシュによってさらにパーティション化されます。

```
CREATE SHARDED TABLE customers
```

```
( cust_id NUMBER NOT NULL
, name VARCHAR2(50)
, address VARCHAR2(250)
, location_id VARCHAR2(20)
, class VARCHAR2(3)
, signup_date DATE
, CONSTRAINT cust_pk PRIMARY KEY(cust_id, class)
)
PARTITIONSET BY LIST (class)
  PARTITION BY CONSISTENT HASH (cust_id)
  PARTITIONS AUTO
(PARTITIONSET gold VALUES ( 'gld' ) TABLESPACE SET tbs1,
 PARTITIONSET silver VALUES ( 'slv' ) TABLESPACE SET tbs2)
;
```



ノート:

Oracle Database 12c リリース 2 では、表の単一のパーティション・セットのみをシャード領域に格納できます。

シャーディング方法は GDSCTL CREATE SHARDCATALOG コマンドで指定し、後から変更することはできません。

サブパーティション化とシャーディングの使用

Oracle Shardingは表のパーティション化に基づくため、Oracle Databaseで提供されるすべてのサブパーティション方法がシャーディングでもサポートされます。

サブパーティション化は各パーティションをさらに小さい部分に分割します。これはシャード内の効率的なパラレル実行、特に、シャード当たりのパーティション数が少ないときの範囲またはリストによるシャーディングの場合に役立つ可能性があります。

管理性の面では、サブパーティションを個別の表領域に割り当てて記憶域階層間で移動することで、サブパーティション化によって記憶域の階層化をサポートできます。シャーディングのスケーラビリティと可用性という利点を損なわず、パーティション・プルーニングと主キーによるパーティション単位の結合の実行を犠牲にすることなく、記憶域階層間でサブパーティションを移行できます。

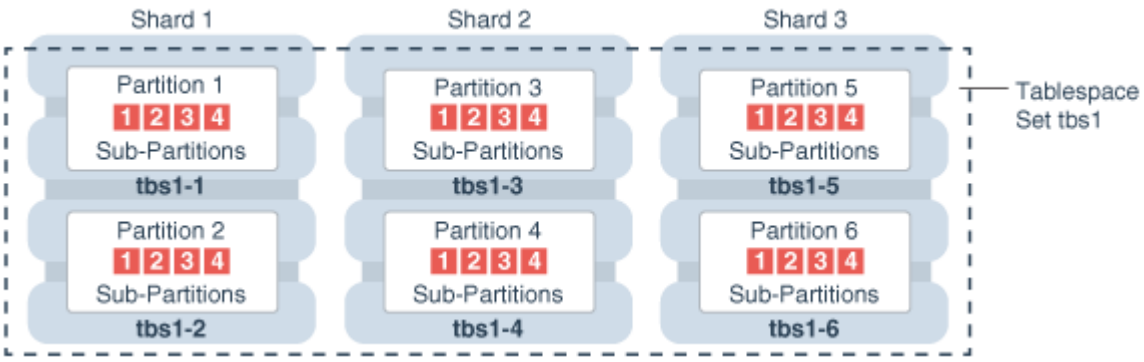
次の例は、コンシステント・ハッシュと範囲によるサブパーティション化を組み合せたシステム管理のシャーディングを示しています。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3)
, signup_date DATE
, CONSTRAINT cust_pk PRIMARY KEY(cust_id, signup_date)
)
TABLESPACE SET ts1
PARTITION BY CONSISTENT HASH (cust_id)
SUBPARTITION BY RANGE (signup_date)
SUBPARTITION TEMPLATE
( SUBPARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
  SUBPARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
  SUBPARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2020','DD/MM/YYYY')),
```

```
  SUBPARTITION future VALUES LESS THAN (MAXVALUE)
)
PARTITIONS AUTO
;
```

次の図は、この文によって作成される表を示しています。

図2-8 親パーティションの表領域に格納されるサブパーティション



この例では、各サブパーティションが親パーティションの表領域に格納されます。日付によってサブパーティション化されているため、サブパーティションを個別の表領域に格納して、古いデータをアーカイブしたり、読取り専用の記憶域に移動できるようにするほうが合理的です。適切な構文を次に示します。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3)
, signup_date DATE NOT NULL
, CONSTRAINT cust_pk PRIMARY KEY(cust_id, signup_date)
)
PARTITION BY CONSISTENT HASH (cust_id)
SUBPARTITION BY RANGE(signup_date)
SUBPARTITION TEMPLATE
( SUBPARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY'))
       TABLESPACE SET ts1,
  SUBPARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY'))
       TABLESPACE SET ts2,
  SUBPARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2020','DD/MM/YYYY'))
       TABLESPACE SET ts3,
  SUBPARTITION future VALUES LESS THAN (MAXVALUE)
       TABLESPACE SET ts4
)
PARTITIONS AUTO
;
```

シャーディングされていないデータベースの場合は、サブパーティション・テンプレートで表領域を指定すると、すべてのパーティションのサブパーティションNが同じ表領域に格納されることに注意してください。これは、シャーディングされている場合に、異なるパーティションに属するサブパーティションを個別の表領域に格納して、再シャーディングの際に移動できるようにする場合とは異なります。

サブパーティション化はコンポジット・シャーディングでも使用できます。この場合、表のデータがパーティション・セット、パーティション

およびサブパーティションの3つのレベルに編成されます。次に、3つのレベルにデータを編成する例を示します。

パーティションセット間でサブパーティションの数と境界を均一にするため、パーティションセット単位でのサブパーティション・テンプレートの指定はサポートされていません。パーティションセット単位でサブパーティションの表領域を指定する必要がある場合は、SUBPARTITIONS STORE IN句を使用できます。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3) NOT NULL
, signup_date DATE NOT NULL
, CONSTRAINT cust_pk PRIMARY KEY(cust_id, class, signup_date)
)
PARTITIONSET BY LIST (class)
PARTITION BY CONSISTENT HASH (cust_id)
SUBPARTITION BY RANGE (signup_date)
  SUBPARTITION TEMPLATE /* applies to both SHARDSPACEs */
  ( SUBPARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY'))
  , SUBPARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY'))
  , SUBPARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2020','DD/MM/YYYY'))
  , SUBPARTITION future VALUES LESS THAN (MAXVALUE)
)
PARTITIONS AUTO
(
  PARTITIONSET gold   VALUES ( 'gld' ) TABLESPACE SET tbs1
 subpartitions store in(tbs1)
, PARTITIONSET silver VALUES ( 'slv' ) TABLESPACE SET tbs2
 subpartitions store in(tbs2)
)
;
```

シャード・データベースのスキーマ・オブジェクト

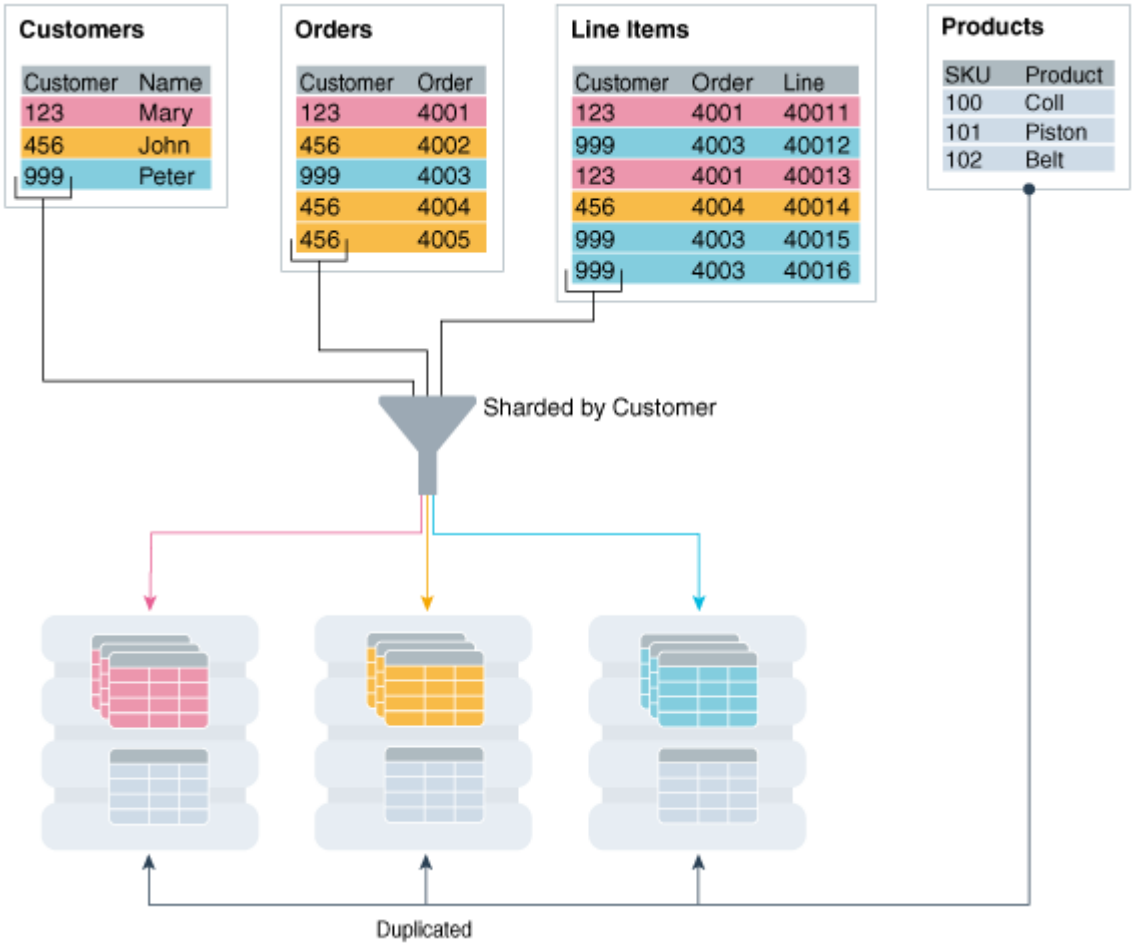
シャーディングの利点を活用するには、1つのシャードで実行されるデータベース・リクエストの数が最大になるように、シャード・データベースのスキーマを設計する必要があります。次のトピックでは、設計を通知するシャード・データベースを形成するスキーマ・オブジェクトを定義および説明します。

シャード表

データベース表はシャード間で分割されるため、各シャードには同じ列を持つが行のサブセットが異なる表が含まれます。この方法で分割された表はシャード表と呼ばれます。

次の図は、左側の1つのデータベースに表示される大規模な表のセット(表ファミリと呼ばれる)を右側に表示される3つのシャード間で水平にパーティション化して、各シャードに赤、黄および青の行で示されるデータのサブセットを含める方法を示しています。

図2-9 シャード間への表の水平パーティション化



パーティションはシャーディング・キーに基づいて表領域レベルでシャード間に分散されます。キーの例として顧客ID、アカウント番号、国IDなどがあります。シャーディング・キーでは次のデータ型がサポートされています。

- NUMBER
- INTEGER
- SMALLINT
- RAW
- (N)VARCHAR
- (N)VARCHAR2
- (N)CHAR
- DATE
- TIMESTAMP

シャード表の各パーティションが個別の表領域に存在し、各表領域が特定のシャードと関連付けられます。シャーディング方法に応じて、関連付けは自動的に確立されるか、管理者によって定義されることができます。

シャード表のパーティションが複数のシャード内に存在する場合でも、アプリケーションにとってその表は、単一データベース内のパーティション表とまったく同じように表示され、動作します。アプリケーションによって発行されたSQL文はシャードを参照する必要はなく、シャードの数およびその構成に依存することもありません。

シャード間で行をパーティション化する方法は、表のパーティション化でよく目にするSQL構文によって指定します。たとえば、次のSQL文はシャーディング・キーcust_idに基づいて、シャード間で表を水平にパーティション化したシャード表を作成します。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, region      VARCHAR2(20)
, class       VARCHAR2(3)
, signup      DATE
,CONSTRAINT cust_pk PRIMARY KEY(cust_id)
)
PARTITION BY CONSISTENT HASH (cust_id)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

シャード表はコンシステント・ハッシュ(スケーラブルな分散システムで一般的に使用される特殊なタイプのハッシュ・パーティション化)によりパーティション化されます。この手法は、シャード間で表領域を自動的に分散するため、データおよびワークロードが均等に分散されます。

ノート:

シャード表でのグローバル索引はサポートされていませんが、ローカル索引はサポートされています。

シャード表ファミリ

シャード表ファミリは、同様にシャーディングされた一連の表です。しばしば、データベースの表には親子関係があり、親表の主キーを参照する子表(外部キー)には参照制約があります。

このような関係でリンクされている複数の表は、通常はツリーのような構造になり、各子が1つの親を持ちます。そのような一連の表は、表ファミリと呼ばれます。表ファミリ内の親を持たない表をルート表と呼びます。1つの表ファミリに存在できるルート表は1つのみです。

表ファミリのシャーディング方法

ここでは、Customers-Orders-LineItemsスキーマを使用して表ファミリをシャーディングします。

シャーディングの前に、スキーマ内の表は次の例のようになります。3つの表に親子関係があり、Customersがルート表になります。

シャーディング前のCustomers表(ルート)

```
CustNo    Name       Address         Location  Class
--------- ---------- --------------- --------- ------
123       Brown      100 Main St     us3       Gold
456       Jones      300 Pine Ave    us1       Silver
999       Smith      453 Cherry St   us2       Bronze
```

シャーディング前のOrders表

```
OrderNo   CustNo   OrderDate
--------- -------- -----------
4001      123      14-FEB-2013
4002      456      09-MAR-2013
```

```
4003        456        05-APR-2013
4004        123        27-MAY-2013
4005        999        01-SEP-2013
```

シャーディング前のLineItems表

```
LineNo   OrderNo   CustNo   StockNo    Quantity
------   -------   ------   -------    --------
40011    4001      123      05683022   1
40012    4001      123      45423609   4
40013    4001      123      68584904   1
40021    4002      456      05683022   1
40022    4002      456      45423509   3
40022    4003      456      80345330   16
40041    4004      123      45423509   1
40042    4004      123      68584904   2
40051    4005      999      80345330   12
```

ルートであるCustomers表の顧客番号CustNoに基づいて表をシャーディングできます。次の表の例には、顧客123に関連するデータが格納されたシャードが示されています。

顧客123のデータを含むCustomers表シャード

```
CustNo     Name       Address          Location   Class
---------  ---------- ---------------  ---------- ------
123        Brown      100 Main St      us3        Gold
```

顧客123のデータを含むOrders表シャード

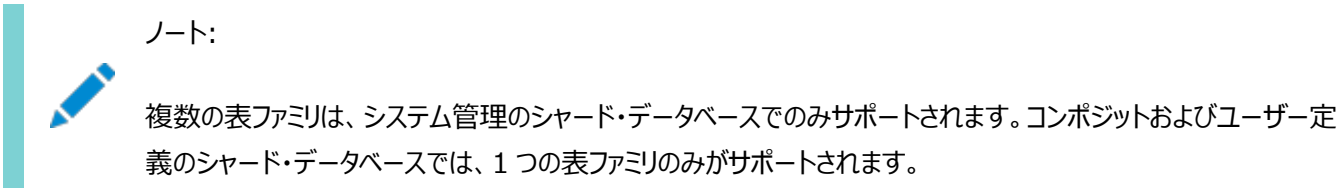
```
OrderNo    CustNo   OrderDate
---------  -------- -----------
4001       123      14-FEB-2013
4004       123      27-MAY-2013
```

顧客123のデータを含むLineItems表シャード

```
LineNo   OrderNo   CustNo   StockNo    Quantity
------   -------   ------   -------    --------
40011    4001      123      05683022   1
40012    4001      123      45423609   4
40013    4001      123      68584904   1
40041    4004      123      45423509   1
40042    4004      123      68584904   2
```

複数の表ファミリを使用したスキーマの設計

シャード・データベース・スキーマには複数の表ファミリを含めることができます。この場合、異なる表ファミリのデータはすべて同じチャンクに存在し、チャンクには同じハッシュ・キー範囲を共有する異なる表ファミリからのパーティションが含まれます。



ノート:

複数の表ファミリは、システム管理のシャード・データベースでのみサポートされます。コンポジットおよびユーザー定義のシャード・データベースでは、1 つの表ファミリのみがサポートされます。

新しい表ファミリを作成するには、ルート・シャード表を作成し、既存の表領域ファミリで使用されない表領域セットを指定します。各表ファミリは、ルート表によって識別されます。異なる表ファミリの表は、互いに関連付けないでください。

各表ファミリには独自のシャーディング・キー定義が必要ですが、子表に同じシャーディング・キー列があるという同じ制限が各表ファミリ内で当てはまります。つまり、異なる表ファミリからのすべての表は、コンシステント・ハッシュと同じ方法で同じ数のチャンクにシャーディングされ、各チャンクにはすべての表ファミリからのデータが含まれます。

異なる表ファミリ間で問い合せるような表ファミリは最小限になるように設計し、シャーディング・コーディネータでのみ実行されるようにします。このような結合の多くはパフォーマンスに影響するためです。

次の例は、システム管理シャーディング方式(PARTITION BY CONSISTENT HASH)でPARENT句を使用して複数の表ファミリを作成する方法を示しています。

```
CREATE SHARDED TABLE Customers <=== Table Family #1
( CustId NUMBER NOT NULL
, Name VARCHAR2(50)
, Address VARCHAR2(250)
, region VARCHAR2(20)
, class VARCHAR2(3)
, signup DATE
)
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE Orders
( OrderNo NUMBER
, CustId NUMBER
, OrderDate DATE
)
PARENT Customers
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE LineItems
( LineNo NUMBER
, OrderNo NUMBER
, CustId NUMBER
, StockNo NUMBER
, Quantity NUMBER
)
)
PARENT Customers
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE Products <=== Table Family #2
( ProdId NUMBER NOT NULL,
  CONSTRAINT pk_products PRIMARY KEY (ProdId)
)
PARTITION BY CONSISTENT HASH (ProdId)
PARTITIONS AUTO
TABLESPACE SET ts_2
```

```
;
```

ノート:

表ファミリに表領域セットを使用しようとし、その表領域セットが既存の表ファミリですでに使用されている場合、ORA-3850 がスローされます。

表ファミリ間の結合は効率がよくないこともあるため、そのような結合が多数ある場合、またはパフォーマンスが重視されている場合は、複数の表ファミリではなく重複表を使用する必要があります。

グローバル・サービスと複数の表ファミリの関連付け

各表ファミリは、異なるグローバル・サービスに関連付ける必要があります。異なる表ファミリからのアプリケーションにはそれぞれ独自の接続プールおよびサービスがあり、適切なシャードにルーティングするために独自のシャーディング・キーを使用します。

最初のルート表(つまり、最初の表ファミリ)を作成すると、すべての既存のグローバル・サービスがその表に自動的に関連付けられます。この例に示すように、GDSCTL MODIFY SERVICEコマンドを使用して、表ファミリの作成後に、表ファミリに関連付けられているサービスを変更できます。

```
GDSCTL> MODIFY SERVICE -GDSPOOL shdpool -TABLE_FAMILY sales.customer -SERVICE sales
```

重複表

Oracle Shardingでは、各シャードに同じ内容の表を重複表と呼びます。

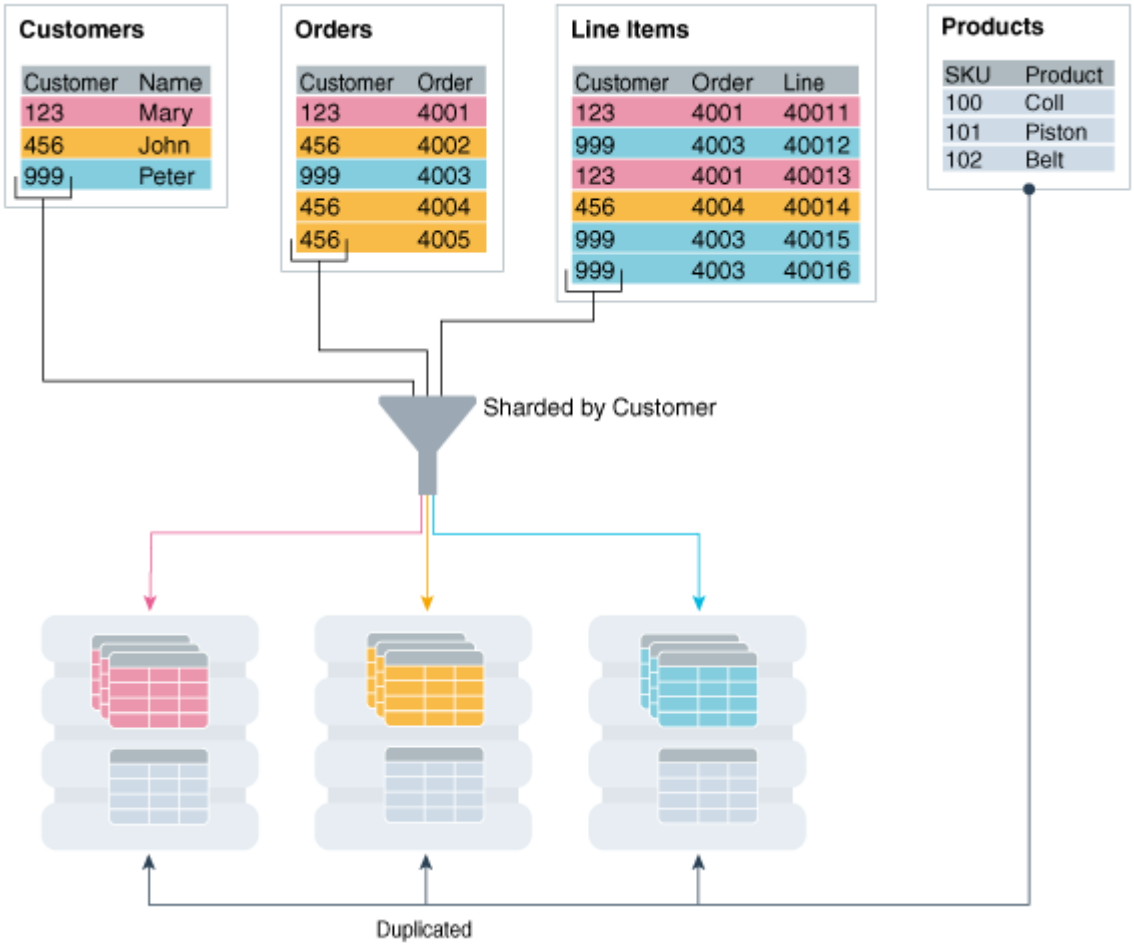
多くのアプリケーションの場合、単一のシャードで処理されるデータベース・リクエストの数は、すべてのシャードに読取り専用またはほぼ読取り専用の表を複製することによって最大化できます。この方法は、頻繁には更新されず、シャード表とともにアクセスされることが多い比較的小さい表に適しています。重複表は、シャード表よりも更新頻度が低い傾向があり、それほど大きくなるとは想定されません。

シャード・データベースには、シャード間で水平にパーティション化されたシャード表、およびすべてのシャードにレプリケートされる重複表が含まれています。重複表には参照情報が含まれています(たとえば、各シャードで共通のStock Items表)。シャード表と重複表の組合せによって、シャーディング・キーに関連付けられているすべてのトランザクションを単一のシャードで処理できます。この技法によって、線形のスケーラビリティおよび障害の分離が可能になります。

重複表が必要な例として、シャード表ファミリで説明される表ファミリを取り上げます。データベース・スキーマにはProducts表も含まれる可能性があり、この表には、この表ファミリに対して作成されたシャード内のすべての顧客が共有するデータが含まれますが、顧客番号でシャーディングすることはできません。注文処理中のマルチシャード問合せを回避するため、表全体をすべてのシャードに複製する必要があります。

シャード表(Customers、OrdersおよびLineItems)と重複表(Products)の違いを次の図に示します。

図2-10 シャード・データベースのシャード表と重複表



すべてのシャードに作成される表以外のオブジェクト

重複表に加えて、他のスキーマ・オブジェクト(ユーザー、ロール、ビュー、索引、シノニム、ファンクション、プロシージャ、パッケージなど)および非スキーマ・データベース・オブジェクト(表領域、表領域セット、ディレクトリ、コンテキストなど)をすべてのシャードに作成できます。

CREATE文に追加のキーワード(SHARDEDまたはDUPLICATED)が必要となる表と異なり、他のオブジェクトは既存の構文を使用してすべてのシャードに作成します。唯一の要件は、SHARD DDLセッション・プロパティを有効にする必要があることです。

すべてのシャードでの次のオブジェクトの自動作成は、このリリースではサポートされません。これらのオブジェクトは、個別のシャードに接続することによって作成できます。

- クラスタ
- 制御ファイル
- データベース・リンク
- ディスク・グループ
- エディション
- フラッシュバック・アーカイブ
- マテリアライズド・ゾーン・マップ
- アウトライン

- Pfile
- プロファイル
- リストア・ポイント
- ロールバック・セグメント
- サマリー

Oracle Database 18c以降では、マテリアライズド・ビューとビュー・ログがサポートされますが、次の制限事項があります。

- シャード表に対して作成されたマテリアライズド・ビューは、カタログ・データベースでは空のままですが、シャード上の対応するマテリアライズド・ビューには個々のシャードのデータが含まれています。
- シャード表のマテリアライズド・ビューでは、REFRESH COMPLETE ON DEMAND USING TRUSTED CONSTRAINTSオプションのみがサポートされます。

シャード・レベルの高可用性

Oracle Shardingは、シャード・レベルの高可用性と障害回復のためにOracle Databaseレプリケーション・テクノロジに統合されています。

次の各項で、Oracleのレプリケーション・テクノロジを使用してシャード・データベースの高可用性を実現する方法を説明します。

シャーディングとレプリケーションについて

Oracle Shardingは、Oracleのレプリケーションおよび障害時リカバリ・テクノロジであるOracle Data GuardおよびOracle GoldenGateと緊密に統合されています。

ノート:

Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。

レプリケーションによって高可用性、障害回復、および読取りのスケーラビリティ向上が実現します。レプリケーションの単位にはシャード、シャードの一部またはシャードのグループがあります。

シャード・データベースのレプリケーション・トポロジは、GDSCTLコマンド構文を使用して宣言的に指定します。2つのテクノロジ(Oracle Data GuardとOracle GoldenGate)のいずれかを選択して、データをレプリケートできます。Oracle Shardingは指定されたレプリケーション・トポロジを自動的にデプロイし、データ・レプリケーションを有効にします。

シャード・データベースの可用性は、1つ以上のシャードが停止したり、処理速度が低下しても影響されません。個別のシャードレベルの高可用性(Oracle Active Data GuardまたはOracle GoldenGate)を提供するために、レプリケーションが使用されます。レプリケーションはシャード・データベースが作成されると自動的に構成およびデプロイされます。オプションで、シャードレベルの高可用性のために、レプリケーションによって補完されるOracle RACを使用すると、クラスタが停止した場合にシャードレベルのデータの可用性を維持できます。Oracle Shardingは、計画外の停止が発生したときに、データベース接続をシャードからレプリカに自動的にフェイルオーバーします。

シャード・データベースでのOracle Data Guardの使用

Oracle Data Guardのレプリケーションは、高可用性およびデータ保護のために、シャード(プライマリ)の1つ以上の同期されたコピー(スタンバイ)を維持します。スタンバイはローカルまたはリモートにデプロイでき、Oracle Active Data Guardを使用する場合は、読取り専用アクセスでオープンすることもできます。

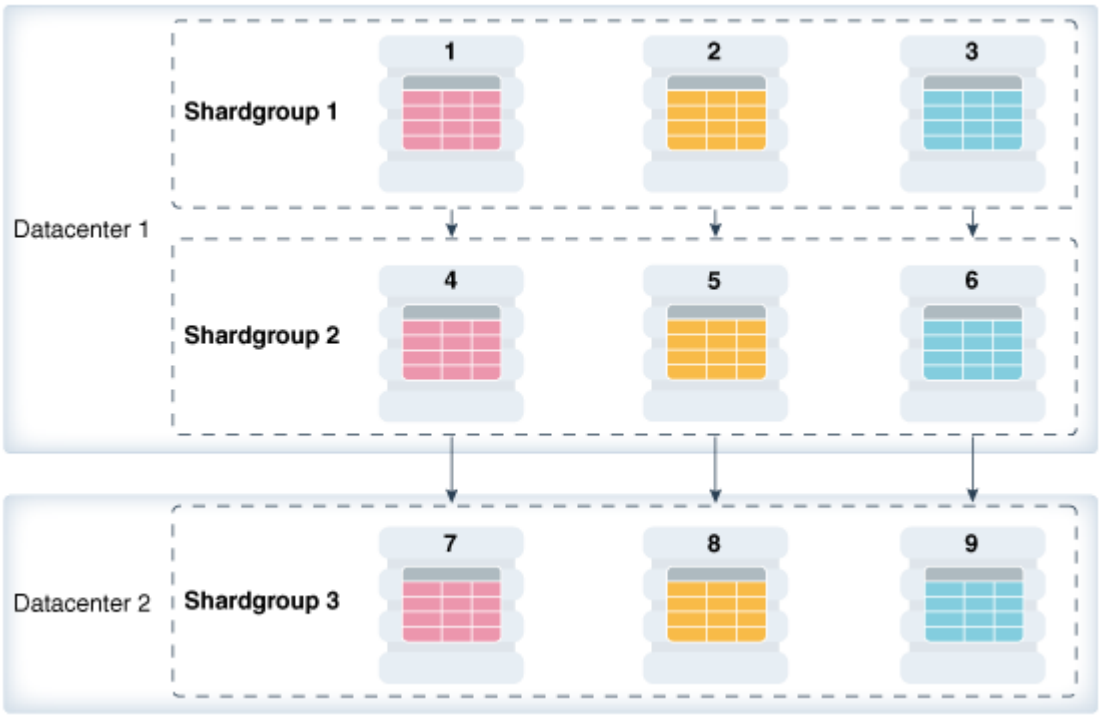
Oracle Data Guardは、シャーディング方法としてシステム管理、ユーザー定義またはコンポジットを使用するシャード・データベースのレプリケーション・テクノロジとして使用できます。

システム管理のシャード・データベースでのOracle Data Guardの使用

システム管理またはコンポジットのシャーディングの場合、レプリケーションの論理単位はシャードグループと呼ばれるシャードのグループです。システム管理のシャーディングの場合、シャード・データベースに格納されるすべてのデータがシャードグループに含まれます。データはシャードグループを構成するシャード間で、均一なハッシュによってシャードされます。シャードグループに属するシャードは通常、同じデータ・センターにあります。シャードグループ全体を、同じまたは異なるデータ・センターの1つ以上のシャードグループに完全にレプリケートできます。

次の図は、システム管理シャーディングとともにData Guardレプリケーションが使用される方法を示しています。図に示す例には、プライマリ・シャードグループのShardgroup 1、および2つのスタンバイ・シャードグループのShardgroup 2とShardgroup 3があります。Shardgroup 1はData Guardプライマリ・データベース(シャード1-3)から構成されます。Shardgroup 2は、同じデータ・センターに存在し、同期レプリケーション用に構成されたローカル・スタンバイ・データベース(シャード4-6)から構成されます。Shardgroup 3は、異なるデータ・センターに存在し、非同期レプリケーション用に構成されたリモート・スタンバイ(シャード7-9)から構成されます。この構成ではOracle Active Data Guardが有効なため、各スタンバイが読取り専用でオープンされています。

図2-11 Data Guardレプリケーションを使用するシステム管理のシャーディング



レプリケーションの論理単位としてのシャードグループという概念によって、レプリケーションの実装の詳細がユーザーには隠されて

います。Data Guardの場合、レプリケーションはシャード(データベース)のレベルで実行されます。前の図のシャード・データベースは、レプリケートされた3つのシャードのセット({1, 4, 7}、{2, 5, 8}および{3, 6, 9})で構成されています。レプリケートされたシャードの各セットは、ファスト・スタート・フェイルオーバー(FSFO)が有効なData Guard Broker構成として管理されます。

レプリケーションをデプロイするには、シャードグループのプロパティ(リージョン、ロールなど)を指定し、そこにシャードを追加します。Oracle ShardingによってData Guardが自動的に構成され、レプリケートされたシャードの各セットに対してFSFOオブザーバが起動されます。さらに、読取り専用ワークロード、ロール・ベースのグローバル・サービス、レプリケーション・ラグのロード・バランシング、および地域ベースのルーティングが提供されます。

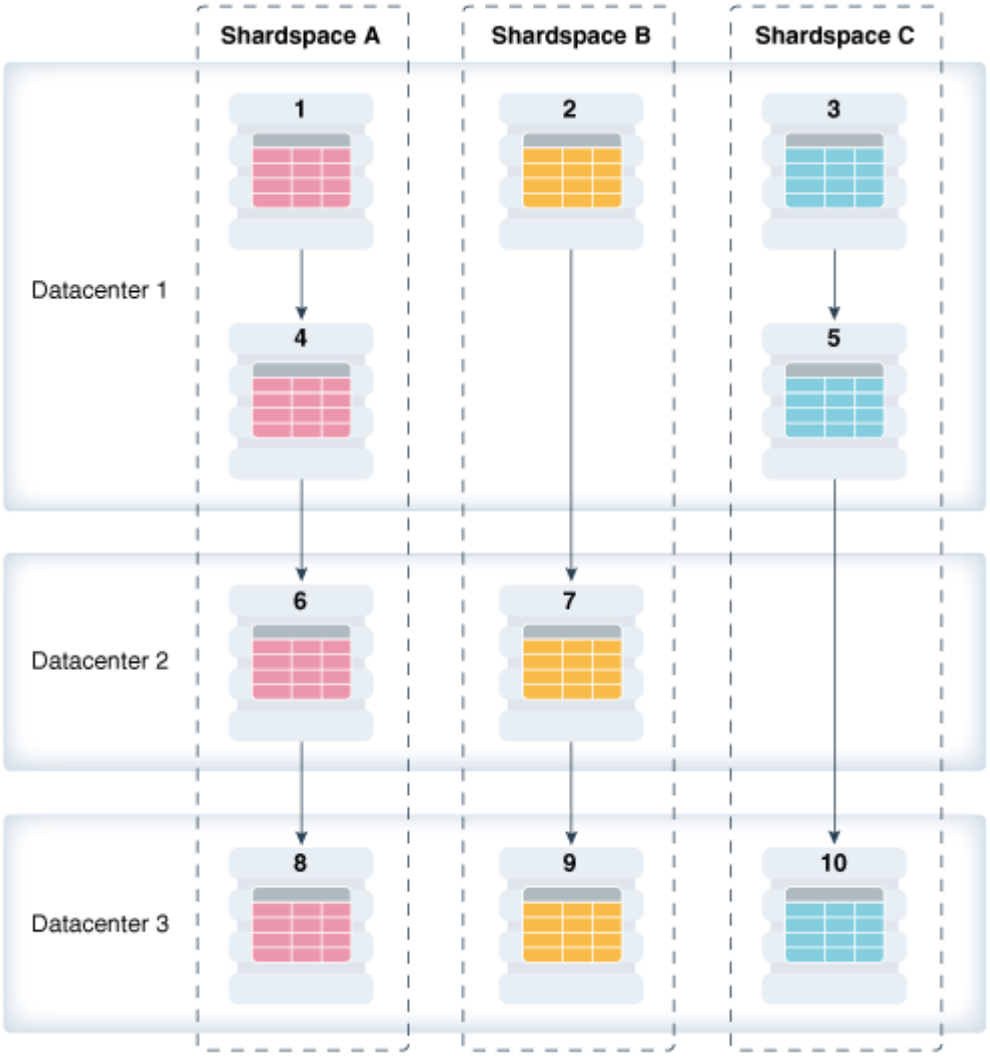
次のGDSCTLコマンドを実行して、前の図に示されている構成例をデプロイします。

```
CREATE SHARDCATALOG -database host00:1521:shardcat -region dc1,dc2
ADD GSM -gsm gsm1 -listener 1571 -catalog host00:1521:shardcat -region dc1
ADD GSM -gsm gsm2 -listener 1571 -catalog host00:1521:shardcat -region dc2
START GSM -gsm gsm1
START GSM -gsm gsm2
ADD SHARDGROUP -shardgroup shardgroup1 -region dc1 -deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup2 -region dc1 -deploy_as active_standby
ADD SHARDGROUP -shardgroup shardgroup3 -region dc2 -deploy_as active_standby
CREATE SHARD -shardgroup shardgroup1 -destination host01 -credential oracle_cred
CREATE SHARD -shardgroup shardgroup1 -destination host02 -credential oracle_cred
CREATE SHARD -shardgroup shardgroup1 -destination host03 -credential oracle_cred
...
CREATE SHARD -shardgroup shardgroup3  -destination host09 -credential oracle_cred
DEPLOY
```

ユーザー定義のシャード・データベースでのOracle Data Guardの使用

ユーザー定義のシャーディングの場合、レプリケーションの論理(および物理)単位はシャードです。シャードはシャードグループに統合されません。各シャードとそのレプリカがシャード領域を構成し、これが1つのData Guard Broker構成に対応します。シャードスペースごとに個別にレプリケーションを構成できます。異なるデータ・センターに存在する異なる数のスタンバイをシャード領域に含めることができます。Data Guardレプリケーションを使用するユーザー定義シャーディングの例を次の図に示します。

図2-12 Data Guardレプリケーションを使用するユーザー定義のシャーディング



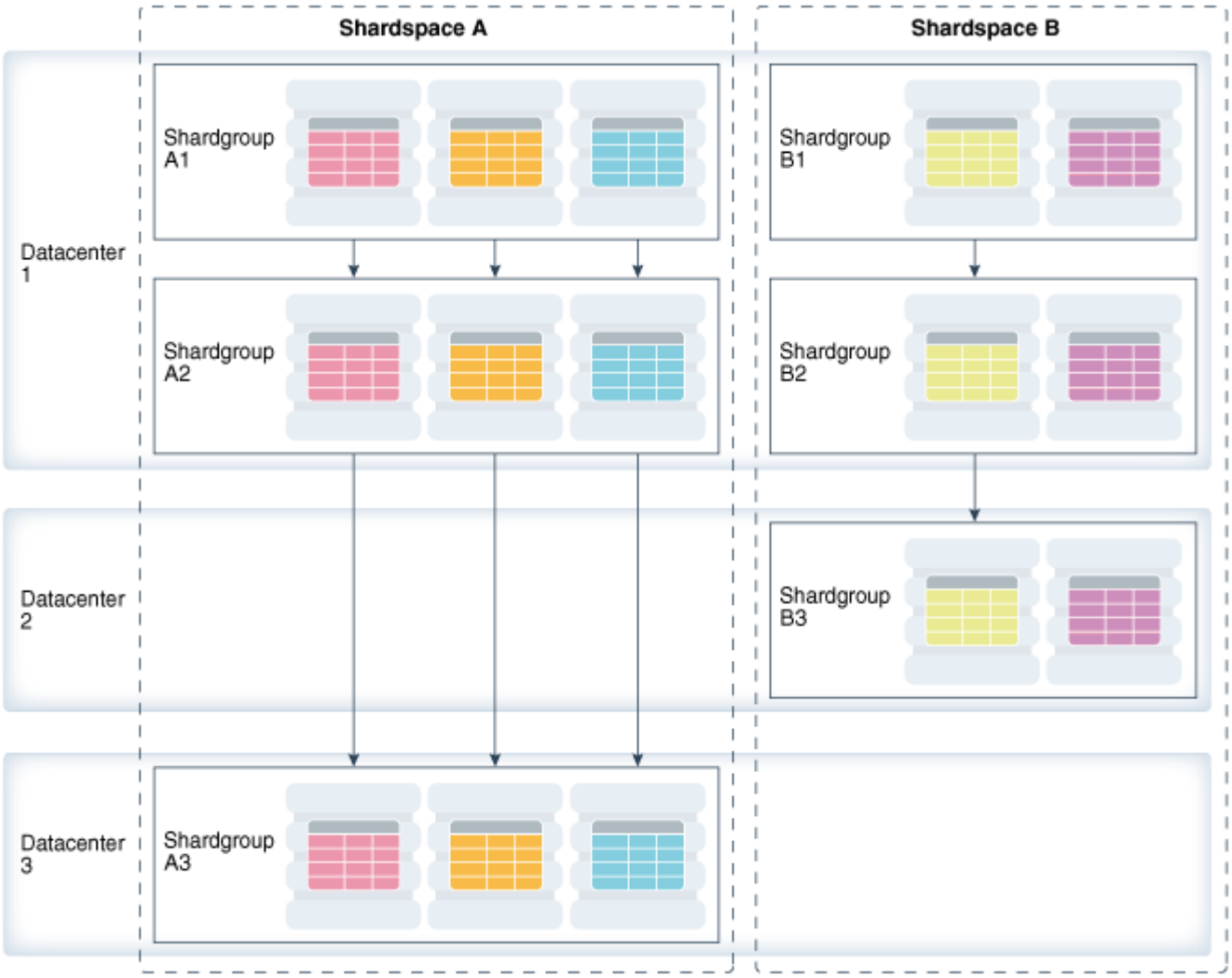
前の図に示されているData Guardレプリケーションを使用するユーザー定義シャード・データベースの例をデプロイするには、次のGDSCTLコマンドを実行します。

```
CREATE SHARDCATALOG -sharding user -database host00:1521:cat -region dc1,dc2,dc3
ADD GSM -gsm gsm1 -listener 1571 -catalog host00:1521:cat -region dc1
ADD GSM -gsm gsm2 -listener 1571 -catalog host00:1521:cat -region dc2
ADD GSM -gsm gsm3 -listener 1571 -catalog host00:1521:cat -region dc3
START GSM -gsm gsm1
START GSM -gsm gsm2
START GSM -gsm gsm3
ADD SHARDSPACE -shardspace shardspace_a
ADD SHARDSPACE -shardspace shardspace_b
ADD SHARDSPACE -shardspace shardspace_c
CREATE SHARD -shardspace shardspace_a -region dc1 -deploy_as primary  -destination
host01 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rsp
CREATE SHARD -shardspace shardspace_a -region dc1 -deploy_as standby  -destination
host04 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rsp
CREATE SHARD -shardspace shardspace_a -region dc2 -deploy_as standby  -destination
host06 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rsp
CREATE SHARD -shardspace shardspace_a -region dc3 -deploy_as standby  -destination
host08 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rsp
CREATE SHARD -shardspace shardspace_b -region dc1 -deploy_as primary  -destination
host08 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rs
...
CREATE SHARD -shardspace shardspace_c -region dc3 -deploy_as standby  -destination
host10 -credential oracle_cred -netparamfile /home/oracle/netca_dbhome.rsp
```

コンポジット・シャード・データベースでのOracle Data Guardの使用

コンポジット・シャーディングの場合、ユーザー定義のシャーディングと同様にシャード・データベースが複数のシャード領域から構成されます。ただし、各シャード領域には、レプリケートされたシャードではなくレプリケートされたシャードグループが含まれます。

図2-13 複合シャーディングとData Guardレプリケーション



次のGDSCTLコマンドを実行して、前の図に示されている構成例をデプロイします。

```
CREATE SHARDCATALOG -sharding composite -database host00:1521:cat -region dc1,dc2,dc3
ADD GSM -gsm gsm1 -listener 1571 -catalog host00:1521:cat -region dc1
ADD GSM -gsm gsm2 -listener 1571 -catalog host00:1521:cat -region dc2
ADD GSM -gsm gsm3 -listener 1571 -catalog host00:1521:cat -region dc3
START GSM -gsm gsm1
START GSM -gsm gsm2
START GSM -gsm gsm3
ADD SHARDSPACE -shardspace shardspace_a
ADD SHARDSPACE -shardspace shardspace_b
ADD SHARDGROUP -shardgroup shardgroup_a1 -shardspace shardspace_a -region dc1
-deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup_a2 -shardspace shardspace_a -region dc1
-deploy_as active_standby
ADD SHARDGROUP -shardgroup shardgroup_a3 -shardspace shardspace_a -region dc3
-deploy_as active_standby
```

```
ADD SHARDGROUP -shardgroup shardgroup_b1 -shardspace shardspace_b -region dc1
-deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup_b2 -shardspace shardspace_b -region dc1
-deploy_as active_standby
ADD SHARDGROUP -shardgroup shardgroup_b3 -shardspace shardspace_b -region dc2
-deploy_as active_standby
CREATE SHARD -shardgroup shardgroup_a1 -destination host01 -credential orcl_cred
...
CREATE SHARD -shardgroup shardgroup_b3 -destination host09 -credential orcl_cred
DEPLOY
```

シャード・データベースでのOracle GoldenGateの使用

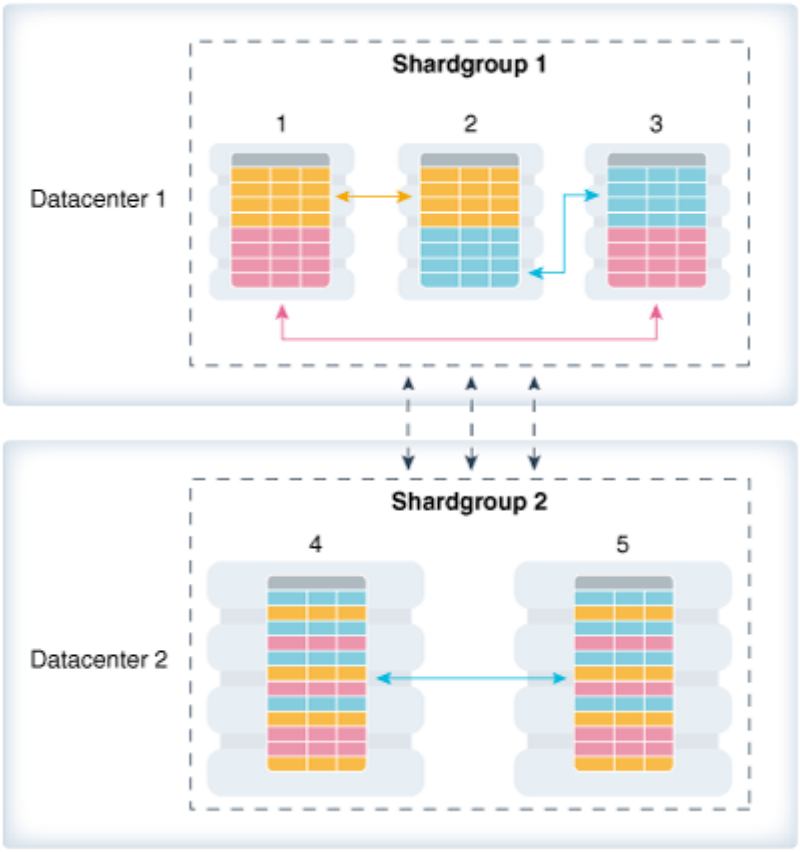
Oracle GoldenGateは、すべてのシャードが書込み可能で、各シャードをシャードグループ内の別のシャードに部分的にレプリケートできるファイングレイン・アクティブ-アクティブ・レプリケーションのために使用されます。

ノート:

Oracle Database 21c では、マルチテナント・アーキテクチャ(CDB)のみがサポートされます。Oracle GoldenGate バージョン 12.3-19.1 では、単一インスタンスの Oracle データベース(リリース 11g から 19c)を使用した Oracle Sharding のみがサポートされます。

Oracle GoldenGateでは、レプリケーションがチャンク・レベルで処理されます。たとえば、次の図のShardgroup 1では、各シャードに格納されているデータの半分が1つのシャードにレプリケートされ、残りの半分が別のシャードにレプリケートされます。いずれかのシャードが使用できなくなった場合、そのワークロードはシャードグループ内の他の2つのシャードに分割されます。フェイルオーバー先が複数存在し、障害が発生したシャードのすべてのワークロードを1つのシャードで処理する必要がないため、シャード障害の影響が軽減されます。

図2-14 GoldenGateレプリケーションを使用するシステム管理シャーディング



Oracle GoldenGateレプリケーションでは、シャードグループにシャード表内の各行の複数のレプリカを含めることができるため、シャードグループ内に高可用性が提供され、Data Guardレプリケーションの場合のようにシャードグループのローカル・レプリカを作成する必要はありません。シャードグループ内で各行がレプリケートされる回数は、シャードグループのレプリケーション・ファクタと呼ばれ、構成可能なパラメータです。

障害時リカバリ機能を提供するため、シャードグループを1つ以上のデータ・センターにレプリケートできます。シャードグループの各レプリカは、異なる数のシャード、レプリケーション・ファクタ、データベース・バージョンおよびハードウェア・プラットフォームを持つことができます。ただし、レプリケーションはチャンク・レベルで実行されるため、すべてのシャードグループ・レプリカが同じ数のチャンクを持つ必要があります。

前の図のShardgroup 2は、Shardgroup 1と同じデータを含んでいますが、異なるデータ・センターにあります。どちらのデータ・センターのシャードも書込み可能です。どちらのシャードグループでもデフォルトのレプリケーション・ファクタ(2)が使用されます。

Shardgroup 2には2つのシャードのみが含まれており、レプリケーション・ファクタが2であるため、シャードは完全にレプリケートされており、シャード・データベースに格納されているすべてのデータが各シャードに含まれています。これは、シャード間を移動せずにこれらのシャードにルーティングされた問合せを実行できることを意味します。このシャードグループにはフェイルオーバー先が1つしかないため、シャードが停止した場合、他のシャードへの負荷が2倍になります。

Oracle Shardingは、異なるシャードの同じ行に対して実行される競合更新の数が最小限になるように設計されています。これは、ハッシュ値の範囲ごとにマスター・チャンクを指定し、対応するデータのほとんどのリクエストをこのチャンクにルーティングすることで実現されます。

状態遷移(チャンクの移動や分割、シャードの起動と停止など)のために、更新の競合を避けられない場合があります。トランザクションの待機時間を最小限にするため、ユーザーが意図的に競合を発生させる場合もあります。このような場合、Oracle GoldenGateは挿入と削除の競合を含むすべての種類の競合を処理する自動的な競合検出および解決機能を提供します。

ノート:

Oracle GoldenGate では、ユーザー定義のシャーディング方法をサポートしていません。

Oracle GoldenGate を使用するシステム管理のシャーディングの場合、シャードには 2 つ以上のチャンクが必要となります。

Oracle GoldenGate は、PDB をシャードとしてサポートしません。

関連項目:

Fusion Middleware*Oracle GoldenGate*マイクロサービス・アーキテクチャの使用ガイドのOracle GoldenGateシャーディングの使用(Oracle ShardingでのOracle GoldenGateの使用の詳細)

問合せ処理と問合せコーディネータ

問合せコーディネータはシャード・カタログの一部です。問合せコーディネータは、シャード・データベースに対する問合せ処理サポートを提供します。シャード・カタログのシャード・データベース・トポロジ・メタデータにアクセスすると、問合せコーディネータが重要な役割を果たす3つの一般的なケースがあります。

1. シャーディング・キーのない単一シャード問合せ

   シャーディング・キーがアプリケーションから渡されない場合、問合せコーディネータは、問合せに必要なデータを含むシャードを特定し、そこに問合せを送信して実行します。

2. マルチシャード問合せ

   問合せコーディネータは、SELECT COUNT(*) FROM Customerなどの**マルチシャード問合せ**と呼ばれる複数のシャードからのデータを必要とする問合せにも役立ちます。

3. 集計問合せ

   問合せコーディネータは、売上データの集計など、レポートで通常使用される集計問合せを処理します。

いずれの場合も、問合せコーディネータのSQLコンパイラは関連するシャードを自動的に識別し、関係するすべてのシャード間で問合せの実行を調整します。

単一シャード問合せのシナリオでは、問合せ全体が単一の参加シャードで実行され、問合せコーディネータは処理された行をクライアントに戻します。

マルチシャード問合せの場合、SQLコンパイラは問合せを分析し、参加シャードによって送信および実行される問合せフラグメントにリライトします。問合せは、関与するシャードでほとんどの問合せ処理が行われて、コーディネータによって集計されるようにリライトされます。

問合せコーディネータは、Oracle Databaseのパラレル問合せエンジンを使用して、マルチシャード問合せをシャードにパラレルに最適化およびプッシュします。各シャードは、保持するデータのサブセットに対して問合せを実行します。次に、結果が問合せコーディネータに戻され、問合せコーディネータがクライアントに返されます。

要するに、シャードは問合せコーディネータによって実行される問合せの計算ノードとして機能します。計算がデータのあるシャードにプッシュされるため、シャードとコーディネータの間のデータの移動が減少します。この仕組みによって、問合せコーディネータの処理負荷ができるだけ多くのシャードに移動するため、リソースを効率的に使用できるようになります。

一貫性レベルの指定

マルチシャード問合せでは、様々な一貫性レベルを指定できます。たとえば、一部の問合せでシャード間のSCN同期のコストを回避する必要がある場合は、それらのシャードをグローバルに分散できます。別のユース・ケースとして、レプリケーション用のスタンバイを使用している場合は、プライマリとそのスタンバイから結果がフェッチされる可能性があるため、マルチシャード問合せで少し古いデータが許容されます。マルチシャード問合せでは、すべてのシャードで最も大きい共通SCNで問合せを発行することによって、グローバルな読込み一貫性(CR)を維持する必要があります。

高可用性およびパフォーマンス

ファスト・スタート・フェイルオーバーを有効にして最大可用性保護モード(データ損失のないフェイルオーバー)でOracle Data Guardを使用して問合せコーディネータを保護することをお薦めします。問合せコーディネータはさらなる可用性およびスケーラビリティのためにオプションでOracle RAC対応にすることができます。マルチシャード問合せワークロードのスケーラビリティおよび可用性を向上させるために、読取り専用モードのOracle Active Data Guardスタンバイ・シャード・カタログ・データベースをマルチシャード問合せコーディネータとして機能させることができます。

集計ユースケースおよびシャーディング・キーを使用しないSQL実行では、直接のキーベースのルーティングに比べてパフォーマンス・レベルが低下します。

クライアント・アプリケーション・リクエストのルーティング

クライアント・アプリケーション・リクエストをシャードに直接ルーティングするには、Oracleドライバを使用してシャードに接続し、リクエストとともにシャーディング・キーを指定します。

シャーディング・キーについて

高いパフォーマンスおよび障害分離を必要とするすべてのデータベース・リクエストは、単一の値のシャーディング・キーに関連付けられているデータにのみアクセスする必要があります。データベース接続を確立するときに、アプリケーションはシャーディング・キーを渡す必要があります。これに該当する場合、リクエストは適切なシャードに直接ルーティングされます。

複数のリクエストは、それらがすべて同じシャーディング・キーに関連しているかぎり、同じセッションで実行できます。通常、そのようなトランザクションは数十行または数百行にアクセスします。単一シャードのトランザクションの例としては、オーダー入力、顧客の請求書レコードの検索と更新、およびサブスクライバのドキュメントの検索と更新があります。

複数の値のシャーディング・キーに関連付けられているデータ、またはシャーディング・キーの値が不明なデータにアクセスする必要があるデータベース・リクエストは、複数のシャードでの問合せのパラレル実行を調整する問合せコーディネータから実行する必要があります。

Oracle接続ドライバについて

実行時に、接続プールは、プールされた接続間でデータベース・リクエストをルーティングすることでシャード・ディレクタとして機能し

ます。Oracle Databaseは、データ・アクセス・ドライバ(OCI、JDBC、ODP.NETなど)の接続プーリングをサポートしています。これらのドライバは、接続リクエストの一部として指定されたシャーディング・キーを認識できます。同様に、JDBCクライアントのOracle Universal Connection Pool (UCP)は、接続URLで指定されたシャーディング・キーを認識できます。また、Oracle UCPを使用すると、非Oracleアプリケーション・クライアント(Apache Tomcat、WebSphereなど)は、Oracle Shardingと連携できます。

OracleクライアントはUCPキャッシュのルーティング情報を使用し、アプリケーションから渡されたシャーディング・キーに基づいて、データベース・リクエストを適切なシャードに直接ルーティングします。データベース・リクエストをそのようにデータ依存でルーティングすることにより、余分なネットワークのホップが排除され、ボリュームの大きいアプリケーションのトランザクション待機時間が減少します。

ルーティング情報は、シャード・ディレクタを使用して確立されるシャードへの最初の接続時にキャッシュされます。キャッシュされた範囲内のシャーディング・キーに対する後続のデータベース・リクエストは、シャード・ディレクタをバイパスしてシャードに直接ルーティングされます。

UCPと同様に、シャード・ディレクタは接続文字列に指定されたシャーディング・キーを処理して、ルーティング情報をキャッシュできます。ただし、UCPはすでに確立されている接続を使用してデータベース・リクエストをルーティングしますが、シャード・ディレクタは接続リクエストをシャードにルーティングします。シャードが使用できなくなった場合、またはシャーディング・トポロジに変更が発生した場合、ルーティング・キャッシュは自動的にリフレッシュされます。高パフォーマンスのデータ依存のルーティングにするには、シャード・データベースのデータへのアクセス時に接続プールを使用することをお薦めします。

問合せコーディネータを介した直接ルーティングおよびルーティング・リクエストには、個別の接続プールを使用する必要があります。直接ルーティングの場合は、読取り書込みワークロードおよび読取り専用ワークロードのために、個別のグローバル・サービスを作成する必要があります。これは、Data Guardレプリケーションが使用されている場合にのみ当てはまります。プロキシ・ルーティングの場合は、シャード・カタログ・データベースでGDS$CATALOGサービスを使用します。

シャード・データベースの管理インタフェース

GDSCTLコマンドライン・ユーティリティは、Oracle Shardingシャード・データベースを構成、デプロイ、監視および管理するために使用します。Oracle Enterprise Manager Cloud Controlは、シャード・データベースの監視および管理にも使用できます。

SQL*Plusと同様に、GDSCTLは、シャード・データベースのライフ・サイクルのすべてのステージを制御できるコマンドライン・ユーティリティです。GDSCTLを別のサーバーまたはラップトップからリモートで実行してシャード・データベース・トポロジを構成およびデプロイし、シャード・データベースをモチエータおよび管理できます。

GDSCTLは、シャード・データベースの構成を指定し、そのデプロイメントを自動化する簡単な宣言的方法を提供します。わずかなGDSCTLコマンドを使用するだけで、シャード・データベースを作成できます。

グラフィカル・ユーザー・インタフェースが必要な場合は、シャード・データベースの監視およびライフ・サイクル管理にCloud Controlを使用することもできます。Cloud Controlを使用すると、可用性およびパフォーマンスを監視でき、シャード、サービス、シャード・ディレクタおよびその他のシャーディング・コンポーネントの追加およびデプロイなどのシャーディング構成を変更できます。

3 シャード・データベースのデプロイ

シャード・データベースのデプロイでは、必要なソフトウェア・コンポーネントのインストール、カタログ、ロール、シャード・データベースの作成、高可用性のためのレプリケーションの構成、およびシャード・データベースのスキーマの作成のための前提条件と手順について説明します。

次の各項で、シャード・データベースのデプロイに必要な概念とタスクについて説明します。

シャード・データベースのデプロイの概要

Oracle Shardingには、シャード・データベースを自動的にデプロイする機能があり、これにはシャードとレプリカの両方が含まれます。

シャード・データベース管理者は、トポロジ(リージョン、シャード・ホスト、レプリケーション・テクノロジ)を定義し、GDSCTLコマンドライン・インタフェースを使用して宣言的に指定することでDEPLOYコマンドを起動します。

始める前に

シャード・データベースには、多種多様な構成とトポロジが使用できる点に注意してください。目的とするシャード・データベースには、多様なOracleソフトウェア・コンポーネント(Oracle Data Guard、Oracle GoldenGate、Oracle Real Application Clusters (Oracle RAC)など)と、各種のシャーディング手法(システム管理シャーディング、コンポジット・シャーディング、ユーザー定義シャーディング)を採用することがあります。

アプリケーションに特有のアーキテクチャとシステム要件に応じて、システムの設計時に複数の選択肢から選択できることもあります。様々なシャーディング手法と障害回復および高可用性のオプションの詳細は、「シャーディング方法」と「シャード・レベルの高可用性」を参照してください。

シャード作成メソッドの選択

シャード構成をデプロイするとき、シャードの追加に使用できる2つの異なるGDSCTLコマンド(ADD SHARDおよびCREATE SHARD)があります。

シャーディング・トポロジの構成を開始する前に、使用するシャード作成メソッドを決定します。これはこの決定が構成ステップの一部に影響するためです。

ADD SHARDメソッドとCREATE SHARDメソッドの違いは、構成手順で必要に応じて説明します。

ADD SHARDメソッド

GDSCTL ADD SHARDコマンドは、Oracle Sharding構成にシャードを追加する場合に使用できます。このコマンドを使用する場合、デプロイメント時にシャードになるOracleデータベースを作成するのはユーザーの役割です。データベースがOracle Sharding構成に含めるための前提条件を満たすかぎり、どのメソッドを使用してデータベースを作成しても構いません。

ADD SHARDメソッドを使用する利点のいくつかを次に示します。

- データベースの作成に使用するプロセスを完全に制御できます。

- データベース・パラメータ、ネーミングおよび記憶域の場所は簡単にカスタマイズできます。
- PDBシャードと非CDBシャードの両方がサポートされます。
- シャード・ホストで構成するOracleソフトウェアが少なくなります。
- GDSCTLコマンドを実行する前にシャード・データベースが作成されるため、デプロイメント・プロセスはそれほど複雑ではありません。

CREATE SHARDメソッド

GDSCTL CREATE SHARDコマンドは、Oracle Sharding構成でシャードを作成する場合に使用できます。CREATE SHARDを使用すると、シャード・カタログはOracleリモート・スケジューラ・エージェントを利用して、各シャード・ホスト上でDatabase Configuration Assistant (DBCA)をリモートで実行し、データベースを作成します。このメソッドではPDBがサポートされないため、追加されるシャード・データベースは非CDBである必要があります。

CREATE SHARDメソッドを使用する利点のいくつかを次に示します。

- データベース管理者以外のためのシャード・データベースを作成しやすくなります。
- 現行方式で標準がない場合、新しいデータベースをプロビジョニングする標準的な方法が提供されます。
- CREATE SHARDで作成されたデータベースは、データベースに対してSQL文を実行したり、データベース・パラメータを調整しなくても、Oracle Sharding用に自動的に正しく構成されます。
- スタンバイ・データベースを自動的に作成できます。

シャード・データベースのデプロイのロードマップ

このロードマップに従って、ホストの設定、必要なソフトウェアのインストールおよびシャード・データベースの構成とデプロイを行います。

大まかなデプロイメント・ステップは次のとおりです。

1. コンポーネントを設定します。
    - 選択したシャーディング構成およびトポロジに必要になるホストをプロビジョニングおよび構成します(「ホストおよびオペレーティング・システムのプロビジョニングと構成」を参照)。
    - 選択したカタログとシャード・ノードにOracle Databaseソフトウェアをインストールします(「Oracle Databaseソフトウェアのインストール」を参照)。
    - シャード・ディレクタ・ノードにグローバル・サービス・マネージャ(GSM)ソフトウェアをインストールします(「シャード・ディレクタ・ソフトウェアのインストール」を参照)。
2. シャーディング・メタデータとアプリケーション・データの保管に必要なデータベースを作成します。
    - シャード・カタログにするデータベースと、障害回復(DR)と高可用性(HA)に必要なレプリカを作成します(「シャード・カタログ・データベースの作成」を参照)。
    - ADD SHARDメソッドを使用してシャードをデプロイする場合は、DRおよびHAに必要なスタンバイ・データベースが含まれる構成で、シャードになるデータベースを作成します(「シャード・データベースの作成」を参照)。
3. GDSCTLコマンドライン・ユーティリティから、次のコマンドの一部またはすべてを使用してシャーディング・トポロジを指定します(「シャード・データベース・トポロジの構成」を参照)。
    - CREATE SHARDCATALOG
    - ADD GSM
    - START GSM
    - ADD SHARDSPACE
    - ADD SHARDGROUP
    - ADD CDB

- ADD SHARD
- ADD CREDENTIAL
- ADD FILE
- CREATE SHARD
- ADD INVITEDNODE

4. DEPLOYを実行して、シャーディング・トポロジ構成をデプロイします(「シャーディング構成のデプロイ」を参照)。
5. シャード・データベース内のシャードへのアクセスに必要なグローバル・サービスを追加します(「グローバル・データベース・サービスの作成と開始」を参照)。
6. 各シャードのステータスを確認します(「シャード・ステータスの確認」)。

シャード・データベース構成のデプロイが正常に完了すると、アプリケーションに必要なシャード・スキーマ・オブジェクトを作成できます。シャード・データベースのスキーマ・オブジェクトを参照してください。

次のトピックでは、それぞれのデプロイメント・タスクについて、システムの各種コンポーネントに固有の要件とともに詳細に説明します。これらのトピックは、プロセスの各ステップごとの設定と構成についてのリファレンスとして利用できます。ただし、それらのみでは完全に機能するシャーディング構成は生成されません。これは、完全なシャーディング・シナリオを実装するのではなく、各ステップの要件のみを示しているためです。

「シャード・データベースのデプロイの例」では、典型的な参照構成の具体的なデプロイメント・シナリオについて説明しています。この項では、すべてのステップの完了後に、完全に機能するシャード・データベースを生成するために必要なすべてのコマンド例を示します。

ホストおよびオペレーティング・システムのプロビジョニングと構成

ソフトウェアをインストールする前に、Oracle Shardingのハードウェア、ネットワークおよびオペレーティング・システムの要件を確認してください。

- Oracle Database Enterprise Editionは、Oracleシャード・データベースの実行時に必要になります。
- シャードのハードウェア要件とオペレーティング・システム要件は、Oracle Databaseの要件と同じです。これらの要件の詳細は、Oracle Databaseのインストール・ドキュメントを参照してください。
- シャード・カタログおよびシャード・ディレクタのハードウェア要件とオペレーティング・システム要件は、グローバル・データ・サービス・カタログおよびグローバル・サービス・マネージャの要件と同じです。これらの要件の詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください。
- ネットワーク要件は低遅延GigEです。
- ポート通信の要件は次のとおりです。
    - すべてのシャードがすべてのシャード・ディレクタのリスナーとONSポートに到達できる必要があります。シャード・ディレクタのリスナー・ポートおよびONSポートが、アプリケーション/クライアント層、すべてのシャード、シャード・カタログおよび他のすべてのシャード・ディレクタに対して開かれている必要があります。

      シャード・ディレクタのデフォルトのリスナー・ポートは、1522です。デフォルトのONSポートは、ほとんどのプラットフォームで6123 (ローカルONS)および6234 (リモートONS)です。
    - すべてのシャードがシャード・カタログ(プライマリとスタンバイの両方)のTNSリスナー・ポート(デフォルトは1521)に到達できる必要があります。
    - 各シャードのTNSリスナー・ポートがすべてのシャード・ディレクタおよびシャード・カタログに対して開かれている

必要があります。

- 前述したポート番号のすべては、デプロイメント構成時に変更できます。ただし、使用するポート番号は、ホスト・ソフトウェアの設定前に決定しておく必要があります。

- ホスト名の解決は、すべてのシャード・カタログ、シャード、およびシャード・ディレクタ・ホストで成功する必要があります。オペレーティング・システムのコマンド(pingなど)は、シャード・データベース構成コマンドで提示されたホスト名の指定時に、特定のホストから別のホストに向けて成功する必要があります。

ホスト・システムの数とサイズ設定

目的とする構成によっては、次のホストも必要になることがあります。

- シャード・カタログ・ホスト。シャード・カタログ・ホストでは、シャード・カタログとして機能するOracle Databaseを実行します。このデータベースには、少量のシャーディング・トポロジ・メタデータと、アプリケーション用に作成した重複表を格納します。また、このデータベースは、シャーディングに対応していないアプリケーションのクロスシャード問合せおよびサービスの接続のための問合せコーディネータとしても機能します。一般に、このデータベースのトランザクション・ワークロードとサイズは、特に大きなものにはなりません。

- シャード・カタログ・データベースのスタンバイ(レプリカ)。プライマリ・シャード・カタログ・データベースのレプリカまたはスタンバイは、少なくとも1つ以上のホストに格納することをお薦めします。このホストは、プライマリ・カタログ・ホストの障害発生時に必要になります。また、このホストは、スタンバイ・データベースとして機能すると同時に、クロスシャード問合せの問合せコーディネータになるように構成することもできます。

- シャード・ディレクタ・ホスト。シャード・ディレクタ(グローバル・サービス・マネージャ)ソフトウェアは、個別のホストに配置することも、シャード・カタログと同じホストに配置することもできます。このシャーディング・システムのコンポーネントは、シャード構成の監視と構成に使用するネットワーク・リスナーと複数のバックグラウンド・プロセスで構成されます。シャード・ディレクタをカタログ・データベースと同じホストに配置する場合は、カタログ・データベースとは別のOracleホームにインストールする必要があります。これは、インストール・パッケージがOracle Databaseに使用するものと異なるためです。

- 複数のシャード・ディレクタ。高可用性のために、シャード・システム内で複数のシャード・ディレクタを実行することをお薦めします。追加のシャード・ディレクタは、専用のホストで実行することも、スタンバイ・シャード・カタログ・データベースを実行するホストで実行することもできます。

- シャード。前述のホストに加えて、システム内で構成される各シャードは、それぞれ個別のホストで実行することも必要になります。このタスク用に選択したホストとその構成では、一般的なOracle Databaseホストと同じ方法で、それぞれのシャードにかかる負荷に応じたサイズを設定する必要があります。

- シャード・スタンバイ(レプリカ)。前述したように、高可用性と障害回復のために、レプリケーション・テクノロジ(Oracle Data GuardやOracle Golden Gateなど)を使用する必要があり、すべてのシャード・データのレプリカを作成します。追加のホストは、こうしたレプリカまたはスタンバイ・データベースを実行するために必要になります。

  ノート:

  Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。

ホストの数と各ホストの容量要件を決定したら、選択した手法を使用して環境に適したハードウェア・リソースをプロビジョニングします。ソフトウェアをインストールする前に、それぞれのホストが前述したポートを通じて相互に通信できることを確認します。シャーディング構成は、本質的に分散システムであるため、デプロイメント・プロセスの次のステップに進む前に、このホスト間およ

びホスト全体の接続を確認しておくことが重要です。ポート・アクセスが適切に設定されていないと、今後のコマンドのエラーの原因になります。

Oracle Databaseソフトウェアのインストール

シャード・カタログ、データベース・シャードまたはレプリカをホストする各システムにOracle Databaseをインストールします。

Oracle Sharding構成内のシャード・カタログとすべてのシャードにはOracle Database Enterprise Editionが必須という要件がありますが、インストールとすべてのインストール後スクリプトの実行が成功していれば、それ以外に必要になる特別なインストールの考慮事項はありません。

オペレーティング・システム・ユーザーの構成の詳細は、対象プラットフォームのインストレーション・ガイド(https://docs.oracle.com/en/database/oracle/oracle-database/)を参照してください。

CREATE SHARDメソッドを使用してシャードを構成に追加する場合は、各シャード・ホストにリモート・スケジューラ・エージェント・ソフトウェアもインストールする必要があります。エージェントは、シャード・カタログ・ホストにインストールする必要はありません。手順は、リモート・ホストでのスケジューラ・エージェントのインストールと構成に関する項を参照してください。

また、CREATE SHARDメソッドでは、各シャード・ホストに2つのディレクトリ($ORACLE_BASE/oradataおよび$ORACLE_BASE/fast_recovery_area)を作成する必要があります。Oracle Databaseソフトウェアの所有者としてシャード・ホストにログインしている間に、これら2つのディレクトリを作成します。権限は、シャード・データベースのデータ・ファイルを保持するディレクトリと同じに設定する必要があり、通常、ソフトウェア所有者のみがフル・アクセスできます。

シャード・ディレクタ・ソフトウェアのインストール

シャード・ディレクタをホストする各システムにグローバル・サービス・マネージャ・ソフトウェアをインストールします。

このソフトウェアのインストールは、Oracle Databaseインストールとは異なる点に注意してください。シャード・ディレクタ・ソフトウェアをシャード・カタログ・データベースと同じホストに配置することにした場合は、個別のOracleホームにインストールする必要があります。

グローバル・サービス・マネージャ・ソフトウェアのインストールの詳細は、『Oracle Database Global Data Services概要および管理ガイド』を参照してください。

シャード・カタログ・データベースの作成

次の情報とガイドラインを使用して、シャード・カタログ・データベースを作成してください。

シャード・カタログ・データベースには、少量のシャーディング・トポロジ・メタデータと、シャード・アプリケーションで使用するために作成するすべての重複表を格納します。カタログ・データベースは、複数のシャードからデータを選択して集計するクロスシャード問合せを実行するための問合せコーディネータとしても機能します。

シャーディングの観点では、カタログ・データベースの作成方法やプロビジョニング方法は重要ではありません。このデータベースは、Database Configuration Assistant (DBCA)で作成することも、SQL*Plusを使用して手動で作成することも、クラウド・インフラストラクチャ・ツールからプロビジョニングすることもできます。

次の特性を備えたシャード・カタログ・ホストでOracle Database Enterprise Editionインスタンスを実行していれば、シャード・カタログとして使用できます。

- シャード・カタログ・データベースとして使用するレガシー・データベースまたはプラガブル・データベース(PDB)を作成します。コンテナ・データベース(CDB)のルート・コンテナ(CDB$ROOT)をシャード・カタログ・データベースとして使用することは、サポートされていません。
- シャード・カタログ・データベースでは、サーバー・パラメータ・ファイル(SPFILE)を使用する必要があります。これが必要になる理由は、シャーディング・インフラストラクチャが内部データベース・パラメータを使用して構成メタデータを保存し、そのデータはデータベースの起動操作と停止操作の間で永続している必要があるためです。

```
$ sqlplus / as sysdba
SQL> show parameter spfile
NAME     TYPE      VALUE
-------- --------- ------------------------------------
spfile   string    /u01/app/oracle/dbs/spfilecat.ora
```

- データベース文字セットと各国語文字セットは、すべてのシャード・データベースで使用されるため、同じにする必要があります。つまり、シャード・カタログまたはシャードのいずれかに挿入される可能性のあるすべての文字が含まれている文字セットを選択する必要があるということです。

  この要件は、MOVE CHUNKコマンドのシャーディング時に、トランスポータブル表領域をシャード間で移動するためにOracle Data Pumpが内部的に使用されることから発生します。このメカニズムの要件は、ソースと宛先で文字セットが一致していることです。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=catalog_pdb_name;
SQL> select * from nls_database_parameters
  2  where parameter like '%CHARACTERSET';
PARAMETER                                VALUE
---------------------------------------- ----------------------
NLS_NCHAR_CHARACTERSET                   AL16UTF16
NLS_CHARACTERSET                         WE8DEC
```

- シャード・カタログ・データベースは、データベース・リンクによってシャードに接続するマルチシャード問合せを実行できるため、データベース初期化パラメータ OPEN_LINKSおよびOPEN_LINKS_PER_INSTANCEの値は、シャード・データベース構成に含まれるシャードの数以上にする必要があります。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=catalog_pdb_name;
SQL> show parameter open_links
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
open_links                           integer     20
open_links_per_instance              integer     20
```

- データベース初期化パラメータDB_FILESは、システム内のチャンクや表領域の合計数以上に設定します。

  シャーディング構成内の各データ・チャンクは、表領域パーティションとして実装され、専用のオペレーティング・システム・データ・ファイル内に存在します。そのため、データベース初期化パラメータDB_FILESは、システム内のチャンク数(CREATE SHARDCATALOGまたはADD SHARDSPACEコマンドで指定)や表領域数の合計以上にする必要があります。

```
$ sqlplus / as sysdba
```

```
SQL> REM run the following command if using a CDB
SQL> alter session set container=catalog_pdb_name;
SQL> show parameter db_files
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
db_files                             integer     1024
```

- CREATE SHARDを使用してシャードをシャーディング構成に追加する場合は、SHARED_SERVERSおよびDISPATCHERSデータベース初期化パラメータを設定して、リモート・スケジューラ・エージェントがXDB接続を介してカタログに接続できるようにする必要があります。ADD SHARDを使用する場合、これは必要ありません。

  具体的には、シャード・ホストで実行されているリモート・スケジューラ・エージェント・プロセスからシャード・カタログへの共有サーバー接続を許可するには、SHARED_SERVERSを0 (ゼロ)より大きくする必要があります。また、DISPATCHERSの値には、Oracle SIDの値に基づいてXDBのサービスが含まれている必要があります。

```
$ sqlplus / as sysdba
SQL> show parameter shared_servers
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
shared_servers                       integer     5
SQL> show parameter dispatchers
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
Dispatchers                          string      (PROTOCOL=TCP), (PROTO
                                                 COL=TCP)(SERVICE=mysid
                                                 XDB)
```

  パラメータ値を適切に設定したら、ALTER SYSTEM REGISTERコマンドを実行して、XDBサービスが着信接続リクエストで使用できることを確認します。

- シャーディング・チャンク管理インフラストラクチャで使用されるOracle Managed Filesをサポートするには、データベース・パラメータDB_CREATE_FILE_DESTに有効な値が設定されている必要があります。

  この場所は、チャンクの移動操作(MOVE CHUNKや自動リバランスなど)の実行中に、チャンク・データを保持するトランスポータブル表領域を保存するために使用されます。さらに、『*Oracle Database*管理者ガイド』のOracle Managed Filesの使用に関する項で説明されているファイルも、Oracle Managed Filesを使用するOracleデータベースの慣例に従って、この場所に保存されます。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=catalog_pdb_name;
SQL> show parameter db_create_file_dest
NAME                 TYPE      VALUE
-------------------- --------- ------------------------------
db_create_file_dest  string    /u01/app/oracle/oradata
```

- スタンバイ・カタログ・データベースがシャーディング構成の一部である場合、スタンバイ・カタログ・データベースに新しいデータベース・ファイルを自動的に作成するために、STANDBY_FILE_MANAGEMENTデータベース・パラメータを設定する必要があります。

  このパラメータがMANUAL (デフォルト)に設定されている場合、たとえばCREATE TABLESPACEコマンドで作成される新しいデータベース・ファイルは、スタンバイには作成されません。これにより、スタンバイがプライマリ・データベースになると、データが使用できなくなり、アプリケーション・エラーが発生します。

```
$ sqlplus / as sysdba
```

```
SQL> alter session set container=catalog_pdb_name;
SQL> show parameter standby_file_management
NAME TYPE VALUE
------------------------------------ ----------- ------------
standby_file_management stirng AUTO
```

- Oracle提供のGSMCATUSERという名前のユーザー・アカウントは、シャード・カタログに指定したレガシー・データベースまたはPDB内でロック解除してパスワードを割り当てる必要があります。このアカウントは、シャード・ディレクタのプロセスがシャード・カタログ・データベースに接続して、シャーディング・コマンドに応じて管理タスクを実行するために使用されます。

  PDBをシャード・カタログとして使用している場合、GSMCATUSERはコンテナ・データベースの共通ユーザーであることに注意してください。そのため、そのパスワードはCDB$ROOTおよびCDB内のすべてのPDBで同じになります。単一のCDB内にある複数のPDBが別々のシャーディング構成のカタログ・データベースとして使用されていると、それらすべてが同じGSMCATUSERのパスワードを共有するようなりセキュリティ上の問題が発生することがあります。これを回避するために、CDBごとにシャード・カタログPDBを1つのみホストして、それ以外のPDBではGSMCATUSERアカウントのロックを解除しないようにします。

  指定したパスワードは、この後のシャーディング・トポロジの作成時に発行するADD GSMコマンドで使用します。これは、シャード・ディレクタによってOracleウォレットに安全に保管され、必要なときにのみ復号化されるため、再指定が必要になることはありません。

  MODIFY GSMコマンドは、その後にシャード・カタログ・データベースでパスワードが変更されたときに、保管したパスワードを更新するために使用できます。

```
$ sqlplus / as sysdba
SQL> alter user gsmcatuser account unlock;
User altered.
SQL> alter user gsmcatuser identified by gsmcatuser_password;
User altered.
```

  PDBをシャード・カタログとして使用している場合、次のコマンドも実行します。

```
SQL> alter session set container=catalog_pdb_name;
SQL> alter user gsmcatuser account unlock;
User altered.
```

- シャード・カタログの管理者アカウントは、シャード・カタログとして指定したレガシー・データベースまたはPDB内で作成し、パスワードを割り当てて、権限を付与する必要があります。

  このアカウントは、シャード・カタログ・データベース内のシャーディング・メタデータに対する管理者アカウントです。管理者がシャード・データベース・トポロジに変更を加えるなどの管理タスクを実行する必要があるときに、GDSCTLユーティリティを使用してシャード・カタログにアクセスするために使用します。

  GDSCTLは、GDSCTLコマンドの実行時に、このユーザーとしてシャード・カタログ・データベースに接続します。指定したユーザー名とパスワードは、この後のCREATE SHARDCATALOGコマンドで使用します。前述したGSMCATUSERと同様に、ユーザー名とパスワードは今後の使用に備えてOracleウォレットに安全に保存されます。保存された資格証明は、GDSCTLから明示的にCONNECTコマンドを発行してウォレット内の値をリセットすることで更新できます。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=catalog_pdb_name;
SQL> create user mysdbadmin identified by mysdbadmin_password;
User created.
SQL> grant gsmadmin_role to mysdbadmin;
```

```
Grant succeeded.
```

- Oracle Net TNSリスナーを設定して選択したポート(デフォルトは1521)で実行します。これにより、シャード・カタログのレガシー・データベースまたはPDBに対する着信接続リクエストを処理できます。

  TNSリスナーは、どのような方法で作成および構成してもかまいません。シャード・カタログがPDBの場合、データベースの作成方法によっては、ALTER SESSION SET CONTAINERを使用する必要のない、PDBへの直接接続リクエストを許可できるデータベース・サービスを明示的に作成することが必要になる場合もあります。

  シャード・カタログにPDBを使用している場合、リスナーが正しく構成されていることを確認するには、前の手順で新しく作成したmysdbadminアカウントと適切な接続文字列を使用して、次の操作を実行します。LSNRCTL SERVICESを実行すると、このリスナーを使用して現在利用可能なすべてのサービスが示されます。

```
$ sqlplus mysdbadmin/mysdbadmin_password@catalog_connect_string
SQL> show con_name
CON_NAME
------------------------
catalog_pdb_name
```

  接続を確認したら、前述のcatalog_connect_stringを記録しておきます。これは、この後の構成プロセスのGDSCTL CREATE SHARDCATALOGコマンドで使用します。一般に、これはhost:port/service_nameの形式になります(たとえば、cathost.example.com:1521/catalog_pdb.example.com)。

前述の要件がすべて満たされていると、新しく作成したデータベースはGDSCTL CREATE SHARDCATALOGコマンドの実行可能対象になります。

高可用性と障害回復のために、1つ以上のスタンバイ・シャード・カタログ・データベースも作成するようにしてください。シャーディングの観点からは、前述の要件がスタンバイ・データベースでも満たされていて、プライマリ・シャード・カタログ・データベースに対するすべての変更がスタンバイに確実に適用されていれば、その他に必要なシャーディング固有の構成ステップはありません。

シャード・データベースの作成

CREATE SHARDメソッドを使用してシャードを構成に追加する場合、このトピックはCREATE SHARDに適用されないためスキップします。それ以外の場合は、シャードとして使用するデータベースをそれぞれのホストで作成する必要があります。

シャード・カタログ・データベースと同様に、シャード・データベースの作成方法やプロビジョニング方法はシャーディングの観点からすると重要ではありません。このデータベースは、Database Configuration Assistant (DBCA)で作成することも、SQL*Plusを使用して手動で作成することも、クラウド・インフラストラクチャ・ツールからプロビジョニングすることもできます。

次の特性を備えた各シェード・ホストでOracle Database Enterprise Editionインスタンスを実行していれば、シャードとして使用できます。

- シャードがCDB内のPDBである場合、Oracle提供のGSMROOTUSERという名前のユーザー・アカウントは、シャードに指定したデータベースのCDB$ROOT内でロック解除してパスワードを割り当てる必要があります。さらに、このユーザーには、システム権限SYSDGおよびSYSBACKUPを付与する必要があります。

  GSMROOTUSERアカウントは、GDSCTLおよびシャード・ディレクタのプロセスがシャード・データベースに接続して、シャーディング・コマンドに応じて管理タスクを実行するするために使用されます。指定したパスワードは、シャーディング・トポロジの作成時にGDSCTLによって発行されるADD CDBコマンドで使用されます。また、シャード・ディレクタでシャード・データベースにOracle Data Guardを構成するためにDEPLOYコマンドを実行するときにも使用されます(必要な場合)。ユー

ザーによる再指定が必要なることはありません。GDSCTLとシャード・ディレクタによってOracleウォレット内に安全に保管され、必要なときにのみ復号化されます。MODIFY CDBコマンドは、その後にシャード・データベースでパスワードが変更されたときに、保管したパスワードを更新するために使用できます。

```
$ sqlplus / as sysdba
SQL> alter user gsmrootuser account unlock;
User altered.
SQL> alter user gsmrootuser identified by gsmrootuser_password;
User altered.
SQL> grant SYSDG, SYSBACKUP to gsmrootuser;
Grant succeeded.
```

- シャードがPDBである場合、シャード・データベースとして使用するPDBを作成します。CDBのルート・コンテナ(CDB$ROOT)をシャードとして使用することは、サポートされていません。
- シャード・データベースでは、サーバー・パラメータ・ファイル(SPFILE)を使用する必要があります。SPFILEが必要になる理由は、シャーディング・インフラストラクチャが内部データベース・パラメータを使用して構成メタデータを保存し、そのデータはデータベースの起動操作と停止操作の間で永続している必要があるためです。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> show parameter spfile
NAME     TYPE      VALUE
-------- --------- ------------------------------------
spfile   string    /u01/app/oracle/dbs/spfileshard.ora
```

- シャード・データベースのデータベース文字セットと各国語文字セットは、シャード・カタログ・データベースとその他のすべてのシャード・データベースで使用されているものと同じにする必要があります。つまり、シャード・カタログまたはシャードのいずれかに挿入される可能性のある文字がすべて含まれている文字セットを選択する必要があるということです。

  この要件は、MOVE CHUNKコマンドのシャーディング時に、トランスポータブル表領域をシャード間で移動するためにOracle Data Pumpが内部的に使用されることから発生します。このメカニズムの要件は、ソースと宛先で文字セットが一致していることです。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> select * from nls_database_parameters
  2  where parameter like '%CHARACTERSET';
PARAMETER                                VALUE
---------------------------------------- --------------------
NLS_NCHAR_CHARACTERSET                   AL16UTF16
NLS_CHARACTERSET                         WE8DEC
```

- COMPATIBLE初期化パラメータを少なくとも12.2.0に設定する必要があります。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> show parameter compatible
NAME                 TYPE        VALUE
-------------------- ----------- -----------------
compatible           string      19.0.0
```

- フラッシュ・データベースは、シャード・データベースがスタンバイ・シャード・データベースを使用する場合に有効にします。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> select flashback_on from v$database;
FLASHBACK_ON
------------------
YES
```

- FORCE LOGGINGモードは、シャード・データベースがスタンバイ・シャード・データベースを使用する場合に有効にする必要があります。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> select force_logging from v$database;
FORCE_LOGGING
---------------------------------------
YES
```

- データベース初期化パラメータDB_FILESは、システム内のチャンクや表領域の合計数以上に設定します。

  シャーディング構成内の各データ・チャンクは、表領域パーティションとして実装され、専用のオペレーティング・システム・データファイル内に存在します。そのため、データベース初期化パラメータDB_FILESは、システム内のチャンク数(CREATE SHARDCATALOGコマンドまたはADD SHARDSPACEコマンドで指定)や表領域数の合計以上にする必要があります。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> show parameter db_files
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
db_files                             integer     1024
```

- シャーディング・チャンク管理インフラストラクチャで使用されるOracle Managed Filesをサポートするには、データベース・パラメータのDB_CREATE_FILE_DESTに有効な値が設定されている必要があります。

  この場所は、チャンクの移動操作(MOVE CHUNKや自動リバランスなど)の実行中に、チャンク・データを保持するトランスポータブル表領域を保存するために使用されます。さらに、『*Oracle Database*管理者ガイド』のOracle Managed Filesの使用に関する項で説明されているファイルも、Oracle Managed Filesを使用するOracleデータベースの慣例に従って、この場所に保存されます。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> show parameter db_create_file_dest
NAME                 TYPE      VALUE
-------------------- --------- ------------------------------
db_create_file_dest  string    /u01/app/oracle/oradata
```

- DATA_PUMP_DIRというディレクトリ・オブジェクトをシャード・データベース内に作成して、GSMADMIN_INTERNALアカウントからアクセスできるようにする必要があります。

  GSMADMIN_INTERNALは、すべてのシャーディング・メタデータ表とPL/SQLパッケージを所有するOracle提供のアカウントです。ロックしたままにして、対話的なログインに使用されないようにしてください。シャーディング・メタデータとPL/SQLを所有することと、それに対するアクセスを制御することのみを目的としたものです。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> create or replace directory DATA_PUMP_DIR as  '/u01/app/oracle/oradata' ;
Directory created.
SQL> grant read, write on directory DATA_PUMP_DIR to gsmadmin_internal;
Grant succeeded.
```

- シャード間のファイル移動をサポートするには、データベース・パラメータのDB_FILE_NAME_CONVERTに有効な値が設定されている必要があります。この場所は、一般的な非シャーディング・データベースのように、スタンバイ・データベースが使用中のときに使用され、チャンク移動操作中にも使用できます。通常のファイル・システムの場所の場合は、このパラメータの末尾をスラッシュ(/)にすることをお薦めします。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> show parameter db_file_name_convert
NAME TYPE VALUE
-------------------- --------- ------------------------------
db_file_name_convert   string    /dbs/SHARD1/, /dbs/SHARD1S/
```

- スタンバイ・シャード・データベースがシャーディング構成の一部である場合、STANDBY_FILE_MANAGEMENTデータベース・パラメータをAUTOに設定して、スタンバイ・シャード・データベースに新しいデータベース・ファイルを自動的に作成する必要があります。

  このパラメータがMANUAL (デフォルト)に設定されている場合、たとえばCREATE TABLESPACEコマンドで作成される新しいデータベース・ファイルは、スタンバイには作成されません。これにより、スタンバイがプライマリ・データベースになると、データが使用できなくなり、アプリケーション・エラーが発生します。

```
$ sqlplus / as sysdba
SQL> alter session set container=shard_pdb_name;
SQL> show parameter standby_file_management
NAME TYPE VALUE
------------------------------------ ----------- ------------
standby_file_management string AUTO
```

- Oracle提供のGSMUSERという名前のユーザー・アカウントは、シャード・データベースとして指定したPDBまたはレガシー・データベース内でロック解除してパスワードを割り当てる必要があります。さらに、このユーザーには、システム権限SYSDGおよびSYSBACKUPを付与する必要があります。

  シャードがPDBの場合、GSMUSERはCDBの共通ユーザーであることに注意してください。そのため、そのパスワードはCDB$ROOTおよびCDB内のすべてのPDBで同じになります。これは、セキュリティ上の問題につながります。これを回避するために、CDBごとにシャードPDBを1つのみホストして、それ以外のPDBではGSMUSERアカウントのロックを解除しないようにします。

  このアカウントは、シャード・ディレクタのプロセスがシャード・データベースに接続して、シャーディング・コマンドに応じて管理タスクを実行するために使用されます。指定したパスワードは、この後のシャーディング・トポロジの作成時に発行するADD SHARDコマンドで使用します。このパスワードは、シャード・ディレクタによってOracleウォレットに安全に保管され、必要なときにのみ復号化されるため、再指定が必要になることはありません。その後、シャード・データベースでパスワードが変更された場合、保管したパスワードはMODIFY SHARDコマンドを使用して更新できます。

```
$ sqlplus / as sysdba
SQL> alter user gsmuser account unlock;
User altered.
```

```
SQL> alter user gsmuser identified by gsmuser_password;
User altered.
SQL> REM run the following commands if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> alter user gsmuser account unlock;
User altered.
SQL> REM all cases run the following command
SQL> grant SYSDG, SYSBACKUP to gsmuser;
Grant succeeded.
```

- Oracle Net TNSリスナーを設定して選択したポート(デフォルトは1521)で実行します。これにより、シャードPDBに対する着信接続リクエストを処理できます。

  TNSリスナーは、どのような方法で作成および構成してもかまいません。シャードがPDBの場合、データベースの作成方法によっては、ALTER SESSION SET CONTAINERを使用する必要のない、PDBへの直接接続リクエストを許可できるデータベース・サービスを明示的に作成することが必要になる場合もあります。

  シャードにPDBを使用している場合、リスナーが正しく構成されていることを確認するには、新しくロック解除したGSMUSERアカウントと適切な接続文字列を使用して、次の操作を実行します。LSNRCTL SERVICESを実行すると、このリスナーを使用して現在利用可能なすべてのサービスが示されます。

  ```
  $ sqlplus gsmuser/gsmuser_password@shard_connect_string
  SQL> show con_name
  CON_NAME
  ------------------------
  shard_pdb_name
  ```

  接続を確認したら、前述のshard_connect_stringを記録しておきます。これは、この後の構成プロセスのGDSCTL ADD SHARDコマンドで使用します。一般に、この接続文字列はhost:port/service_nameの形式になります(たとえば、shardhost.example.com:1521/shard_pdb.example.com)。

シャード・データベースの検証

前述の要件がすべて満たされていることを確認するには、Oracle提供のプロシージャ validateShardを実行します。これにより、シャード・データベースを検査して、発生した問題があるときに報告します。このプロシージャは、読取り専用であり、データベース構成に変更を加えることはありません。

validateShardプロシージャは、シャード・データベース構成に含まれるプライマリ、マウント済(未オープン)スタンバイ、およびActive Data Guardスタンバイのデータベースに対して実行する必要があります。validateShardは、シャード・データベースのライフサイクル期間中に、アップグレード後やパッチ適用後など、いつでも何度でも実行できます。

validateShardパッケージを実行するには、次のように操作します。

```
$ sqlplus / as sysdba
SQL> REM run the following command if using a CDB
SQL> alter session set container=shard_pdb_name;
SQL> set serveroutput on
SQL> execute dbms_gsm_fix.validateShard
```

このプロシージャでは、次のような出力が生成されます。

```
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is SHARD1.
```

```
INFO: Database unique name is shard1.
INFO: Database ID is 4183411430.
INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
INFO: Flashback is on.
INFO: Force logging is on.
INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is WE8DEC. This value must match the character set of the catalog
database.
INFO: 'compatible' initialization parameter validated successfully.
INFO: Database is a multitenant container database.
INFO: Current container is SHARD1_PDB1.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '/u01/app/oracle/dbs'
INFO: db_recovery_file_dest set to: '/u01/app/oracle/dbs'
INFO: db_files=1000. Must be greater than the number of chunks and/or
tablespaces to be created in the shard.
INFO: dg_broker_start set to TRUE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '/dbs/SHARD1/, /dbs/SHARD1S/'
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is '/u01/app/oracle/dbs/9830571348DFEBA8E0537517C40AF64B'.
```

INFOのマークが付いているすべての出力行は、情報の提示を目的としています。この行の情報が目的の構成に適っていることを確認する必要があります。

ERRORのマークが付いているすべての行は、デプロイメントの次のステップに進む前に修正する必要があります。こうした問題が解決されていないと、シャーディング作成操作のエラーの原因になります。

WARNINGのマークが付いているすべての出力行は、目的の構成に適していることも、適していないこともあります。たとえば、このデプロイメントにはスタンバイ・データベースを使用しないことにしていた場合、スタンバイ・データベースやリカバリに関連する警告は無視できます。特に、本番以外のデプロイ、概念実証のデプロイ、アプリケーション開発のデプロイなどの場合に当てはまります。すべての警告を確認して、必要に応じて解決してください。

ここまでのすべてのステップを完了すると、新しく作成したデータベースはGDSCTL ADD SHARDコマンドの実行可能対象になります。

高可用性と障害回復のために、1つ以上のスタンバイ・シャード・データベースも作成するようにしてください。シャーディングの観点からは、前述の要件がスタンバイ・データベースでも満たされていて、プライマリ・シャード・データベースに対するすべての変更がスタンバイに確実に適用されていれば、スタンバイ・データベースに必要な作業は、ADD SHARDコマンドでシャーディング構成を追加することのみです。

シャード・データベース・トポロジの構成

シャード・データベース・トポロジは、シャード・カタログ・データベースのシャーディング・メタデータによって記述されます。GDSCTLを使用して、シャード・データベース・トポロジを構成します。

シャード・データベース・トポロジは、シャーディング方法、レプリケーション(高可用性)テクノロジ、シャード・データベースに用意するチャンクのデフォルト数、シャード・ディレクタの場所と数、シャード・データベース内のシャードグループ、シャード領域、リージョンおよびシャードの数、およびシャード・データベースへの接続に使用するグローバル・サービスで構成されます。

『*Oracle Database Global Data Services*概要および管理ガイド』のGlobal Data Services Control Utility (GDSCTL)コマンド・リファレンスを手元に用意して、構成手順で使用するGDSCTLコマンドの使用方法とオプションの詳細を調べてください。

- 次に示す手順に従い、示された順序で、シャード・データベース・トポロジの構成を完了してください。

  GDSCTLコマンドライン・インタフェースは、シャード・ディレクタ(グローバル・サービス・マネージャ)インストールの一部としてインストールされるため、コマンドはシャード・ディレクタ・ホストから実行してください。

シャード・カタログの作成

GDSCTL CREATE SHARDCATALOGコマンドは、シャード・データベース・トポロジについての情報を示すメタデータをシャード・カタログ・データベースに作成するために使用します。

CREATE SHARDCATALOGを実行して、残りのシャーディング・メタデータが作成されると、いくつかのメタデータのプロパティはシャード・データベース全体を最初から再作成しないと変更できなくなります。こうしたものには、シャーディング方法(システム管理、コンポジット、ユーザー定義)、レプリケーション・テクノロジ(Oracle Data Guard、Oracle GoldenGate)、データベース内のチャンクのデフォルト数などがあります。コマンドに使用可能なオプションとそのデフォルト値の完全なリストは、GDSCTLのリファレンス・ドキュメントを参照してください。

コマンドの使用方法は、GDSCTLのドキュメントを参照するか、GDSCTL HELP CREATE SHARDCATALOGを実行してください。

シャード・カタログ接続文字列

CREATE SHARDCATALOGコマンドを実行すると、GDSCTLは指定されたユーザー名と接続文字列でシャード・カタログ・データベースに接続します。

高可用性または障害回復のために、シャード・カタログ・データベースにスタンバイ・データベースが関連付けられている場合は、接続文字列(次の例のcatalog_connect_string)で、すべてのプライマリ・データベースおよびスタンバイ・データベースを指定する必要があります。接続文字列にスタンバイ・データベースを含めていないと、シャード・ディレクタのプロセスはプライマリ・シャード・カタログが使用不可のときにスタンバイに接続できなくなります。

シャード・カタログ・データベースがPDBの場合、catalog_connect_stringでは、CDB$ROOTではなく、シャード・カタログ・データベースのPDBを指定する必要があります。

次に、簡潔なtnsnames.oraのエントリを示します。

```
CATALOG_CONNECT_STRING=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = tcp)(HOST = primary_catalog)(PORT = 1521))
      (ADDRESS = (PROTOCOL = tcp)(HOST = standby_catalog)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = catpdb.example.com)
    )
  )
```

ADD SHARDメソッドを使用してシャードを作成する場合は、最初のステップのみを実行します。CREATE SHARDメソッドを使用する

場合は、両方のステップを実行します。

1. 計画したシャーディング・トポロジに適した設定で、CREATE SHARDCATALOGを実行します。

   CREATE SHARDメソッドに必要な追加パラメータ

   CREATE SHARDメソッドを使用してシャードを構成に追加する場合は、CREATE SHARDCATALOGを実行するときに、次の追加パラメータを設定する必要があります。これは、次のステップでリモート・スケジューラ・エージェント登録に必要です。

   - -agent_passwordでは、リモート・スケジューラ・エージェントがシャード・カタログへの登録に使用するパスワードを指定します。
   - -agent_portでは、エージェントがシャード・カタログへのXDB接続の作成に使用するポート番号を指定します。このパラメータのデフォルト値は8080です。

   システム管理のシャーディング方法

   次の例では、システム管理シャーディング構成用のシャード・データベース・メタデータが作成されます。この構成には、region1およびregion2という2つのリージョンがあります。システム管理はデフォルトのシャーディング方法であるため、-shardingパラメータで指定する必要はありません。

   ```
   GDSCTL> create shardcatalog -database catalog_connect_string
    -user mysdbadmin/mysdbadmin_password -repl DG -region region1,region2
   ```

   -shardspaceの指定も省略すると、shardspaceoraというデフォルトのシャード領域が作成されます。-regionの指定を省略すると、regionoraというデフォルトのリージョンが作成されます。単一のデフォルト・リージョンがデフォルト・シャード領域とともに作成されると、そのシャード領域にshardspaceora_regionoraというデフォルトのシャードグループも作成されます。

   コンポジット・シャーディング方法

   次の例は、コンポジット・シャード・データベース用のシャード・カタログ・メタデータの作成方法を示しています。ここでは、MaxAvailability保護モードのData Guardレプリケーション、シャード領域ごとに60チャンク、および2つのシャード領域を設定します。

   ```
   GDSCTL> create shardcatalog -database catalog_connect_string
    -user mysdbadmin/mysdbadmin_password -sharding composite -chunks 60
    -protectmode maxavailability -shardspace shardspace1,shardspace2
   ```

   ユーザー定義のシャーディング方法

   次の例は、ユーザー定義のシャード・データベース用のシャード・カタログ・メタデータの作成方法を示しています。ここでは、Data Guardレプリケーションを設定しています。

   ```
   GDSCTL> create shardcatalog -database catalog_connect_string
    -user mysdbadmin/mysdbadmin_password -sharding user
    -protectmode maxperformance
   ```

2. CREATE SHARDメソッドのみの場合: リモート・スケジューラ・エージェントをシャード・カタログに登録し、各シャード・ホストでエージェントを起動します。

   各シャード・ホストに移動して、Oracleソフトウェア・インストールの所有者としてログインし、シャード・データベースの実行元となるOracleホームで次のschagentコマンドを実行します。

   ```
   schagent -registerdatabase catalog_hostname agent_port
   schagent -start
   ```

前述のschagentコマンドで、catalog_hostnameをシャード・カタログ・ホストの名前に置き換え、agent_portを前述のCREATE SHARDCATALOGで構成したポート番号に置き換えます。

たとえば:

```
$ $ORACLE_HOME/bin/schagent -registerdatabase cathost.example.com 8080
$ $ORACLE_HOME/bin/schagent -start
```

正常にエージェントを登録すると、シャード・ホストはGDSCTL DEPLOY中にシャード・カタログからリモート・ジョブ・リクエストを受信できます。特定のホストに正常にデプロイされると、リモート・スケジューラ・エージェントは、シャード・データベースのライフ・サイクルでは使用されなくなり、次のコマンドを使用して安全に停止できます。

```
$ $ORACLE_HOME/bin/schagent -stop
```

シャード・カタログへの今後の接続

GDSCTLは、シャード・カタログ管理者の資格証明をローカル・ホストのウォレットに保管します。ただし、次回以降の別のホストでのGDSCTLセッションでは、次に示すようにGDSCTL CONNECTコマンドを使用して、管理タスクを実行するために明示的にシャード・カタログに接続することが必要になる場合があります。

```
GDSCTL> connect mysdbadmin/mysdbadmin_password@catalog_connect_string
```

シャード・ディレクタの追加と起動

構成にシャード・ディレクタを追加して起動します。シャード・ディレクタでは、GDSCTLコマンドなどのイベントに応じてシャーディング・システムの監視や、バックグラウンド・タスクを実行します。

次のコマンドは、シャード・ディレクタのプロセスを実行するホストで実行する必要があります。これは、シャード・カタログ・ホストまたはシャード・ディレクタ・プロセスの専用ホストのどちらかになります。

1. 次の例に示すように、シャード・ディレクタ(GSM)を追加して起動します。

   ```
   GDSCTL> connect mysdbadmin/mysdbadmin_password@catalog_connect_string
   GDSCTL> add gsm -gsm sharddirector1 -catalog catalog_connect_string -pwd gsmcatuser_password
   GDSCTL> start gsm -gsm sharddirector1
   ```

   -gsmパラメータの値は、今後のGDSCTLコマンドで、このシャード・ディレクタを参照するために使用する名前です。-catalogパラメータと-pwdパラメータの値は、シャード・カタログ・データベースの作成時に使用したものと同じにする必要があります。

   パラメータの-listener、-localons、および-remoteonsは、GDSCTLのリファレンスで説明されているように、それぞれのポート番号1522、6123、および6234をオーバーライドするために使用します。使用するポート番号は、デフォルトかユーザー定義化にかかわらず、ホストで使用可能なことと、実行中の別のソフトウェアやOracleリスナーと競合していないことを必ず確認してください。

2. 追加のシャード・ディレクタがある場合は、それぞれのシャード・ディレクタ・ホストでADD GSMコマンドとSTART GSMコマンドを繰り返します。

   シャード・ディレクタの名前(この例では、sharddirector1)は、シャード・ディレクタごとに適切な名前に置き換えます。

   複数のシャード・ディレクタを使用する場合は、CREATE SHARDCATALOGコマンドで、それらのための複数のリージョンを作成しておく必要があります。また、ADD REGIONを実行することで後からリージョンを追加することもできます。

シャード・ディレクタごとのリージョンは、次に示すように、ADD GSMごとの-regionパラメータで指定します。

```
GDSCTL> add gsm -gsm sharddirector2 -catalog catalog_connect_string -pwd gsmcatuser_password -
region dc2
```

今後のGDSCTLセッションでは、管理するシャード・ディレクタの明示的な指定が必要になることがあります。デフォルトのGSMORA シャード・ディレクタを示すエラー・メッセージが表示され場合は、次に示すように、GDSCTL SET GSMを実行してから作業を進めてください。

```
GDSCTL> set gsm -gsm sharddirector1
```

シャード領域の追加(必要な場合)

コンポジット・シャーディングまたはユーザー定義シャーディングを使用するときに、目的のシャーディング・トポロジの達成にシャード領域の追加が必要な場合は、ADD SHARDSPACEコマンドを使用してシャード領域を追加します

- 次に示すように、ADD SHARDSPACEを実行します。

  ```
  GDSCTL> add shardspace -shardspace shardspace2
  ```

  デフォルトでは、ADD SHARDSPACEコマンドは、CREATE SHARDCATALOGコマンドで使用した-chunksと-protectmodeの値を継承します。チャンクの数とData Guardの保護モードは、ADD SHARDSPACEに-chunksパラメータと-protectmodeパラメータを使用することでシャード領域ごとに指定できます。

シャードグループの追加(必要な場合)

シャード・データベース・トポロジにシステム管理またはコンポジットのシャーディング方法を使用する場合は、アプリケーション用に必要な追加のシャードグループを追加することもできます。

それぞれのシャード領域には、少なくとも1つのプライマリ・シャードグループを含める必要があり、任意の数またはタイプのスタンバイ・シャードグループを含めることができます。シャードグループは、ユーザー定義のシャーディング方法では使用しません。

- ADD SHARDGROUPを実行して、構成にシャードグループを追加します。

  ```
  GDSCTL> add shardgroup -shardgroup shardgroup_primary -shardspace shardspace1
   -deploy_as primary -region region1
  GDSCTL> add shardgroup -shardgroup shardgroup_standby -shardspace shardspace1
   -deploy_as active_standby -region region2
  ```

  ADD SHARDGROUPを実行するときに-deploy_asパラメータを使用すると、シャードグループの3つのタイプprimary、standby (マウント済、未オープン)、およびactive_standby (オープン、問合せに使用可能)を指定できます(デフォルトは、standbyです)。

  この後でシャードグループに追加したシャードは、そのシャードグループの-deploy_as設定に対応するモードでオープンする必要があります。たとえば、プライマリ・シャードグループの場合は読取り/書込み、スタンバイ・シャードグループの場合はマウント済、またはアクティブ・スタンバイ・シャードグループの場合は読取り専用を適用します。

  シャードのデプロイ後、シャードの現在のモードはシャード・ディレクタによって監視され、その後のスイッチオーバー操作やフェイルオーバー操作によっては同じシャードグループ内に異なるオープン・モードのシャードが存在する可能性や予測などがシャード・カタログに通知されます。

シャーディング・トポロジの検証

シャード・データベースに関する情報をカタログに追加する前に、シャーディング・トポロジが適切なことを確認します。その後で、各種のGDSCTL CONFIGコマンドを使用して作業を進めてください。

シャードを追加してデプロイした後では、シャード・カタログ・メタデータの大部分が変更できなくなります。そのため、この時点で構成を検証することが重要なタスクになります。

- GDSCTL CONFIGを実行して、全体的な構成情報を表示します。

```
GDSCTL> config
Regions
------------------------
region1
region2
GSMs
------------------------
sharddirector1
sharddirector2
Sharded Database
------------------------
orasdb
Databases
------------------------
Shard Groups
------------------------
shardgroup_primary
shardgroup_standby
Shard spaces
------------------------
shardspaceora
Services
------------------------
GDSCTL pending requests
------------------------
Command                   Object                  Status
-------                   ------                  ------
Global properties
------------------------
Name: oradbcloud
Master GSM: sharddirector1
DDL sequence #: 0
```

シャード領域やシャードグループなどのシャード・カタログ・オブジェクトに関する詳細な情報は、各種のGDSCTL CONFIGコマンドを使用して表示できます。様々なGDSCTL CONFIGコマンドの完全なリストについては、GDSCTLのリファレンス・ドキュメントを参照するか、GDSCTL HELPを実行してください。

シャードCDBの追加

シャードがCDB内のPDBである場合、ADD CDBコマンドを使用して、シャードPDBを格納するCDBをシャーディング構成に追加します。シャードとして非CDBを使用するか、またはCREATE SHARDを使用してシャードを追加する場合は、次の項へスキップします。

1. 次に示すように、ADD CDBコマンドを実行します。

```
GDSCTL> add cdb -connect cdb_connect_string -pwd gsmrootuser_password
```

このコマンドにより、GDSCTLはSYSDGとしてGSMROOTUSER/gsmrootuser_password@cdb_connect_stringに接続し、設定を検証して、CDBのDB_UNIQUE_NAMEを取得します。これが、シャード・カタログでのCDB名になります。

2. 構成内のシャードPDBを格納するすべてのCDBに対して、ADD CDBコマンドを繰り返します。
3. すべてのCDBを追加したら、GDSCTL CONFIG CDBを実行してカタログ内のCDBのリストを表示します。

```
GDSCTL> config cdb
```

シャードの追加

シャードを構成に追加するのにADD SHARDとCREATE SHARDのどちらを使用するかに応じて、次の適切な手順に従います。

GDSCTL ADD SHARDを使用したシャードの追加

GDSCTL ADD SHARDコマンドを使用して、シャード情報をシャード・カタログに追加します。

1. 次の例で示すように、対象のシャーディング方法に適した使用方法でADD SHARDを実行します。

   システム管理またはコンポジットのシャーディングの場合は、次に示すパラメータでADD SHARDを実行します。

   ```
   GDSCTL> add shard -connect shard_connect_string -pwd gsmuser_password
   -shardgroup shardgroup_name -cdb cdb_name
   ```

   ユーザー定義のシャーディングの場合は、わずかにコマンドの使用方法が異なります。

   ```
   GDSCTL> add shard -connect shard_connect_string -pwd gsmuser_password
   -shardspace shardspace_name -deploy_as db_mode -cdb cdb_name
   ```

   PDBをシャードとして使用しない場合は、-cdbパラメータを指定しないでください。

   前述の例で、-cdbパラメータでは、シャードPDBが存在しているCDBの名前を指定します。-shardgroupまたは-shardspaceでは、シャーディング・トポロジでのシャードの場所を指定します。また、-deploy_asでは、シャードのオープン・モード(primary、standby、active_standby)を指定します。

   ノート:

   接続文字列に server=dedicated を設定することをお薦めします。

   ADD SHARDを実行すると、GDSCTLはSYSDGとしてGSMUSER/gsmuser_password@shard_connect_stringに接続してシャードの設定を検証し、dbms_gsm_fix.validateShardを再実行してエラーがないか確認します。その後、GDSCTLは、次の規則を使用してシャード名を構成します。

   - PDBシャードの場合: db_unique_name_of_CDB_PDB_name (たとえばcdb1_pdb1)
   - レガシー・データベース・シャードの場合: db_unique_name_of_DB (単純にdb_unique_nameです)

   最後に、シャードについての情報を示すメタデータがシャード・カタログに追加されます。

2. GDSCTL CONFIG SHARDを実行して、シャード・カタログにあるシャード・メタデータを表示します。

```
GDSCTL> config shard
Name      Shard Group          Status    State    Region    Availability
--------- -------------------  ------    -----    ------    ------------
```

```
cdb1_pdb1 shardgroup_primary   U         none     region1   -
cdb2_pdb1 shardgroup_standby   U         none     region2   -
cdb3_pdb2 shardgroup_primary   U         none     region1   -
cdb4_pdb2 shardgroup_standby   U         none     region2   -
```

「Status」の値は、"アンデプロイ"のUになります。「State」と「Availability」は、DEPLOYコマンドの実行が正常に完了するまではnoneと-になります。

GDSCTL CREATE SHARDを使用したシャードの追加

GDSCTL CREATE SHARDコマンドを使用して、シャード・データベースを作成し、シャード情報をシャード・カタログに追加します。

次の例で示すように、対象のシャーディング方法に適したパラメータを使用してCREATE SHARDを実行します。

システム管理またはコンポジットのシャーディングの場合は、次に示すパラメータでCREATE SHARDを実行します。

```
GDSCTL> create shard -shardgroup shardgroup_name -destination shard_hostname
 -osaccount account_name -ospassword account_password
```

ユーザー定義のシャーディングの場合は、わずかにコマンドの使用方法が異なります。

```
GDSCTL> create shard -shardspace shardspace_name -deploy_as db_mode
 -destination shard_hostname -osaccount account_name -ospassword account_password
```

-shardgroupまたは-shardspaceパラメータではシャーディング・トポロジ内のシャードの場所を指定し、-deploy_asではシャードの目的のオープン・モード(primary、standby、active_standby)を指定します。

-destinationパラメータでは、NETCAおよびDBCAを生成してシャード・データベースを作成するために、シャード・カタログがやり取りするリモート・スケジューラ・エージェントを指定します。この値は通常、シャード・ホストのホスト名です。使用可能な宛先のリストを表示するには、シャード・カタログ・データベースのALL_SCHEDULER_EXTERNAL_DESTSビューから選択します。

-osaccountおよび-ospasswordパラメータでは、シャード・ホストでNETCAおよびDBCAプロセスを生成するときに使用されるオペレーティング・システムのユーザー名およびパスワードを指定します。通常、ユーザー名はOracle Databaseソフトウェアの所有者です。

パスワードの暗号化

各CREATE SHARDコマンドでアカウントのクリアテキスト・パスワードを指定しないようにするには、後で使用するために、GDSCTL ADD CREDENTIALコマンドを使用して、暗号化されたパスワードをシャード・カタログに格納できます。CREATE SHARDコマンドで、次に示すように-osaccountおよび-ospasswordではなくコマンド・パラメータに資格証明名を指定します。

```
GDSCTL> add credential -credential credential_name
 -osaccount account_name -ospassword account_password
GDSCTL> create shard -shardgroup shardgroup_name -destination shard_hostname
 -credential credential_name
```

CREATE SHARDの実行時の処理

CREATE SHARDを実行すると、GDSCTLによって入力パラメータおよびシャード・ホスト設定が検証され、シャード・メタデータがシャード・カタログに追加され、その結果、GDSCTL DEPLOY中に次の操作が実行されます。

- ポート1521で、リスナー名"LISTENER" (デフォルト)を使用して、TNSリスナー・プロセスをシャード・ホストに作成して起動します
- プライマリ・シャードの場合、$ORACLE_HOME/assistants/dbca/templates/General_Purpose.dbc (デフォルト)のシャード・ホストにあるデフォルトのDBCAテンプレートを使用して、シャード・データベースを作成します。

  プライマリ・シャードには、デフォルトで次の特性があります。

  - SYS、SYSTEMおよびGSMUSERに対してランダム生成されるパスワード
  - db_unique_name、db_nameおよび'shNN'形式のSID (NNは、追加されたシャードを一意に識別するための順序ベースの番号)
  - db_domain値: 指定した宛先に対応するALL_SCHEDULER_EXTERNAL_DESTS.HOSTNAME列にあるドメインと同じ。ドメインが見つからない場合、db_domainはシャード・カタログ・データベースのdb_domainに設定されます。
  - NLS_CHARACTERSET値およびNLS_NCHAR_CHARACTERSET値: シャード・カタログ・データベースの値と同じ
  - db_file_name_convertパラメータ: '*','$ORACLE_BASE/oradata/'に設定
  - db_create_file_destパラメータ: $ORACLE_BASE/oradataに設定
  - remote_login_passwordfileパラメータ: EXCLUSIVEに設定
  - データベース: アーカイブ・ログ・モード
  - 強制ロギング: 有効
  - データベース・フラッシュバック: オン
  - Oracle Data GuardレプリケーションがCREATE SHARDCATALOGに指定されている場合は、次のパラメータが設定されます。
    - dg_broker_start: TRUEに設定
    - db_recovery_file_dest: $ORACLE_BASE/fast_recovery_areaに設定
    - db_recovery_file_dest_size: 51200 MBに設定
    - standby_file_management: AUTOに設定
    - db_flashback_retention_target: 60に設定
- スタンバイ・シャードの場合、DBCAおよびRMANを使用して、既存のプライマリに基づいてスタンバイ・データベースを作成します。一般に、すべてのプライマリ・データベース・パラメータはスタンバイによって継承されます。

データベースのカスタマイズに関するCREATE SHARDの使用上のヒント

GDSCTL CREATE SHARDコマンドには、シャード・データベースをカスタマイズできるパラメータがいくつかあります。

- -sys_passwordおよび-system_passwordパラメータを使用すると、SYSおよびSYSTEMアカウントのパスワードを新しいシャードに指定できます。

  このアカウントは対話型ログインで使用されるものではないため、GSMUSERパスワードは常にランダムに作成されます。デプロイメント後にGSMUSERパスワードを変更するには、ALTER USERを使用してデータベース上のパスワードを変更し、GDSCTL MODIFY SHARDコマンドを使用して新しいパスワードでシャーディング・メタデータを更新します。
- -netparamおよび-netparamfileパラメータを使用すると、TNSリスナーの名前およびポート番号をシャード・ホストで作成する際にカスタマイズできます。

  これらのパラメータに指定する値は、NETCAレスポンス・ファイルのファイル名です。レスポンス・ファイルの例は、シャード・ホストの$ORACLE_HOME/assistants/netcaにあります。
- 同様に、-dbtemplateおよび-dbtemplatefileパラメータを使用して、シャード・データベースの作成時に使用されるDBCAテンプレート・ファイルを指定できます。

デフォルト・テンプレートは、シャード・ホスト上の
$ORACLE_HOME/assistants/dbca/templates/General_Purpose.dbcです。

- コマンドラインからDBCAを実行するときに入力するのと同様に、-dbparamおよび-dbparamfileパラメータを使用して、DBCAコマンドライン・パラメータをシャード・ホストのDBCAプロセスに直接渡すことができます。

  プライマリ・データベースの作成に使用可能なパラメータをすべて表示するには、dbca -help -createDatabaseを実行します。スタンバイ・データベースを作成するためのパラメータを表示するには、dbca -help -createDuplicateDBを実行します。たとえば、プライマリ・シャードのグローバル・データベース名およびSIDを変更するには、次に示すように1行のファイルを作成し、そのファイル名を-dbparamまたは-dbparamfileに指定します。

  ```
  -gdbName mydb.example.com -sid mysid
  ```

- これらのパラメータのいずれかを指定する場合は、GDSCTLのように、-netparamfile、-dbtemplatefileまたは-dbparamfileをオペレーティング・システム・ファイル名とともに使用できます。

  あるいは、ADD FILEコマンドを使用して、ファイルの内容をシャード・カタログ・データベースに保存してから、CREATE SHARDコマンドで-netparam、-dbtemplateまたは-dbparamを使用することもできます。

  ```
  GDSCTL> create shard -dbtemplatefile /home/user/mytemplate.dbc -netparamfile
  /home/user/mynetca.rsp ...
  ```

  または

  ```
  GDSCTL> add file -file mytemplate -source /home/user/mytemplate.dbc
  GDSCTL> add file -file mynetca -source /home/user/mynetca.rsp
  GDSCTL> create shard -dbtemplate mytemplate -netparam mynetca ...
  ```

特定のホストのスタンバイ・シャードが特定のプライマリ・シャードと同じData Guard構成にあることを保証する場合は、まずプライマリ・シャードを作成し、次にCREATE SHARDで目的の-destination値を使用してスタンバイ・シャードを作成することをお薦めします。複数のプライマリ・シャードを順次作成してから、複数のスタンバイ・シャードを作成する場合、Data Guard構成ではプライマリとスタンバイは非決定的な方法で一致します。

シャード構成の確認

GDSCTL CONFIG SHARDを実行して、シャード・カタログのシャード・メタデータが想定どおりであることを確認します。

```
GDSCTL> config shard
Name      Shard Group          Status    State    Region    Availability
--------- -------------------  ------    -----    ------    ------------
sh1       shardgroup_primary   U         none     region1   -
sh2       shardgroup_primary   U         none     region1   -
sh3       shardgroup_standby   U         none     region2   -
sh4       shardgroup_standby   U         none     region2   -
```

「Status」の値は、"アンデプロイ"のUになります。「State」と「Availability」は、GDSCTL DEPLOYコマンドの実行が正常に完了するまではnoneと-になります。

ホスト・メタデータの追加

すべてのシャード・ホストのホスト名とIPアドレスをシャード・カタログに追加します。

デプロイメント・プロセスの一環として、シャード・ディレクタはシャードと通信して、シャード・ディレクタのTNSリスナー・プロセスに登

録するように指示します。このリスナー・プロセスは、信頼できるソースからの着信登録リクエストのみを受け入れ、不明なホストからの登録リクエストを拒否します。

シャード・ホストに複数のホスト名またはネットワーク・インタフェースが割り当てられている場合、シャード・ディレクタへの着信登録リクエストは、ADD SHARDまたはCREATE SHARDの実行時に自動的に追加されていなかったホストから送信される可能性があります。この場合、その登録リクエストは拒否され、シャードは正常にデプロイされなくなります。この問題について目視できる現象は、DEPLOYの完了後に、CONFIG SHARDがシャードの「Availability」にPENDINGを示すことです。

この問題を回避するために、GDSCTL ADD INVITEDNODEコマンドを使用して、シャード・ホストのすべてのホスト名とIPアドレスをシャード・カタログ・メタデータに手動で追加します。

1. 信頼できるホストのリストを表示します。

   デフォルトでは、ADD SHARDおよびCREATE SHARDコマンドは、シャード・カタログ・メタデータにシャード・ホストのデフォルトのホスト名を追加します。そのため、そのホストからシャード・ディレクタへの登録リクエストがすべて受け入れられるようになります。信頼できるホストのリストは、GDSCTL CONFIG VNCRコマンドを実行することで表示できます。

   ```
   GDSCTL> config vncr
   ```

2. 構成内のすべてのホストからPingを実行して、ホスト名の解決が成功することを確認します。

   CONFIG VNCRの出力にリストされたすべてのホストは、その他のトポロジ内のすべてのホストから名前でアクセスできる必要があります。シャード、シャード・カタログ、およびシャード・ディレクタのホストからpingコマンドを使用して、リストされているすべてのホスト名のホスト名解決が成功することを確認します。

   問題を解決するには、オペレーティング・システムのコマンドや設定を使用して、すべてのホスト名を解決できることを確認します。

3. REMOVE INVITEDNODEコマンドを実行して、どのホストからも必要とされていないホスト名と解決できないホスト名を手動で削除します。
4. ADD INVITEDNODEコマンドを実行して、シャード・ホストのすべてのホスト名とIPアドレスをシャード・カタログ・メタデータに手動で追加します。

   ```
   GDSCTL> add invitednode 127.0.0.1
   ```

シャーディング構成のデプロイ

GDSCTLコマンドでシャード・データベース・トポロジの構成を完了したら、GDSCTL DEPLOYコマンドを実行してシャード・データベース構成をデプロイします。

GDSCTL DEPLOYコマンドを実行すると、ADD SHARDコマンドを使用してシャードを構成した場合、出力は次のようになります。

```
GDSCTL> deploy
deploy: examining configuration...
deploy: requesting Data Guard configuration on shards via GSM
deploy: shards configured successfully
The operation completed successfully
```

CREATE SHARDを使用してシャードを構成した場合、GDSCTL DEPLOYコマンドの出力は次のようになります。

```
GDSCTL> deploy
```

```
deploy: examining configuration...
deploy: deploying primary shard 'sh1' ...
deploy: network listener configuration successful at destination 'shard1'
deploy: starting DBCA at destination 'shard1' to create primary shard 'sh1' ...
deploy: deploying primary shard 'sh2' ...
deploy: network listener configuration successful at destination 'shard2'
deploy: starting DBCA at destination 'shard2' to create primary shard 'sh2' ...
deploy: waiting for 2 DBCA primary creation job(s) to complete...
deploy: waiting for 2 DBCA primary creation job(s) to complete...
deploy: DBCA primary creation job succeeded at destination 'shard1' for shard 'sh1'
deploy: DBCA primary creation job succeeded at destination 'shard2' for shard 'sh2'
deploy: deploying standby shard 'sh3' ...
deploy: network listener configuration successful at destination 'shard3'
deploy: starting DBCA at destination 'shard3' to create standby shard 'sh3' ...
deploy: deploying standby shard 'sh4' ...
deploy: network listener configuration successful at destination 'shard4'
deploy: starting DBCA at destination 'shard4' to create primary shard 'sh4' ...
deploy: waiting for 2 DBCA standby creation job(s) to complete...
deploy: waiting for 2 DBCA standby creation job(s) to complete...
deploy: DBCA standby creation job succeeded at destination 'shard3' for shard 'sh3'
deploy: DBCA standby creation job succeeded at destination 'shard4' for shard 'sh4'
deploy: requesting Data Guard configuration on shards via GSM
deploy: shards configured successfully
The operation completed successfully
```

デプロイ時の処理

DEPLOYを実行すると、いくつかの処理が発生します。

- GDSCTLは、シャード・カタログでシャード・データベース・トポロジ構成を調べるPL/SQLプロシージャをコールして、デプロイ可能なアンデプロイ状態のシャードが存在するかどうかを確認します。
- CREATE SHARDメソッドを使用してシャードを作成する場合、シャード・カタログのPL/SQLコードによって、NETCAを実行する各シャード・ホストでリモート・スケジューラ・エージェント・ジョブがスケジュールされ、TNSリスナーが作成および起動されます。その後、2つ目のジョブがシャード・ホストでDBCAを実行するようにスケジュールされ、シャード・データベースが作成されます。スタンバイをデプロイする場合は、別の一連のNETCAジョブとDBCAジョブが実行され、プライマリ・データベースが正常に作成された後、それぞれのホストでスタンバイ・データベースが作成されます。
- デプロイされるシャードについては、シャードのデータベース・パラメータを更新してシャードのトポロジ・メタデータを移入し、シャード・ディレクタに登録するためにシャードを送信するよう、シャード・カタログがシャード・ディレクタに向けてリクエストを送信します。
- Oracle Data Guardレプリケーションを使用しているときに、デプロイにスタンバイ・データベースが存在している場合、シャード・ディレクタは、プライマリ・シャードでPL/SQL APIをコールしてData Guard構成を作成するか、プライマリとスタンバイのセットに既存の構成を検証します。ファスト・スタート・フェイルオーバー機能が、すべてのシャードで有効化されます。さらに、シャード・ディレクタは、そのホストでData Guardオブザーバ・プロセスを起動して、Data Guard構成を監視します。
- すでにデプロイされたシャードが含まれている既存のシャード・データベースに新しいシャードを追加する場合(増分デプロイメント)は、以前に実行されたDDL文が新しいシャードで実行され、すべてのシャード間でアプリケーション・スキーマが同じになるようにします。
- 最後に、システム管理またはコンポジットのシャーディング方法を使用しているシャード・データベースに増分デプロイメントを実施する場合は、バックグラウンドでの自動チャンク移動がスケジュールされます。これは、現在の構成でチャンクの数がシャード間に均等に分散されるようにするためです。このプロセスは、DEPLOYコマンドがGDSCTLに制御を戻した後で、

GDSCTL CONFIG CHUNKSコマンドを使用することで監視できます。

デプロイメント成功時の表示

PDBおよびADD SHARDメソッドを使用したデプロイメントが正常に完了すると、Data Guardアクティブ・スタンバイ・シャードが使用されているときのCONFIG SHARDからの出力は、次のようになります。

```
GDSCTL> config shard
Name      Shard Group          Status   State     Region   Availability
--------- -------------------  -------  --------  -------  ------------
cdb1_pdb1 shardgroup_primary   Ok       Deployed  region1  ONLINE
cdb2_pdb1 shardgroup_standby   Ok       Deployed  region2  READ ONLY
cdb3_pdb2 shardgroup_primary   Ok       Deployed  region1  ONLINE
cdb4_pdb2 shardgroup_standby   Ok       Deployed  region2  READ ONLY
```

CREATE SHARDメソッドを使用した場合、またはADD SHARDを非CDBとともに使用した場合、シャード名はシャード・データベースのdb_unique_name値になります。

マウント済で未オープンのスタンバイが使用されていると、シャード・ディレクタはマウント済データベースのステータスをチェックするためにログインできないため、出力は次のようになります。

```
GDSCTL> config shard
Name      Shard Group          Status        State     Region   Availability
--------- -------------------  ------------- --------  -------  ------------
cdb1_pdb1 shardgroup_primary   Ok            Deployed  region1  ONLINE
cdb2_pdb1 shardgroup_standby   Uninitialized Deployed  region2  -
cdb3_pdb2 shardgroup_primary   Ok            Deployed  region1  ONLINE
cdb4_pdb2 shardgroup_standby   Uninitialized Deployed  region2  -
```

問題の修正方法

シャードの可用性にPENDINGが示されている場合は、トポロジ構成のADD INVITEDNODEおよびCONFIG VNCRに関連するすべてのステップが完了していることを確認します。完了していない場合は、そのステップを今すぐ完了してから、GDSCTL SYNC DATABASE -database shard_nameを実行することでシャードのデプロイメントを完了します。

CREATE SHARDメソッドを使用してシャードを構成に追加し、リモート・スケジューラ・エージェント・ジョブからGDSCTL DEPLOY中にNETCAまたはDBCAからのエラーが戻された場合は、次のステップを実行してエラーを解決し、デプロイメントを再試行します。

1. 問題を解決します。

   GDSCTL DEPLOYによって戻されたエラー・メッセージには、シャード・ホストで失敗したジョブの出力を表示するための十分な情報が含まれています。

   通常、シャード・ホストの$ORACLE_BASE/cfgtoollogsに、NETCAまたはDBCA実行からのトレース・ファイルおよびログ・ファイルがあります。失敗の原因となった根本的な問題(不正なパラメータ、ホスト上のリソースの問題など)を解決します。

2. シャード・ホストをリセットします。

   a. デプロイの試行中に作成された実行中のTNSリスナーを停止します。
   b. デプロイの試行中に起動された実行中のシャード・データベースを停止します。
   c. $ORACLE_HOME/network/admin/listener.oraを削除します
   d. 失敗したシャード作成に関連付けられているすべてのファイルを$ORACLE_BASE/oradataおよび

$ORACLE_BASE/fast_recovery_areaから削除します

3. GDSCTL DEPLOYを再度実行します。

グローバル・データベース・サービスの作成と開始

シャードのデプロイが正常に完了して、適切なステータスであることを確認したら、アプリケーションからの着信接続リクエストを処理するためにシャードにグローバル・データベース・サービスを作成して、そのサービスを開始します。

たとえば、次の例のコマンドでは、構成内のプライマリ・シャードに読取り/書込みサービスが作成され、スタンバイ・シャードに読取り専用サービスが作成されます。これらのサービス名は、接続文字列で使用することで、アプリケーションから正しいシャードに適切にリクエストをルーティングできるようになります。

サービスが開始されたら、シャード・データベースはアプリケーション・スキーマの作成および着信クライアント接続リクエストの準備ができました。

例3-1 すべてのプライマリ・シャードで実行されるグローバル・サービスの追加と開始

次のコマンドでは、oltp_rw_srvcというグローバル・サービスを作成して開始します。このサービスは、クライアントがシャード・データベースに接続するために使用できます。oltp_rw_srvcサービスはプライマリ・シャードで読取り/書込みトランザクションを実行します。

```
GDSCTL> add service -service oltp_rw_srvc -role primary
GDSCTL> start service -service oltp_rw_srvc
```

例3-2 スタンバイ・シャードで実行する読取り専用のワークロードのためのグローバル・サービスの追加と開始

スタンバイ・シャードで読取り専用のワークロードを実行するために、oltp_ro_srvcグローバル・サービスが作成および開始されます。これは、スタンバイ・シャードが、読取り専用アクセスでオープンされるOracle Active Data Guardスタンバイ・シャードであることを前提としています。マウント済で未オープンのスタンバイは読取り専用接続に対応できません。そのようなスタンバイは、障害回復と高可用性のためにのみ存在します。

```
GDSCTL> add service -service oltp_ro_srvc -role physical_standby
GDSCTL> start service -service oltp_ro_srvc
```

例3-3 グローバル・サービスのステータスの確認

```
GDSCTL> config service
Name         Network name                    Pool    Started Preferred all
----         ------------                    ----    ------- -------------
oltp_rw_srvc oltp_rw_srvc.orasdb.oracdbcloud orasdb  Yes     Yes
oltp_ro_srvc oltp_ro_srvc.orasdb.oracdbcloud orasdb  Yes     Yes
GDSCTL> status service
Service "oltp_rw_srvc.orasdb.oradbcloud" has 2 instance(s). Affinity: ANYWHERE
   Instance "orasdb%1", name: "cdb1_pdb1", db: "cdb1_pdb1", region: "region1", status: ready.
   Instance "orasdb%21", name: "cdb3_pdb2", db: "cdb3_pdb2", region: "region1", status: ready.
Service "oltp_ro_srvc.orasdb.oradbcloud" has 2 instance(s). Affinity: ANYWHERE
   Instance "orasdb%11", name: "cdb2_pdb1", db: "cdb2_pdb1", region: "region2", status: ready.
   Instance "orasdb%31", name: "cdb4_pdb2", db: "cdb4_pdb2", region: "region2", status: ready.
```

シャード・ステータスの確認

シャーディング構成のデプロイでDEPLOYステップを完了したら、各シャードの詳細なステータスを確認します。

GDSCTL CONFIG SHARDを実行して、各シャードの詳細なステータスを表示します。

```
GDSCTL> config shard -shard cdb1_pdb1
Name: cdb1_pdb1
Shard Group: shardgroup_primary
Status: Ok
State: Deployed
Region: region1
Connection string:shard_connect_string
SCAN address:
ONS remote port: 0
Disk Threshold, ms: 20
CPU Threshold, %: 75
Version: 19.0.0.0
Failed DDL:
DDL Error: ---
Management error:
Failed DDL id:
Availability: ONLINE
Rack:
Supported services
------------------------
Name Preferred Status
---- --------- ------
oltp_ro_srvc Yes Enabled
oltp_rw_srvc Yes Enabled
```

シャード・データベースのデプロイの例

この例では、複数のレプリカを備えた一般的なシステム管理シャード・データベースのデプロイ方法を示します。このデプロイでは、高可用性のためにOracle Data Guardを使用します。この例のシャード・カタログおよびシャードはPDBであり、シャードはADD SHARDコマンドで構成に追加されます。

システム管理のシャード・データベースをデプロイするには、シャードグループおよびシャードを作成し、シャードとして使用するデータベースを作成および構成し、DEPLOYコマンドを実行してロールベースのグローバル・サービスを作成します。

システム管理のシャーディングでは、シャードにデータをマップする必要はありません。これは、コンシステント・ハッシュによるパーティション化を使用して、データがシャード間に自動的に分散されるためです。パーティション化アルゴリズムにより、データがシャード間に均一およびランダムに分散されます。システム管理のシャード・データベースの概念に関する詳細は、システム管理のシャーディングを参照してください。

シャード・データベース・トポロジの例

次に示すシステム管理シャード・データベース構成について検討します。この構成では、シャードグループ1にプライマリ・シャードが格納され、シャードグループ2と3にスタンバイ・レプリカが格納されます。

さらに、シャードグループ2のレプリカはOracle Active Data Guardのスタンバイ(読取り専用アクセスでオープンされたデータ

ベース)であり、シャードグループ3のレプリカは未オープンのマウント済データベースだと仮定します。

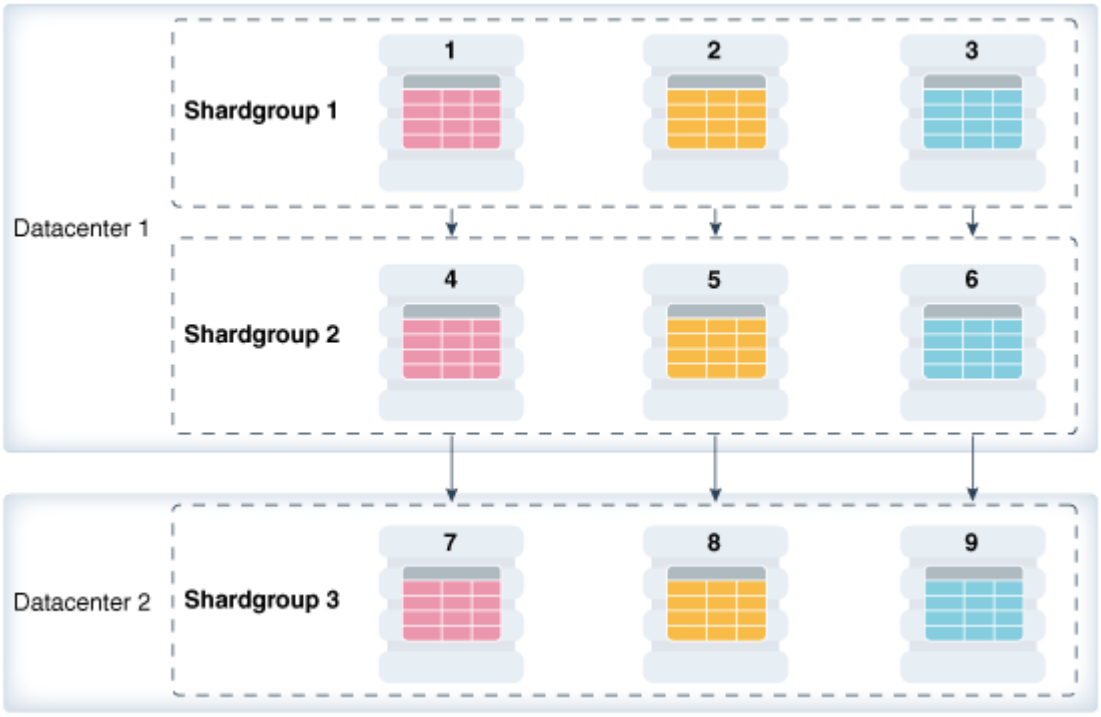


表3-1 サンプルのシステム管理トポロジのホスト名

トポロジ・オブジェクト	説明
シャード・カタログ・データベース	すべてのシャード・データベース・トポロジに、シャード・カタログが必要です。この例では、シャード・カタログ・データベースに 2 つのスタンバイがあります(データ・センターごとに 1 つ)。 プライマリ ● データ・センター= 1 ● ホスト名= cathost ● DB_UNIQUE_NAME = catcdb ● PDB 名= catpdb ● 接続サービス名= catpdb アクティブ・スタンバイ ● データ・センター= 1 ● ホスト名= cathost1 スタンバイ ● データ・センター= 2

トポロジ・オブジェクト	説明
	● ホスト名= cathost2
リージョン	この構成には 2 つのデータ・センターが関与しているため、それに対応する 2 つのリージョンがシャード・カタログ・データベースに作成されています。 データ・センター1 ● リージョン名= dc1 データ・センター2 ● リージョン名= dc2
シャード・ディレクタ(グローバル・サービス・マネージャ)	それぞれのリージョンには、そのデータ・センター内のホストで実行するシャード・ディレクタが必要です。この例は、リージョンごとに 2 つのシャード・ディレクタを使用する方法を示しており、これがベスト・プラクティスの推奨です。 データ・センター1 ● シャード・ディレクタ・ホスト名= gsmhost1 および gsmhost1b ● シャード・ディレクタ名= gsm1 および gsm1b データ・センター2 ● シャード・ディレクタ・ホスト名= gsmhost2 および gsmhost2b ● シャード・ディレクタ名= gsm2 および gsm2b
シャードグループ	データ・センター1 ● sg1 ● sg2 データ・センター2 ● sg3
シャード	● ホスト名= shardhost1, …, shardhost9 ● DB_UNIQUE_NAME = cdb1、…、cdb9 ● PDB 名= pdb1, pdb2, pdb3 スタンバイ・レプリカの PDB 名は、それらに対応するプライマリの PDB 名と同じになります

サンプル・シャード・データベースのデプロイ

次のステップを実行して、サンプルのシステム管理シャード・データベースをデプロイします。このデータベースは、高可用性のためにOracle Data Guardを使用して、複数のレプリカを備えています。

1. ホストcathost、cathost1、cathost2、gsmhost1、gsmhost1b、gsmhost2、gsmhost2bおよびshardhost1からshardhost9までをプロビジョニングして構成します。

   詳細は、「ホストおよびオペレーティング・システムのプロビジョニングと構成」を参照してください。

2. ホストcathost、cathost1、cathost2およびshardhost1からshardhost9にOracle Databaseソフトウェアをインストールします。

   詳細は、「Oracle Databaseソフトウェアのインストール」を参照してください。

3. ホストgsmhost1、gsmhost1b、gsmhost2およびgsmhost2bにシャード・ディレクタ・ソフトウェアをインストールします。

   詳細は、「シャード・ディレクタ・ソフトウェアのインストール」を参照してください。

4. シャード・カタログ・データベースを作成し、cathostでOracle TNSリスナーを起動します。

   さらに、カタログのスタンバイ・レプリカをcathost1およびcathost2に作成して、それらのスタンバイにプライマリ・カタログへの変更が適用されていることを確認します。

   詳細は、「シャード・カタログ・データベースの作成」を参照してください。

5. ホストshardhost1、shardhost2およびshardhost3に、シャード・データを格納する3つのプライマリ・データベースを作成します。

   それに対応するレプリカを作成します。その場所と名前は、次のとおりです。

   - shardhost4 (cdb4)およびshardhost7 (cdb7)に、shardhost1 (cdb1/pdb1)のレプリカ
   - shardhost5 (cdb5)およびshardhost8 (cdb8)に、shardhost2 (cdb2/pdb2)のレプリカ
   - shardhost6 (cdb6)およびshardhost9 (cdb9)に、shardhost3 (cdb3/pdb3)のレプリカ

   9つのコンテナ・データベース(CDB)のdb_unique_nameは、cdb1からcdb9にする必要があります。そこにあるPDBの名前は、3つのプライマリおよびレプリカでpdb1、pdb2およびpdb3にする必要があります。

   CDBのサービス名はcdb1からcdb9にする必要があり、そのPDBシャードのサービス名はpdb1、pdb2、およびpdb3です。

   詳細は、「シャード・データベースの作成」を参照してください。

6. すべてのポート番号がデフォルトであるとすると、シャード・データベース・トポロジを構成するには、次のGDSCTLコマンドを発行します。このとき、ドメインとパスワードは適切な値に置き換えます。

   - ホストgsmhost1で、GDSCTLから次のコマンドを実行します。

     ```
     create shardcatalog -database cathost.example.com:1521/catpdb.example.com -user
     mydbsadmin/mydbsadmin_password -region dc1,dc2
     add gsm -gsm gsm1 -region dc1 -catalog cathost.example.com:1521/catpdb.example.com -pwd
     gsmcatuser_password
     start gsm -gsm gsm1
     ```

     詳細は、「シャード・カタログの作成」および「シャード・ディレクタの追加と起動」を参照してください。

- ホストgsmhost1bで、GDSCTLから次のコマンドを実行します。

```
connect mydbsadmin/mydbsadmin_password@cathost.example.com:1521/catpdb.example.com
add gsm -gsm gsm1b -region dc1 -catalog cathost.example.com:1521/catpdb.example.com -
pwd gsmcatuser_password
start gsm -gsm gsm1b
```

詳細は、「シャード・ディレクタの追加と起動」を参照してください。

- ホストgsmhost2で、GDSCTLから次のコマンドを実行します。

```
connect mydbsadmin/mydbsadmin_password@cathost.example.com:1521/catpdb.example.com
add gsm -gsm gsm2 -region dc2 -catalog cathost.example.com:1521/catpdb.example.com -pwd
gsmcatuser_password
start gsm -gsm gsm2
```

詳細は、「シャード・ディレクタの追加と起動」を参照してください。

- ホストgsmhost2bで、GDSCTLから次のコマンドを実行します。

```
connect mydbsadmin/mydbsadmin_password@cathost.example.com:1521/catpdb.example.com
add gsm -gsm gsm2b -region dc2 -catalog cathost.example.com:1521/catpdb.example.com -
pwd gsmcatuser_password
start gsm -gsm gsm2b
```

詳細は、「シャード・ディレクタの追加と起動」を参照してください。

- ホストgsmhost1に戻って、GDSCTLから次のコマンドを実行して、シャード・データベースの設定を完了します。

```
add shardgroup -shardgroup sg1 -deploy_as primary -region dc1
add shardgroup -shardgroup sg2 -deploy_as active_standby -region dc1
add shardgroup -shardgroup sg3 -deploy_as standby -region dc2
add cdb -connect shardhost1.example.com:1521/cdb1.example.com -pwd gsmrootuser_password
add cdb -connect shardhost2.example.com:1521/cdb2.example.com -pwd gsmrootuser_password
```

shardhost3からshardhost9およびcdb3からcdb9でADD CDBコマンドを繰り返してから、次のコマンドを実行します。

```
add shard -connect shardhost1.example.com:1521/pdb1.example.com -pwd gsmuser_password -
shardgroup sg1 -cdb cdb1
add shard -connect shardhost2.example.com:1521/pdb2.example.com -pwd gsmuser_password -
shardgroup sg1 -cdb cdb2
add shard -connect shardhost3.example.com:1521/pdb3.example.com -pwd gsmuser_password -
shardgroup sg1 -cdb cdb3
add shard -connect shardhost4.example.com:1521/pdb1.example.com -pwd gsmuser_password -
shardgroup sg2 -cdb cdb4
add shard -connect shardhost5.example.com:1521/pdb2.example.com -pwd gsmuser_password -
shardgroup sg2 -cdb cdb5
add shard -connect shardhost6.example.com:1521/pdb3.example.com -pwd gsmuser_password -
shardgroup sg2 -cdb cdb6
add shard -connect shardhost7.example.com:1521/pdb1.example.com -pwd gsmuser_password -
shardgroup sg3 -cdb cdb7
add shard -connect shardhost8.example.com:1521/pdb2.example.com -pwd gsmuser_password -
shardgroup sg3 -cdb cdb8
add shard -connect shardhost9.example.com:1521/pdb3.example.com -pwd gsmuser_password -
shardgroup sg3 -cdb cdb9
```

詳細は、「シャードグループの追加(必要な場合)」、「シャードCDBの追加」および「GDSCTL ADD

SHARDを使用したシャードの追加」を参照してください。

- コマンドCONFIG VNCRとADD INVITEDNODEを使用して、VNCRエントリのすべてが有効でデプロイメントが成功するために不足がないことを確認します。

  詳細は、「ホスト・メタデータの追加」を参照してください。

- GDSCTLからDEPLOYを実行して、シャード・データベースの構成を完了します。

  詳細は、「シャーディング構成のデプロイ」を参照してください。

- 読取り/書込みアクセスと読取り専用アクセスのサービスをシャード/データベースに追加して開始します。

```
add service -service oltp_rw_srvc -role primary
start service -service oltp_rw_srvc
add service -service oltp_ro_srvc -role physical_standby
start service -service oltp_ro_srvc
```

  詳細は、「グローバル・データベース・サービスの作成と開始」を参照してください。

7. コマンドGDSCL CONFIG、CONFIG SHARD、およびCONFIG SERVICEを使用すると、シャードとサービスのすべてがオンラインになっていて実行されていることを確認できます。

   詳細は、「シャード・ステータスの確認」を参照してください。

Oracle Shardingでの透過的データ暗号化の使用

Oracle Shardingでは透過的データ暗号化(TDE)がサポートされますが、TDEを有効にした状態でシャード・データベース内のチャンクを正常に移行できるように、すべてのシャードが暗号化された表領域に対する同じ暗号化キーを共有して使用する必要があります。

シャード・データベースは、複数の独立したデータベースと1つのカタログ・データベースで構成されます。特にシャード間でデータを移動するときにTDEが正しく機能するように、一定の制限が適用されます。データが暗号化されているときにシャード間のチャンク移動が正常に機能するためには、すべてのシャードで同じ暗号化キーを使用する必要があります。

これを実現するには、次の2つの方法があります。

- シャード・カタログから暗号化キーを作成してエクスポートし、すべてのシャードに個々にキーをインポートしてアクティブ化します。
- ウォレットを共有の場所に格納し、シャード・カタログおよびすべてのシャードで同じウォレットを使用します。

シャードDDLを有効にしたシャード・カタログで次のTDE文を実行すると、その操作がシャードに自動的に伝播されます。

- alter system set encryption wallet open/close identified by password
- alter system set encryption key
- administer key management set keystore [open|close] identified by password
- administer key management set key identified by password
- administer key management use key identified by password
- administer key management create key store identified by password

制限事項

Oracle ShardingでのTDEの使用には、次の制限事項が適用されます。

- MOVE CHUNKが正常に機能するには、すべてのシャード・データベース・ホストが同じプラットフォームに存在する必要があります。
- MOVE CHUNKでは、データ転送中にパフォーマンスに影響を及ぼす可能性がある圧縮を使用できません。
- 表領域レベルでの暗号化のみがサポートされます。特定の列に対する暗号化はサポートされません。

関連項目:

TDEの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

すべてのシャードに対する単一の暗号化キーの作成

シャード・データベース構成内のすべてのデータベースに単一の暗号化キーを伝播するには、シャード・カタログでマスター暗号化キーを作成し、ウォレットをエクスポートしてシャードにインポートし、キーをアクティブ化する必要があります。

> ノート:
>
> この手順は、キーストア・パスワードとウォレット・ディレクトリ・パスがシャード・カタログおよびすべてのシャードで同じであることを前提としています。異なるパスワードとディレクトリ・パスが必要な場合は、各シャードとシャード・カタログで、シャード DDL を無効化し、シャードの独自のパスワードとパスを使用して、すべてのコマンドを個別に発行する必要があります。

次のステップは、データの暗号化を実行する前に行う必要があります。

1. シャード・カタログで暗号化キーを作成します。

   シャードDDLを有効にして、次の文を発行します。

   ```
   ADMINISTER KEY MANAGEMENT CREATE KEYSTORE wallet_directory_path IDENTIFIED BY
    keystore_password;
   ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY keystore_password;
   ```

   ウォレットのオープンおよびクローズ・コマンドをカタログから一元的に発行する場合は、keystore_passwordが同じである必要があります。

   > ノート:
   >
   > ウォレット・ディレクトリ・パスは、対応する sqlnet.ora の ENCRYPTION_WALLET_LOCATION と一致する必要があります。
   >
   > ENCRYPTION_WALLET_LOCATION パラメータは非推奨です。かわりに、WALLET_ROOT 静的初期化パラメータおよび TDE_CONFIGURATION 動的初期化パラメータを使用することをお薦めします。

   シャードDDLを無効にして、次の文を発行します。

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY keystore_password WITH BACKUP;
```

シャード・カタログ・データベースのウォレットに暗号化キーが作成され、アクティブ化されます。

DDLを有効にしてこの文を発行すると、各シャードのウォレットにもカタログのキーとは異なる暗号化キーが作成されます。データ移動を正常に実行するには、各シャード上の異なる暗号化キーを使用しないでください。

2. シャード・カタログのキーストアからマスター・キーのIDを取得します。

```
SELECT KEY_ID  FROM V$ENCRYPTION_KEYS
WHERE ACTIVATION_TIME =
 (SELECT MAX(ACTIVATION_TIME) FROM V$ENCRYPTION_KEYS
  WHERE ACTIVATING_DBID = (SELECT DBID FROM V$DATABASE));
```

3. シャードDDLを無効にして、カタログの暗号化キーを含むウォレットをエクスポートします。

```
ADMINISTER KEY MANAGEMENT EXPORT ENCRYPTION KEYS WITH SECRET secret_phrase TO
 wallet_export_file IDENTIFIED BY keystore_password;
```

(オプション)ここでステップの結果を入力します。

4. ウォレット・ファイルを各シャード・ホストの対応するウォレット・エクスポート・ファイルの場所に物理的にコピーするか、すべてのシャードがアクセスできる共有ディスクに格納します。
5. シャードDDLを無効にして、各シャードにログオンし、キーを含むウォレットをインポートします。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY keystore_password;
ADMINISTER KEY MANAGEMENT IMPORT ENCRYPTION KEYS WITH SECRET secret_phrase FROM
 wallet_export_file IDENTIFIED BY keystore_password WITH BACKUP;
```

6. シャード・データベースを再起動します。
7. すべてのシャード上のキーをアクティブ化します。

カタログでシャードDDLを有効にして、次のコマンドを実行します。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY keystore_password;
ADMINISTER KEY MANAGEMENT USE KEY master_key_id IDENTIFIED BY keystore_password
 WITH BACKUP;
```

これで、すべてのシャードとシャード・カタログ・データベースで同じ暗号化キーがアクティブ化され、データの暗号化に使用する準備が完了しました。シャード・カタログで、(シャードDDLを有効にして)次のようなTDE DDLを発行できます。

- 暗号化された表領域および表領域セットを作成します。
- 暗号化された表領域を使用してシャード表を作成します。
- 暗号化された列を含むシャード表を作成します(制限があります)。

すべてのシャード上のキーIDがシャード・カタログ上のIDと一致していることを検証します。

```
SELECT KEY_ID  FROM V$ENCRYPTION_KEYS
WHERE ACTIVATION_TIME =
 (SELECT MAX(ACTIVATION_TIME) FROM V$ENCRYPTION_KEYS
  WHERE ACTIVATING_DBID = (SELECT DBID FROM V$DATABASE));
```

4 Oracle Cloud InfrastructureでのOracle Database Shardingの使用

Oracle Shardingのツールには、シャード・データベースのデプロイ操作を自動化し、さらに簡略化するためのTerraform、KubernetesおよびAnsibleのスクリプトが含まれています。

Kubernetesでのシャード・データベースのデプロイ

Oracle Cloud Infrastructure AnsibleモジュールおよびHelm/チャートを使用して、Oracle Kubernetes Engine (OKE)でのシャード・データベースのプロビジョニングを自動化します。

OKEにOracle Shardingをデプロイするには、Oracle Cloud Infrastructure Ansibleモジュールでコンピュート・リソースを作成し、ネットワークを構成し、Ansibleプレイブックに渡されるyamlファイルを使用してブロック・ストレージ・ボリュームを作成します。

Kubernetesでのシャード・データベース・デプロイメントの手順およびダウンロードは、https://github.com/oracle/db-sharding/tree/master/oke-based-sharding-deploymentを参照してください。

Terraformによるシャード・データベースのデプロイ

Oracle Shardingのツールには、Oracle Cloud Infrastructureとオンプレミスの両システムでシャード・データベースのデプロイを自動化するためのTerraformモジュールおよびスクリプトが含まれています。

Terraformモジュールおよびスクリプトでは、シャード・ディレクタ、シャード・カタログおよびシャードを含む完全なシャード・データベース・インフラストラクチャを作成して構成します。このスクリプトには、シャード・データの高可用性および障害時リカバリを実現するためにレプリケーションにOracle Data Guardを使用し、スタンバイ・シャードおよびシャード・カタログをデプロイするためのオプションも用意されています。

設定プロセスの一環として、Terraformバイナリをインストールし、Oracle Shardingシャード・ディレクタ・インストール・パッケージをダウンロードし、またオンプレミス・デプロイメントの場合には、Oracle Databaseインストール・ファイルをダウンロードします。

次の場所で、ターゲット・システムのTerraformベースのシャード・データベースのデプロイに関する手順およびダウンロードを参照してください。

- Oracle Cloud Infrastructure: https://github.com/oracle/db-sharding/tree/master/deployment-with-terraform/sdb-terraform-oci。
- オンプレミス: https://github.com/oracle/db-sharding/tree/master/deployment-with-terraform/sdb-terraform-onprem

Dockerによるシャード・データベースのデプロイ

Oracle Shardingには、DevOpsユーザーのシャード・データベースのインストール、構成および環境設定を容易にするサンプ

ルのDockerビルド・ファイルが用意されています。

このプロセスでは、Dockerエンジンのインストールおよび構成、グローバル・サービス・マネージャ(シャード・ディレクタ)およびOracle Databaseイメージの作成、ネットワーク・ブリッジの作成、Oracle Shardingオブジェクトおよびシャード・ディレクタのコンテナの作成、およびコンテナのデプロイを行います。

Dockerによるシャード・データベース・デプロイメントの手順およびダウンロードは、https://github.com/oracle/db-sharding/tree/master/docker-based-sharding-deploymentを参照してください。

5 シャード・データベース・スキーマの設計

シャーディングの利点を活用するには、1つのシャードで実行されるデータベース・リクエストの数が最大になるように、シャード・データベースのスキーマを設計する必要があります。

シャード・データベース・スキーマの設計に関する考慮事項

データベース・スキーマの設計はシャード・データベースのパフォーマンスとスケーラビリティに大きく影響します。スキーマの設計が不適切な場合、データおよびワークロードがシャード間にバランスよく分散されず、マルチシャード操作の割合が大きくなります。

データ・モデルは、単一のルート表を持つ階層ツリー構造である必要があります。Oracle Shardingは任意のレベル数の階層をサポートします。

シャーディングの利点を活用するには、1つのシャードで実行されるデータベース・リクエストの数が最大になるように、シャード・データベースのスキーマを設計する必要があります。

シャード・データベース・スキーマは、次の特性を持つシャード表ファミリおよび重複表で構成されます。

シャード表ファミリ

- シャーディング・キーによって等価パーティション化された一連の表。
    - 関連するデータは、常にまとめて格納および移動されます。
    - 結合および整合性制約チェックはシャード内で実行されます。
- シャーディング方法およびキーは、アプリケーションの要件に基づいています。
- シャーディング・キーは、主キーに含まれている必要があります。

重複表

- すべてのシャードにレプリケートされる非シャード表。
- 通常、共通参照データが含まれます。
- 各シャードで読取りおよび更新できます。

シャード・データベース・スキーマの設計の計画

シャード・データーベースにデータを移入した後では、表をシャーディングするのか複製するのか、シャーディング・キーなど、スキーマの属性の多くは変更できません。したがって、シャード・データベースをデプロイする前に次の点を慎重に考慮してください。

- どの表をシャーディングするか?
- どの表を複製するか?
- どのシャード表をルート表にするか?
- 他の表をルート表にリンクするためにどの方法を使用するか?
- どのシャーディング方法を使用するか?
- どのシャーディング・キーを使用するか?
- 使用するスーパー・シャーディング・キー(シャーディング方法がコンポジットである場合)

シャーディング・キーの選択

シャード表パーティションはシャーディング・キーに基づいて表領域レベルでシャード間に分散されます。キーの例として顧客ID、アカウント番号、国IDなどがあります。

シャーディング・キーは、次の特性に従う必要があります。

- シャーディング・キーが非常に安定していて、その値がほとんど変更されることがない必要があります。
- シャーディング・キーはすべてのシャード表に存在する必要があります。これにより、シャーディング・キーに基づく等価パーティション表のファミリを作成できます。
- 表ファミリ内の表の結合は、シャーディング・キーを使用して実行する必要があります。

システム管理シャード・データベースのシャーディング・キー

システム管理のシャーディング方法の場合、シャーディング・キーはカーディナリティが高い列に基づいている必要があります。この列の一意の値の数は、シャードの数より大幅に多い必要があります。たとえば、顧客IDはシャーディング・キーのよい候補ですが、米国の州名は適していません。

シャーディング・キーには、単一列または複数列を指定できます。複数の列が存在する場合、列のハッシュが連結されてシャーディング・キーが形成されます。

次の例では、Customersというシャード表を作成し、列cust_idおよびnameが表のシャーディング・キーを構成するように指定します。

```
CREATE SHARDED TABLE customers
(cust_id     NUMBER NOT NULL
, name         VARCHAR2(50)
, address      VARCHAR2(250)
, region       VARCHAR2(20)
, class        VARCHAR2(3)
, signup       DATE,
CONSTRAINT cust_pk PRIMARY KEY(cust_id, name))
PARTITION BY CONSISTENT HASH (cust_id,name)
PARTITIONS AUTO
TABLESPACE SET ts1;

CREATE SHARDED TABLE Orders
( OrderNo   NUMBER NOT NULL
, CustNo    NUMBER NOT NULL
, Name      VARCHAR2(50) NOT NULL
, OrderDate DATE
, CONSTRAINT OrderPK PRIMARY KEY (CustNo, Name, OrderNo)
, CONSTRAINT CustFK  FOREIGN KEY (CustNo, Name) REFERENCES Customers(Cust_ID, Name)
)
PARTITION BY REFERENCE (CustFK);
```

コンポジット・シャード・データベースのシャーディング・キー

コンポジット・シャーディングを使用すると、2つのレベルのシャーディング(リストまたは範囲によるシャーディングとコンシステント・ハッシュによるシャーディング)を行うことができます。これは、2つのキー(スーパー・シャーディング・キーおよびシャーディング・キー)を提供するアプリケーションによって実現されます。

コンポジット・シャーディングは、次に示すように、複数列のLISTパーティション・セットをサポートしていません。

```
CREATE SHARDED TABLE customers (
cust_id     NUMBER NOT NULL,
Name    VARCHAR2(50) NOT NULL,
class VARCHAR2(3) NOT NULL ,
class2 number not null,
CONSTRAINT cust_pk PRIMARY KEY(cust_id,name,class))
PARTITIONSET BY LIST (class, class2)
PARTITION BY CONSISTENT HASH (cust_id,name)
PARTITIONS AUTO (
PARTITIONSET silver VALUES (('SLV',1),('BRZ',2)) TABLESPACE SET ts1
PARTITIONSET gold   VALUES (('GLD',3),('OTH',4)) TABLESPACE SET ts2);
PARTITION BY CONSISTENT HASH (cust_id,name)
*
ERROR at line 8:
ORA-02514: list PARTITIONSET method expects a single partitioning column
```

次に示すように、複数列のRANGEパーティション・セットがサポートされています。

```

CREATE SHARDED TABLE customers (
cust_id     NUMBER NOT NULL,
Name    VARCHAR2(50) NOT NULL,
class number NOT NULL ,
class2 number not null,
CONSTRAINT cust_pk PRIMARY KEY(cust_id,name,class))
PARTITIONSET BY RANGE (class, class2)
PARTITION BY CONSISTENT HASH (cust_id,name)
PARTITIONS AUTO (
PARTITIONSET silver VALUES LESS THAN (10,100) TABLESPACE SET ts1,
PARTITIONSET gold   VALUES LESS THAN (20,200) TABLESPACE SET ts2);
Table created.
```

前述のいずれの場合も、シャーディング・キー(パーティション・セット・キーではありません)は複数列にできます。

ユーザー定義シャード・データベースのシャーディング・キー

ユーザー定義シャーディングのリストによるパーティション化の場合、Oracle Shardingは単一のシャーディング・キー列を想定しています。リスト・パーティション化されたシャード表に複数の列が指定されている場合、エラーがスローされます。

```
CREATE SHARDED TABLE accounts
( id             NUMBER
, account_number NUMBER
, customer_id    NUMBER
, branch_id      NUMBER
, state          VARCHAR(2) NOT NULL
, state2         VARCHAR(2) NOT NULL
, status         VARCHAR2(1)
)
PARTITION BY LIST (state,state2)
( PARTITION p_northwest VALUES ('OR', 'WA') TABLESPACE ts1
, PARTITION p_southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE ts2
, PARTITION p_northcentral VALUES ('SD', 'WI') TABLESPACE ts3
, PARTITION p_southcentral VALUES ('OK', 'TX') TABLESPACE ts4
, PARTITION p_northeast VALUES ('NY', 'VM', 'NJ') TABLESPACE ts5
, PARTITION p_southeast VALUES ('FL', 'GA') TABLESPACE ts6
```

```
);
ERROR at line 1:
ORA-03813: list partition method expects a single partitioning column in
user-defined sharding
```

レンジ・パーティション化されたシャード表の場合、シャーディング・キー列として複数の列を指定できます。

```
CREATE SHARDED TABLE accounts
( id             NUMBER
, account_number NUMBER
, customer_id    NUMBER
, branch_id      NUMBER
, state          NUMBER NOT NULL
, state2         NUMBER NOT NULL
, status         VARCHAR2(1)
)
PARTITION BY RANGE (state, state2)
( PARTITION p_northwest VALUES LESS THAN(10, 100) TABLESPACE ts1
, PARTITION p_southwest VALUES LESS THAN(20,200) TABLESPACE ts2);
Table created.
```

ただし、どちらの場合も、シャーディング・キー(パーティションセット・キーではありません)は複数列にできます。

シャーディング・キー・タイプのサポート

シャーディング・キーでは次のデータ型がサポートされています。

- NUMBER
- INTEGER
- SMALLINT
- RAW
- (N)VARCHAR
- (N)VARCHAR2
- (N)CHAR
- DATE
- TIMESTAMP

主キーおよび外部キーの制限事項

シャーディング環境では、主キーの制約および外部キーの制約は次のルールによって制御されます。

- 主キーの場合、シャード表に一意制約および一意索引があり、列リストにシャーディング・キー列が含まれている必要があります。以前のOracleリリースでは、シャーディング・キーはこのような列の接頭辞である必要がありましたが、このルールはより緩和されました。
- あるシャード表から別のシャード表への外部キーにもシャーディング・キーが含まれている必要があります。外部キーは参照表の主キーまたは一意の列を参照するため、これは自動的に強制されます。
- シャード表の外部キーは、同じ表ファミリ内にある必要があります。異なる表ファミリには異なるシャーディング・キー列があるため、これは必須です。

- ローカル表を参照するシャード表の外部キーは許可されていません。
- 重複表を参照するシャード表の外部キーは許可されていません。
- シャード表を参照する重複表の外部キーは許可されていません。

シャード表の索引

シャード表に作成できるのはローカル索引のみです。シャード表の一意のローカル索引にはシャーディング・キーを含める必要があります。

シャード表のグローバル索引は、オンライン・チャンク移動のパフォーマンスを損なう可能性があるため、許可されません。

次の例では、account表のid列にid1というローカル索引を作成します。

```
CREATE INDEX id1 ON account (id) LOCAL;
```

次の例では、account表のid列およびstate列に対して、id2というローカル一意索引を作成します。

```
CREATE UNIQUE INDEX id2 ON account (id, state) LOCAL;
```

シャード・データベースでのDDLの実行

シャード・データベースにスキーマを作成するには、シャード・カタログ・データベースでDDLコマンドを発行する必要があります。シャード・カタログ・データベースは、DDLを検証し、シャードで実行する前にそれらをローカルで実行します。

シャード・カタログ・データベースには、シャード・データベースに存在するすべてのオブジェクトのローカル・コピーが含まれており、シャード・データベースのスキーマのマスター・コピーとして機能します。シャード・カタログの検証およびDDLの実行が成功した場合、そのDDLはすべてのシャードに自動的に伝播され、シャード・カタログで発行された順序で適用されます。

DDLの伝播中にシャードが停止しているか、アクセスできない場合、シャード・カタログはそのシャードに適用できなかったDDLを追跡し、そのシャードがアクセス可能になったときにそれらを適用します。

新しいシャードがシャード・データベースに追加された場合、クライアントがアクセスできるようになる前に、シャード・データベースで実行されたすべてのDDLがそのシャードに同じ順序で適用されます。

シャード・データベースでDDLを発行するには、次の2つの方法があります。

- GDSCTL SQLコマンドを使用します。

  次の例に示すように、GDSCTL SQLコマンドを使用してDDLを発行すると、GDSCTLはすべてのシャードがDDLの実行を終了するまで待機し、実行のステータスを返します。

  ```
  GDSCTL> sql “create tablespace set tbsset”
  ```

- GDS$CATALOG.sdbnameサービスを使用し、SQL*Plusを使用してシャード・カタログ・データベースに接続します。

  シャード・カタログ・データベースに対してDDLを発行すると、ローカルでの実行が完了したときにステータスが返されますが、すべてのシャードへのDDLの伝播はバックグラウンドで非同期に行われます。

  ```
  SQL> create tablespace set tbsset;
  ```

ノート:

SYS アカウントを使用したシャード DDL の実行はお薦めしません。その目的のための特権アカウントを作成します。

Oracle ShardingのDDL構文拡張の詳細は、Oracle ShardingのDDL構文拡張を参照してください。

ローカルまたはグローバルでのオブジェクトの作成

GDSCTLを使用して作成されたオブジェクトは、グローバルでシャード・データベース・オブジェクトを作成しますが、SQL*Plusを使用してシャード・カタログに接続することで、ローカルまたはグローバル・オブジェクトを作成できます。

オブジェクトを作成するDDLがGDSCTLのsqlコマンドを使用して発行された場合、オブジェクトがすべてのシャードで作成されます。オブジェクトのマスター・コピーが、シャード・カタログ・データベースにも作成されます。すべてのシャードおよびシャード・カタログ・データベースに存在するオブジェクトは、シャード・データベース・オブジェクトと呼ばれます。

SQL*Plusを使用してシャード・カタログに接続する場合、2つのタイプのオブジェクト(シャード・データベース・オブジェクトおよびローカル・オブジェクト)を作成できます。ローカル・オブジェクトは、シャード・カタログにのみ存在する従来のオブジェクトです。ローカル・オブジェクトは、管理のために使用したり、シャード・カタログ・データベースから発行されるマルチシャード問合せで(たとえば、レポートを生成および格納するために)使用したりできます。

シャード・オブジェクトは、ローカル・オブジェクトに対する依存性を持つことができません。たとえば、ローカル表に全シャード表示は作成できません。

SQL*Plusセッションで作成されるオブジェクトのタイプ(シャード・データベースまたはローカル)は、セッションでSHARD DDLモードが有効にされているかどうかによって異なります。全シャード・ユーザー(すべてのシャードおよびシャード・カタログ・データベースに存在するユーザー)の場合、このモードは、シャード・カタログ・データベースではデフォルトで有効になっています。セッションでSHARD DDLが有効になっている間に作成されるすべてのオブジェクトは、シャード・データベース・オブジェクトです。

セッションでSHARD DDLを有効にするには、全シャード・ユーザーが次を実行する必要があります。

```
ALTER SESSION ENABLE SHARD DDL
```

SHARD DDLが無効であるときに作成されたすべてのオブジェクトは、ローカル・オブジェクトです。ローカル・オブジェクトを作成するには、最初に全シャード・ユーザーを実行する必要があります

```
ALTER SESSION DISABLE SHARD DDL
```

SHARD DDLセッション・パラメータの詳細は、ALTER SESSIONを参照してください。

Oracle ShardingのDDL構文の機能拡張

Oracle Shardingには、シャード・データベースに対してのみ実行できる構文を持つSQL DDL文が含まれています。

Oracle Shardingをサポートするために問合せおよびDML文を変更する必要はなく、DDL文の変更も非常に限られています。ほとんどの既存のDDL文は、非シャード・データベースと同じ構文およびセマンティクスを使用して、シャード・データベースで同様

に機能します。

CREATE TABLESPACE SET

この文は、1つ以上のシャード表および索引の論理ストレージ・ユニットとして使用できる表領域セットを作成します。表領域セットは、シャード領域のシャード間に分散された複数のOracle表領域で構成されます。

CREATE TABLESPACE SET文はOracle Sharding専用です。構文はCREATE TABLESPACEと似ています。

```
CREATE TABLESPACE SET tablespace_set
           [IN SHARDSPACE shardspace]
                [USING TEMPLATE (
    { MINIMUM EXTENT size_clause
    | BLOCKSIZE integer [ K ]
    | logging_clause
    | FORCE LOGGING
    | ENCRYPTION tablespace_encryption_spec
    | DEFAULT [ table_compression ] storage_clause
    | { ONLINE | OFFLINE }
    | extent_management_clause
    | segment_management_clause
    | flashback_mode_clause
    }...
   )];
```

システム管理のシャーディングでは、シャード・データベースにデフォルトの1つのシャード領域のみがあります。表領域セット内の表領域の数は自動的に決定され、対応するシャード領域内のチャンクの数と同じです。

表領域セット内のすべての表領域はビッグファイル表領域であり、プロパティが同じです。プロパティはUSING TEMPLATE句で指定され、表領域セット内の単一の表領域のプロパティを記述します。この句は、通常の表領域のpermanent_tablespace_clauseと同じですが、datafile_tempfile_spec句にデータファイル名を指定できないことが異なります。表領域セット内の各表領域のデータファイル名は自動的に生成されます。

表領域セットは永続表領域のみで構成され、システム、UNDOまたは一時表領域セットはありません。また、次の例では、表領域セットのデータファイルの合計サイズは100mx Nです(Nは表領域セット内の表領域の数です)。

例

```
CREATE TABLESPACE SET TSP_SET_1 IN SHARDSPACE sgr1
USING TEMPLATE
( DATAFILE SIZE 100m
  EXTEND MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO
);
```

ALTER TABLESPACE SET

この文は、1つ以上のシャード表および索引の論理ストレージ・ユニットとして使用できる表領域セットを変更します。

表領域セットのSHARDSPACEプロパティは変更できません。表領域セットの他のすべての属性は、通常の永続表領域と同様に変更できます。表領域セット内の表領域はビッグファイルであるため、ADD DATAFILE句およびDROP DATAFILE句はサポートされません。

DROP TABLESPACE SETおよびPURGE TABLESPACE SET

これらの文は、1つ以上のシャード表および索引の論理ストレージ・ユニットとして使用できる表領域セットを削除またはパージします。

これらの文の構文およびセマンティクスは、DROP文およびPURGE TABLESPACE文と似ています。

CREATE TABLE

CREATE TABLE文は、シャード表および複製表を作成するために拡張され、表ファミリを指定します。

構文

```
CREATE [ { GLOBAL TEMPORARY | SHARDED | DUPLICATED} ]
             TABLE [ schema. ] table
      { relational_table | object_table | XMLType_table }
        [ PARENT [ schema. ] table ] ;
```

CREATE TABLE文の次の部分は、Oracle Shardingをサポートするためのものです。

- SHARDEDキーワードおよびDUPLICATEDキーワードは、表のコンテンツがシャード間でパーティション化されるか、すべてのシャードで複製されることをそれぞれ示します。DUPLICATEDキーワードは、重複表を作成するための唯一の構文の変更です。以降で説明する他のすべての変更はシャード表にのみ適用されます。
- PARENT句は、シャード表を表ファミリのルート表にリンクします。
- システムおよび複合シャーディングでは、シャード表を作成するために、TABLESPACEのかわりにTABLESPACE SETが使用されます。TABLESPACEが含まれるすべての句は、TABLESPACE SETが含まれるように拡張されました。
- 3つの句: table_partitioning_clauses内のconsistent_hash_partitions、consistent_hash_with_subpartitionsおよびpartition_set_clause。

```
table_partitioning_clauses ::=
{range_partitions
| hash_partitions
| list_partitions
| composite_range_partitions
| composite_hash_partitions
| composite_list_partitions
| reference_partitioning
| system_partitioning
| consistent_hash_partitions
| consistent_hash_with_subpartitions
| partition_set_clause
}
```

例

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3)
, signup_date DATE
,
CONSTRAINT cust_pk PRIMARY KEY(cust_id, class)
```

```
)
PARTITIONSET BY LIST (class)
PARTITION BY CONSISTENT HASH (cust_id)
PARTITIONS AUTO
(PARTITIONSET gold   VALUES ( 'gld' ) TABLESPACE SET ts2,
 PARTITIONSET silver VALUES ( 'slv' ) TABLESPACE SET ts1)
;
```

consistent_hash_with_subpartitionsの例

```
CREATE SHARDED TABLE Customers
       ( "custi_id" NUMBER NOT NULL
       , name VARCHAR2(50)
       , class VARCHAR2(3) NOT NULL
       , signup_date DATE
       ,
       CONSTRAINT cust_pk PRIMARY KEY("custi_id",name,signup_date,class)
       )
       PARTITIONSET BY LIST (class)
       PARTITION BY CONSISTENT HASH ("custi_id",name)
       SUBPARTITION BY RANGE (signup_date)
       SUBPARTITION TEMPLATE
          ( SUBPARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY'))
          , SUBPARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY'))
          , SUBPARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2020','DD/MM/YYYY'))
          , SUBPARTITION future VALUES LESS THAN (MAXVALUE))
       PARTITIONS AUTO
       (
         PARTITIONSET "gold" VALUES ('Gld','BRZ') TABLESPACE SET ts1 SUBPARTITIONS STORE
IN(TBS1,TBS2,TBS3,TBS4)
       , PARTITIONSET "silver" VALUES ('Slv','OTH') TABLESPACE SET ts2 SUBPARTITIONS STORE
IN(TBS5,TBS6,TBS7,TBS8)
       ) ;
```

制限事項

現在のリリースでのシャード表に関する制限:

- シャード表のデフォルトの表領域セットはありません。
- 一時表はシャーディングまたは複製できません。
- 索引構成シャード表はサポートされません。
- シャード表には、ネストした表の列またはアイデンティティ列を含めることはできません。
- シャード表に定義される主キー制約には、シャーディング列を含める必要があります。重複表の列を参照するシャード表の列の外部キー制約はサポートされていません。
- システム・パーティション化および間隔レンジ・パーティション化は、シャード表ではサポートされません。個別のハッシュ・パーティションの指定は、コンシステント・ハッシュによるパーティション化ではサポートされません。
- PARTITION BY句またはPARTITIONSET BY句に使用されるシャード表の列には、仮想列を指定できません。

現在のリリースの重複表では、次のことはサポートされません。

- システム・パーティション表および参照パーティション表

- LONG、抽象(MDSYSデータ型はサポートされています)、REFデータ型
- 主キーを除く列の最大数は999個です
- nologgingおよびinmemoryオプション
- 重複表のXMLType列は非ASSM表領域で使用できません

Oracle Shardingをサポートする句の使用方法の詳細は、CREATE TABLEを参照してください。

ALTER TABLE

ALTER TABLE文は、シャード表および重複表を変更するために拡張されています。

シャード・データベースに対してALTER TABLEを使用する場合は、制限事項があります。

次のオプションは、システム管理またはコンポジット・シャード・データベースのシャード表ではサポートされません。

- 名前変更
- 外部キー制約の追加
- 個々のパーティションおよびサブパーティションに対するすべての操作
- シャードに対するすべてのパーティション関連の操作(TRUNCATE PARTITION、UNUSABLE LOCAL INDEXESおよびREBUILD UNUSABLE LOCAL INDEXESを除く)

次の操作は、重複表ではサポートされていません。

- データ型: LONG、抽象(MDSYSデータ型はサポートされています)、REF
- 列のオプション: ベクトル・エンコード、不可視の列、ネストした表
- オブジェクト型
- クラスタ化表
- 外部表
- ILMポリシー
- PARENT句
- 表のフラッシュバック操作
- システムおよび参照パーティション化
- NOLOGGINGオプションの有効化
- 重複表のマテリアライズド・ビュー・ログの削除
- シャード上の重複表のマテリアライズド・ビューの削除
- シャード上の(重複表の)マテリアライズド・ビューの変更

ALTER SESSION

ALTER SESSION文は、シャード・データベースをサポートするように拡張されています。

セッションレベルのSHARD DDLパラメータは、シャード・カタログ・データベースに対して発行されたDDLのスコープを設定します。

```
ALTER SESSION { ENABLE | DISABLE } SHARD DDL;
```

SHARD DDLを有効にした場合、セッションで発行されたすべてのDDLはシャード・カタログおよびすべてのシャードで実行されます。SHARD DDLを無効にした場合、DDLはシャード・カタログ・データベースに対してのみ実行されます。シャード・データベース・ユーザー(すべてのシャードおよびカタログに存在するユーザー)の場合、SHARD DDLはデフォルトで有効にされます。シャード・データベース・ユーザーを作成するには、CREATE USERを実行する前に、SHARD DDLパラメータを有効にする必要があります。

シャード・データベースでのPL/SQLプロシージャの実行

構成内のすべてのシャードでDDL文を実行するのと同じ方法で、特定のOracle提供のPL/SQLプロシージャも実行できます。これらの特定のプロシージャ・コールは、シャードDDL文であるかのように動作します。この場合、それらはすべてのシャードに伝播され、カタログによって追跡され、新しいシャードが構成に追加されるたびに実行されます。

次のプロシージャはすべて、シャードDDL文であるかのように動作します。

- DBMS_FGAパッケージのプロシージャ
- DBMS_RLSパッケージのプロシージャ
- DBMS_STATSパッケージの次のプロシージャ:
    - GATHER_INDEX_STATS
    - GATHER_TABLE_STATS
    - GATHER_SCHEMA_STATS
    - GATHER_DATABASE_STATS
    - GATHER_SYSTEM_STATS
- DBMS_GOLDENGATE_ADMパッケージの次のプロシージャ:
    - ADD_AUTO_CDR
    - ADD_AUTO_CDR_COLUMN_GROUP
    - ADD_AUTO_CDR_DELTA_RES
    - ALTER_AUTO_CDR
    - ALTER_AUTO_CDR_COLUMN_GROUP
    - PURGE_TOMBSTONES
    - REMOVE_AUTO_CDR
    - REMOVE_AUTO_CDR_COLUMN_GROUP
    - REMOVE_AUTO_CDR_DELTA_RES

シャードDDL文と同じ方法でいずれかのプロシージャを実行するには、次のステップを実行します。

1. gsm_pooladmin_roleを持つデータベース・ユーザーとしてSQL*Plusを使用して、シャード・カタログ・データベースに接続します。
2. alter session enable shard ddlを使用してシャードDDLを有効にします。
3. SYS.EXEC_SHARD_PLSQLという名前のシャーディング固有のPL/SQLプロシージャを使用して、ターゲット・プロシージャを実行します。

   このプロシージャは、単一のCLOB引数を取ります。これは、完全修飾プロシージャ名とその引数を指定する文字列です。EXEC_SHARD_PLSQLを使用せずにターゲット・プロシージャを実行すると、そのプロシージャはカタログでのみ実行され、すべてのシャードに伝播されません。完全修飾名(たとえば、 SYS.DBMS_RLS.ADD_POLICY)を指定せずにプロシージャを実行すると、エラーが発生します。

たとえば、すべてのシャードでDBMS_RLS.ADD_POLICYを実行するには、シャードDLLを有効にした後にSQL*Plusから次を実行

します。

```
exec sys.exec_shard_plsql('sys.dbms_rls.add_policy(object_schema                =>
          ''testuser1'',

          object_name     => ''DEPARTMENTS'',

          policy_name     => ''dept_vpd_pol'',

          function_schema => ''testuser1'',

          policy_function => ''authorized_emps'',

          statement_types => ''INSERT, UPDATE, DELETE, SELECT, INDEX'',

          update_check    => TRUE)'

          ) ;
```

コール仕様自体はexec_shard_plsqlへの文字列パラメータであるため、ターゲット・プロシージャ・コール仕様内に2つの一重引用符が必要なことに注意してください。

ターゲット・プロシージャがシャード・カタログ・データベースで正しく実行されると、現在デプロイされているすべてのシャードでの処理用にキューに入れられます。カタログでのターゲット・プロシージャ実行のエラーは、SQL*Plusセッションに戻されます。シャードでの実行時のエラーは、DDLの場合と同じ方法で追跡できます。

シャード・データベースのスキーマ・オブジェクトの作成

次のトピックでは、シャード・データベースにスキーマ・オブジェクトを作成する方法を示します。これらのオブジェクトの概念については、第2章のシャード・データベースのスキーマ・オブジェクトに関する項を参照してください。

全シャード・ユーザーの作成

シャード・カタログ・データベースにのみ存在するローカル・ユーザーには、シャード・データベースにスキーマ・オブジェクトを作成するための権限がありません。シャード・データベース・スキーマを作成する最初のステップは、**全シャード・ユーザー**を作成することです。

権限のあるユーザーとしてシャード・カタログ・データベースに接続して、SHARD DDLを有効にし、CREATE USERコマンドを実行して全シャード・ユーザーを作成します。全シャード・ユーザーがシャード・カタログ・データベースに接続すると、SHARD DDLモードがデフォルトで有効になります。

ノート:

SHARD DDL モードを有効にしている場合、ローカル・ユーザーは非スキーマのシャード・データベース・オブジェクト(表領域、ディレクトリ、コンテキストなど)を作成できます。ただし、スキーマ・オブジェクト(表、ビュー、索引、ファンクション、プロシージャなど)は作成できません。

シャード・オブジェクトは、ローカル・オブジェクトに対する依存性を持つことができません。たとえば、ローカル表に全

シャード表示は作成できません。

シャード DDL を使用して、シャード・ユーザーに SYS 権限を付与することはできません。各シャードにログインし、そのシャードでアカウントに権限を手動で付与する必要があります。

シャード表ファミリの作成

SQL CREATE TABLE文を使用してシャード表ファミリを作成します。参照パーティション化または同一レベル・パーティション化を使用して、表間の親子関係を指定できます。

参照パーティション化を使用した表間の親子関係の指定

シャード表ファミリは、参照パーティション化を使用し、表間の親子関係を指定して作成することをお薦めします。

参照パーティション化の場合、ルート表のみにパーティション化スキームを指定するため構文が簡単です。また、ルート表に対して実行されるパーティション管理操作が自動的に子孫に伝播されます。たとえば、ルート表にパーティションを追加すると、すべての子孫に新しいパーティションが作成されます。

システム管理のシャーディング方法を使用したCustomers-Orders-LineItemsスキーマの適切なCREATE TABLE文を次に示します。最初の文でCustomersという表ファミリのルート表を作成します。

```
CREATE SHARDED TABLE Customers
( CustNo      NUMBER NOT NULL
, Name        VARCHAR2(50)
, Address     VARCHAR2(250)
, CONSTRAINT RootPK PRIMARY KEY(CustNo)
)
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

次の2つの文で、Customers表の子と孫であるOrders表とLineItems表を作成します。

```
CREATE SHARDED TABLE Orders
( OrderNo   NUMBER NOT NULL
, CustNo    NUMBER NOT NULL
, OrderDate DATE
, CONSTRAINT OrderPK PRIMARY KEY (CustNo, OrderNo)
, CONSTRAINT CustFK  FOREIGN KEY (CustNo) REFERENCES Customers(CustNo)
)
PARTITION BY REFERENCE (CustFK)
;

CREATE SHARDED TABLE LineItems
( CustNo    NUMBER NOT NULL
, LineNo    NUMBER(2) NOT NULL
, OrderNo   NUMBER(5) NOT NULL
, StockNo   NUMBER(4)
, Quantity  NUMBER(2)
, CONSTRAINT LinePK  PRIMARY KEY (CustNo, OrderNo, LineNo)
, CONSTRAINT LineFK  FOREIGN KEY (CustNo, OrderNo) REFERENCES Orders(CustNo, OrderNo)
)
```

```
PARTITION BY REFERENCE (LineFK)
;
```

前述の例の文では、ファミリ内のすべての表の対応するパーティションが同じ表領域セットTS1に格納されます。ただし、各表に個別の表領域セットを指定できます。

前述の例の文では、シャーディング・キーとして使用されているパーティション化列CustNoは、3つの表のすべてに存在します。これは、参照パーティション化では一般に子表にキー列を複製することなく親表を使用して子表を等価パーティション化できるという事実と異なります。この理由は、子表をその親表にリンクするために使用される子表の外部キー制約には主キーを指定する必要があるので、参照パーティション化では親表に主キーが必要となるためです。ただし、シャード表の主キーは、シャーディング・キーと同じであるか、シャーディング・キーを含む必要があります。これにより、線形スケーラビリティの重要な要件である主キーのグローバルな一意性が、他のシャードと調整することなく維持されます。

要約すると、シャード・データベースで参照パーティション表を使用するには、次のルールに従う必要があります。

- シャード表の主キーは、シャーディング・キーと同じであるか、シャーディング・キーを含む必要があります。他のシャードと調整することなく、主キーのグローバルな一意性を維持するためにこれが必要となります。
- 子表をその親表にリンクするための子表の外部キー制約には主キーを指定する必要があるため、参照パーティション化では親表に主キーが必要となります。親表にUNIQUE制約のみがあり、PRIMARY KEYがない場合は、外部キー制約を設定することもできます。シャーディング・キーもNOT NULLである必要があります。

  たとえば、LineItems (子)表をOrders (親)表にリンクするには、Orders表に主キーが必要となります。2番目のルールは、Orders表の主キーにCustNo値が含まれていることを意味します。(これはOracle Shardingに固有ではない既存のパーティション化のルールです。)

同一レベル・パーティション化を使用した表間の親子関係の指定

場合によっては、参照パーティション化に必要な主キー制約と外部キー制約を作成することが不可能か望ましくないことがあります。このような場合、表ファミリで親子関係を指定するには、すべての表が明示的に同一レベル・パーティション化されている必要があります。各子表は、親の名前を含むCREATE SHARDED TABLEのPARENT句を使用して作成されます。次に、構文の例を示します。

```
 CREATE SHARDED TABLE Customers
( CustNo      NUMBER NOT NULL
, Name        VARCHAR2(50)
, Address     VARCHAR2(250)
, region      VARCHAR2(20)
, class       VARCHAR2(3)
, signup      DATE
)
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE Orders
( OrderNo   NUMBER
, CustNo    NUMBER NOT NULL
, OrderDate DATE
)
PARENT Customers
PARTITION BY CONSISTENT HASH (CustNo)
```

```
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE LineItems
( LineNo    NUMBER
, OrderNo   NUMBER
, CustNo    NUMBER NOT NULL
, StockNo   NUMBER
, Quantity  NUMBER
)
PARENT Customers
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

すべてのCREATE SHARDED TABLE文でパーティション化スキームが完全に指定されているため、任意の表を個別にサブパーティション化できます。ルート表のみにサブパーティションを指定でき、表ファミリのすべての表のサブパーティション化スキームが同じである参照パーティション化の場合、これは許可されません。

この方法は2レベルの表ファミリのみをサポートすることに注意してください。つまり、すべての子が同じ親を持つ必要があり、孫は存在できません。親表のパーティション化列がすべての子表に存在するかぎり、これは制限にはなりません。

関連項目:

参照パーティション化の詳細は、Oracle Database VLDBおよびパーティショニング・ガイドを参照

シャード表の作成

シャード表は、複数のデータベース間でより小さい管理しやすい断片にパーティション化された表であり、シャードと呼ばれます。

表領域セットのサイズ設定

システム管理およびコンポジットのシャーディング方法では、シャード・カタログに表領域セットを作成するときに、シャード・カタログおよび各シャードに作成されたすべての表領域に対して十分な領域があることを確認する必要があります。これは、従量制の使用環境では特に重要です。

たとえば、構成にシャード・カタログと3つのシャードがある場合、次の文を発行します。

```
ALTER SESSION ENABLE SHARD DDL;
CREATE TABLESPACE SET TSP_SET_1 IN SHARDSPACE SHSPC_1 USING TEMPLATE
 (DATAFILE SIZE 100M  AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED);
```

シャードごとにデフォルトの120個のチャンクを想定すると、このコマンドは、シャード・カタログおよび各シャードにそれぞれ初期表領域100Mの360個の表領域を作成します。これは大量の記憶域のようには聞こえませんが、データベース管理者が最初に100Gを割り当てるときにはシャード当たり3.6TBとは想定していません。その量の記憶域が計画されていない場合、DDLの失敗につながる可能性があり、リカバリにかなりの労力が必要になります。

システム管理のシャード・データベースでのシャード表の作成

システム管理のシャード・データベースでは、データはコンシステント・ハッシュによるパーティション化を使用してシャード間で自動的に分散されます。

シャード表を作成する前に、CREATE TABLESPACE SETを使用して表パーティションを格納する表領域セットを作成します。

```
CREATE TABLESPACE SET ts1;
```

表領域属性をカスタマイズする必要がある場合は、この例に示すように、USING TEMPLATE句をCREATE TABLESPACE SETに追加します。

```
CREATE TABLESPACE SET ts1
USING TEMPLATE
( DATAFILE SIZE 10M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K
  SEGMENT SPACE MANAGEMENT AUTO
  ONLINE
)
;
```

CREATE SHARDED TABLEを使用してシャード表を作成し、シャーディング・キーcust_idに基づいてシャード間で表を水平方向にパーティション化します。

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, region      VARCHAR2(20)
, class       VARCHAR2(3)
, signup      DATE
CONSTRAINT cust_pk PRIMARY KEY(cust_id)
)
PARTITION BY CONSISTENT HASH (cust_id)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

システム管理のシャード表は、PARTITION BY CONSISTENT HASH (primary_key_column)を指定することで、コンシステント・ハッシュでパーティション化されます。

PARTITIONS AUTO句は、パーティションの数が表領域セットts1の表領域の数に自動的に設定され、各パーティションが別々の表領域に格納されることを指定します。

ユーザー定義シャード・データベースでのシャード表の作成

ユーザー定義シャード・データベースでは、データを個々のシャードに明示的にマップします。ユーザー定義シャード・データベースのシャード表は、レンジまたはリストでパーティション化できます。

ユーザー定義シャード表の表領域セットは作成しません。ただし、次に示すように、各表領域を個別に作成し、シャード・データベース構成にデプロイされたシャード領域に明示的に関連付ける必要があります。

```
CREATE TABLESPACE tbs1 IN SHARDSPACE west;
CREATE TABLESPACE tbs2 IN SHARDSPACE west;
CREATE TABLESPACE tbs3 IN SHARDSPACE central;
```

```
CREATE TABLESPACE tbs4 IN SHARDSPACE central;
CREATE TABLESPACE tbs5 IN SHARDSPACE east;
CREATE TABLESPACE tbs6 IN SHARDSPACE east;
```

シャード表を作成する場合は、次の例に示すように、各表領域に格納されるデータの範囲またはリストを使用してパーティションを定義します。

```
CREATE SHARDED TABLE accounts
( id             NUMBER
, account_number NUMBER
, customer_id    NUMBER
, branch_id      NUMBER
, state          VARCHAR(2) NOT NULL
, status         VARCHAR2(1)
)
PARTITION BY LIST (state)
( PARTITION p_northwest VALUES ('OR', 'WA') TABLESPACE ts1
, PARTITION p_southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE ts2
, PARTITION p_northcentral VALUES ('SD', 'WI') TABLESPACE ts3
, PARTITION p_southcentral VALUES ('OK', 'TX') TABLESPACE ts4
, PARTITION p_northeast VALUES ('NY', 'VM', 'NJ') TABLESPACE ts5
, PARTITION p_southeast VALUES ('FL', 'GA') TABLESPACE ts6
)
;
```

コンポジット・シャード・データベースでのシャード表の作成

コンポジット・シャーディング方法を使用するシャード・データベースでは、コンシステント・ハッシュでパーティション化された表のキー値の範囲またはリストに対応するデータのサブセットをパーティション化できます。

コンポジット・シャーディングでは、他のシャーディング方法と同様に、表領域を使用してシャードへのパーティションのマッピングを指定します。シャード表のデータのサブセットをパーティション化するには、次の例に示すように、シャード・データベース構成にデプロイされたシャード領域ごとに個別の表領域セットを作成する必要があります。

```
CREATE TABLESPACE SET tbs1 IN SHARDSPACE shspace1;
CREATE TABLESPACE SET tbs2 IN SHARDSPACE shspace2;
```

次の例に示す文では、サービスのクラスに基づいて、シャード表をgoldとsilverという2つのパーティション・セットにパーティション化しています。各パーティション・セットが個別の表領域に格納されます。次に、各パーティション・セットのデータが、顧客IDのコンシステント・ハッシュによってさらにパーティション化されます。

```
CREATE SHARDED TABLE customers
( cust_id NUMBER NOT NULL
, name VARCHAR2(50)
, address VARCHAR2(250)
, location_id VARCHAR2(20)
, class VARCHAR2(3)
, signup_date DATE
, CONSTRAINT cust_pk PRIMARY KEY(cust_id, class)
)
PARTITIONSET BY LIST (class)
  PARTITION BY CONSISTENT HASH (cust_id)
  PARTITIONS AUTO
(PARTITIONSET gold VALUES ( ‘gld’ ) TABLESPACE SET tbs1,
```

```
 PARTITIONSET silver VALUES ( 'slv' ) TABLESPACE SET tbs2)
;
```

重複表の作成

単一のシャードによって処理されるデータベース・リクエストの数は、すべてのシャード間で読取り専用または読取りの大部分の表を複製することで最大化できます。この方法は、頻繁には更新されず、シャード表とともにアクセスされることが多い比較的小さい表に適しています。

重複表にはいくつかの制限事項があります。次の操作は、重複表ではサポートされていません。

- NOLOGGING
- ALTER TABLE ADD/DROP CONSTRAINT: 主キーのみ
- ALTER TABLE ADD/DROP PRIMARY KEY
- ALTER TABLE RENAME COLUMN
- PARTITION BY REFERENCE
- PARTITION BY SYSTEM
- CLUSTERED TABLE
- 非最終UDTまたはNESTED TABLE
- LONG DATATYPE
- COLUMN VECTOR ENCODE
- INVISIBLE COLUMN
- 列の暗号化
- 情報ライフサイクル管理(ILM)ポリシー
- CTASパラレル
- 通常、重複表とシャード表の間の外部キー制約は許可されませんが、ユーザー定義のシャーディングでは、シャード表と重複表の間にDISABLE NOVALIDATE外部キー制約を作成できます。

重複表Productsは次の文を使用して作成できます。

```
CREATE DUPLICATED TABLE Products
( StockNo     NUMBER PRIMARY KEY
, Description VARCHAR2(20)
, Price       NUMBER(6,2))
;
```

重複表の更新とその内容の同期

Oracle Shardingは、マテリアライズド・ビュー・レプリケーションを使用して重複表の内容を同期します。

各シャードの重複表がマテリアライズド・ビューとして表示されます。マテリアライズド・ビューのマスター表はシャード・カタログにあります。CREATE DUPLICATED TABLE文によって、マスター表、マテリアライズド・ビュー、およびマテリアライズド・ビューのレプリケーションに必要なその他のオブジェクトが自動的に作成されます。

任意のシャードに接続し、シャード上の重複表を直接更新できます。更新は、まずデータベース・リンクを介してシャードからシャード・カタログのマスター表に伝播されます。その後、マテリアライズド・ビューのリフレッシュの結果として、更新が他のすべてのシャードに非同期的に伝播されます。

すべてのシャードのマテリアライズド・ビューは、次のいずれかのオプションを使用してリフレッシュできます。

- 表ごとの構成可能な頻度での自動リフレッシュ
- ストアド・プロシージャの実行によるオンデマンド・リフレッシュ

自動リフレッシュでは、リフレッシュのパフォーマンスを向上させるために、ストアド・プロシージャ・インタフェースを使用してマテリアライズド・ビューのリフレッシュ・グループを作成することもできます。

ノート:

シャードで実行されるトランザクションがシャード・カタログで削除された行を更新しようとすると、競合状態になる可能性があります。この場合、エラーが戻され、シャードのトランザクションがロールバックされます。

シャード上の重複表を更新する場合、次のユース・ケースはサポートされません。

- データベース・リンクでサポートされていない LOB またはデータ型の更新
- 同じトランザクションによって挿入された行の更新または削除

スキーマの作成例

次の例は、システム管理、ユーザー定義方法およびコンポジット・シャーディング方法を使用してシャード・データベースのスキーマを作成するステップを示しています。

システム管理シャード・データベース・スキーマの作成

システム管理シャーディング方法を使用するシャード・データベースの表領域セット、シャード表および重複表を作成します。

1. シャード・カタログ・データベースに接続し、アプリケーション・スキーマのユーザーを作成して、そのユーザーに権限とロールを付与します。

   この例では、アプリケーション・スキーマのユーザーはapp_schemaという名前です。

   ```
   $ sqlplus / as sysdba
   SQL> alter session enable shard ddl;
   SQL> create user app_schema identified by app_schema_password;
   SQL> grant all privileges to app_schema;
   SQL> grant gsmadmin_role to app_schema;
   SQL> grant select_catalog_role to app_schema;
   SQL> grant connect, resource to app_schema;
   SQL> grant dba to app_schema;
   SQL> grant execute on dbms_crypto to app_schema;
   ```

2. シャード表の表領域セットを作成します。

   ```
   SQL> CREATE TABLESPACE SET TSP_SET_1 using template
    (datafile size 100m autoextend on next 10M maxsize unlimited
     extent management local segment space management auto);
   ```

3. 列にLOBを使用する場合は、LOBの表領域セットを指定できます。

   ```
   SQL> CREATE TABLESPACE SET LOBTS1;
   ```

ノート:

システム管理のシャーディングでは、LOB の表領域セットをサブパーティション・レベルで指定できません。

4. 重複表の表領域を作成します。

この例では、重複表はサンプルのCustomers-Orders-ProductsスキーマのProducts表です。

```
SQL> CREATE TABLESPACE products_tsp datafile size 100m
 autoextend on next 10M maxsize unlimited
 extent management local uniform size 1m;
```

5. ルート表のシャード表を作成します。

この例では、ルート表はサンプルのCustomers-Orders-ProductsスキーマのCustomers表です。

```
SQL> CONNECT app_schema/app_schema_password
SQL> CREATE SHARDED TABLE Customers
  (
    CustId      VARCHAR2(60) NOT NULL,
    FirstName   VARCHAR2(60),
    LastName    VARCHAR2(60),
    Class       VARCHAR2(10),
    Geo         VARCHAR2(8),
    CustProfile VARCHAR2(4000),
    Passwd      RAW(60),
    CONSTRAINT pk_customers PRIMARY KEY (CustId),
    CONSTRAINT json_customers CHECK (CustProfile IS JSON)
  ) TABLESPACE SET TSP_SET_1
  PARTITION BY CONSISTENT HASH (CustId) PARTITIONS AUTO;
```

ノート:

列に LOB が含まれる場合は、次に示すように親表の作成文に表領域セットを含めることができます。

```
SQL> CREATE SHARDED TABLE Customers
  (
    CustId      VARCHAR2(60) NOT NULL,
    FirstName   VARCHAR2(60),
    LastName    VARCHAR2(60),
    Class       VARCHAR2(10),
    Geo         VARCHAR2(8),
    CustProfile VARCHAR2(4000),
    Passwd      RAW(60),
    image       BLOB,
    CONSTRAINT pk_customers PRIMARY KEY (CustId),
    CONSTRAINT json_customers CHECK (CustProfile IS JSON)
  ) TABLESPACE SET TSP_SET_1
 LOB(image) store as (TABLESPACE SET LOBTS1)
  PARTITION BY CONSISTENT HASH (CustId) PARTITIONS AUTO;
```

6. 表ファミリの他の表のシャード表を作成します。

この例では、シャード表はサンプルのCustomers-Orders-ProductsスキーマのOrders表およびLineItems表とし

て作成されます。

Ordersシャード表を最初に作成します。

```
SQL> CREATE SHARDED TABLE Orders
  (
    OrderId     INTEGER NOT NULL,
    CustId      VARCHAR2(60) NOT NULL,
    OrderDate   TIMESTAMP NOT NULL,
    SumTotal    NUMBER(19,4),
    Status      CHAR(4),
    CONSTRAINT  pk_orders PRIMARY KEY (CustId, OrderId),
    CONSTRAINT  fk_orders_parent FOREIGN KEY (CustId)
    REFERENCES Customers ON DELETE CASCADE
  ) PARTITION BY REFERENCE (fk_orders_parent);
```

OrderId列に使用される順序を作成します。

```
SQL> CREATE SEQUENCE Orders_Seq;
```

LineItemsのシャード表を作成します。

```
SQL> CREATE SHARDED TABLE LineItems
  (
    OrderId     INTEGER NOT NULL,
    CustId      VARCHAR2(60) NOT NULL,
    ProductId   INTEGER NOT NULL,
    Price       NUMBER(19,4),
    Qty         NUMBER,
    CONSTRAINT  pk_items PRIMARY KEY (CustId, OrderId, ProductId),
    CONSTRAINT  fk_items_parent FOREIGN KEY (CustId, OrderId)
    REFERENCES Orders ON DELETE CASCADE
  ) PARTITION BY REFERENCE (fk_items_parent);
```

7. 必要な重複表を作成します。

この例では、Products表は重複しているオブジェクトです。

```
SQL> CREATE DUPLICATED TABLE Products
  (
    ProductId  INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Name       VARCHAR2(128),
    DescrUri   VARCHAR2(128),
    LastPrice  NUMBER(19,4)
  ) TABLESPACE products_tsp;
```

次に、DDL実行を監視し、表領域セット、表およびチャンクがすべてのシャードで正しく作成されたことを確認する必要があります。

ユーザー定義シャード・データベース・スキーマの作成

ユーザー定義シャーディング方法を使用するシャード・データベースのスキーマ・ユーザー、表領域セット、シャード表および重複表を作成します。

1. シャード・カタログ・データベースに接続し、アプリケーション・スキーマのユーザーを作成して、そのユーザーに権限とロールを付与します。

この例では、アプリケーション・スキーマのユーザーはapp_schemaという名前です。

```
$ sqlplus / as sysdba
SQL> alter session enable shard ddl;
SQL> create user app_schema identified by app_schema_password;
SQL> grant all privileges to app_schema;
SQL> grant gsmadmin_role to app_schema;
SQL> grant select_catalog_role to app_schema;
SQL> grant connect, resource to app_schema;
SQL> grant dba to app_schema;
SQL> grant execute on dbms_crypto to app_schema;
```

2. シャード表の表領域を作成します。

```
SQL> CREATE TABLESPACE ck1_tsp DATAFILE SIZE 100M autoextend on next 10M maxsize
unlimited extent management local segment space management auto in
 shardspace shspace1;
SQL> CREATE TABLESPACE ck2_tsp DATAFILE SIZE 100M autoextend on next 10M maxsize
unlimited extent management local segment space management auto in
 shardspace shspace2;
```

3. 列にLOBを使用する場合は、LOBの表領域を指定できます。

```
SQL> CREATE TABLESPACE lobts1 ... in shardspace shspace1;
SQL> CREATE TABLESPACE lobts2 ... in shardspace shspace2;
```

4. 重複表の表領域を作成します。

この例では、重複表はサンプルのCustomers-Orders-Productsスキーマの Products表です。

```
SQL> CREATE TABLESPACE products_tsp datafile size 100m autoextend
 on next 10M maxsize unlimited extent management local uniform size 1m;
```

5. ルート表のシャード表を作成します。

この例では、ルート表はサンプルのCustomers-Orders-Productsスキーマの Customers表です。

```
SQL> CONNECT app_schema/app_schema_password

SQL> ALTER SESSION ENABLE SHARD DDL;
SQL> CREATE SHARDED TABLE Customers
  (
    CustId      VARCHAR2(60) NOT NULL,
    CustProfile VARCHAR2(4000),
    Passwd      RAW(60),
    CONSTRAINT pk_customers PRIMARY KEY (CustId),
    CONSTRAINT json_customers CHECK (CustProfile IS JSON)
  ) PARTITION BY RANGE (CustId)
  ( PARTITION ck1 values less than ('m') tablespace ck1_tsp,
    PARTITION ck2 values less than (MAXVALUE) tablespace ck2_tsp
  );
```

ノート:

シャード表の列に LOB が含まれる場合は、次に示すように CREATE SHARDED TABLE 文に LOB

表領域を含めることができます。

```
SQL> CREATE SHARDED TABLE Customers
  (
    CustId      VARCHAR2(60) NOT NULL,
    CustProfile VARCHAR2(4000),
    Passwd      RAW(60),
    image       BLOB,
    CONSTRAINT pk_customers PRIMARY KEY (CustId),
    CONSTRAINT json_customers CHECK (CustProfile IS JSON)
  ) PARTITION BY RANGE (CustId)
  ( PARTITION ck1 values less than ('m') tablespace ck1_tsp
      lob(image) store as (tablespace lobts1),
    PARTITION ck2 values less than (MAXVALUE) tablespace ck2_tsp
      lob(image) store as (tablespace lobts2)
  );
```

6. 表ファミリの他の表のシャード表を作成します。

この例では、シャード表はサンプルのCustomers-Orders-ProductsスキーマのOrders表およびLineItems表として作成されます。

Ordersシャード表を最初に作成します。

```
SQL> CREATE SHARDED TABLE Orders
  (
    OrderId     INTEGER NOT NULL,
    CustId      VARCHAR2(60) NOT NULL,
    OrderDate   TIMESTAMP NOT NULL,
    SumTotal    NUMBER(19,4),
    Status      CHAR(4),
    CONSTRAINT  pk_orders PRIMARY KEY (CustId, OrderId),
    CONSTRAINT  fk_orders_parent FOREIGN KEY (CustId)
    REFERENCES Customers ON DELETE CASCADE
  ) PARTITION BY REFERENCE (fk_orders_parent);
```

OrderId列に使用される順序を作成します。

```
SQL> CREATE SEQUENCE Orders_Seq;
```

LineItemsのシャード表を作成します。

```
SQL> CREATE SHARDED TABLE LineItems
  (
    OrderId     INTEGER NOT NULL,
    CustId      VARCHAR2(60) NOT NULL,
    ProductId   INTEGER NOT NULL,
    Price       NUMBER(19,4),
    Qty         NUMBER,
    CONSTRAINT  pk_items PRIMARY KEY (CustId, OrderId, ProductId),
    CONSTRAINT  fk_items_parent FOREIGN KEY (CustId, OrderId)
    REFERENCES Orders ON DELETE CASCADE
  ) PARTITION BY REFERENCE (fk_items_parent);
```

7. 必要な重複表を作成します。

この例では、Products表は重複しているオブジェクトです。

```
SQL> CREATE DUPLICATED TABLE Products
  (
    ProductId  INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Name       VARCHAR2(128),
    DescrUri   VARCHAR2(128),
    LastPrice  NUMBER(19,4)
  ) TABLESPACE products_tsp;
```

次に、DDL実行を監視し、表領域セット、表およびチャンクがすべてのシャードで正しく作成されたことを確認する必要があります。

コンポジット・シャード・データベース・スキーマの作成

コンポジット・シャーディング方法を使用するシャード・データベースのスキーマ・ユーザー、表領域セット、シャード表および重複表を作成します。

1. シャード・カタログのホストに接続し、ORACLE_SIDにシャード・カタログ名を設定します。
2. シャード・カタログ・データベースに接続し、アプリケーション・スキーマのユーザーを作成して、そのユーザーに権限とロールを付与します。

   この例では、アプリケーション・スキーマのユーザーはapp_schemaという名前です。

```
$ sqlplus / as sysdba
SQL> connect / as sysdba
SQL> alter session enable shard ddl;
SQL> create user app_schema identified by app_schema_password;
SQL> grant connect, resource, alter session to app_schema;
SQL> grant execute on dbms_crypto to app_schema;
SQL> grant create table, create procedure, create tablespace,
 create materialized view to app_schema;
SQL> grant unlimited tablespace to app_schema;
SQL> grant select_catalog_role to app_schema;
SQL> grant all privileges to app_schema;
SQL> grant gsmadmin_role to app_schema;
SQL> grant dba to app_schema;
```

3. シャード表の表領域セットを作成します。

```
SQL> CREATE TABLESPACE SET
  TSP_SET_1 in shardspace cust_america using template
  (datafile size 100m autoextend on next 10M maxsize
   unlimited extent management
   local segment space management auto );
SQL> CREATE TABLESPACE SET
  TSP_SET_2 in shardspace cust_europe using template
  (datafile size 100m autoextend on next 10M maxsize
   unlimited extent management
   local segment space management auto );
```

4. 列にLOBを使用する場合は、LOBの表領域セットを指定できます。

```
SQL> CREATE TABLESPACE SET LOBTS1 in shardspace cust_america ... ;
SQL> CREATE TABLESPACE SET LOBTS2 in shardspace cust_europe ... ;
```

ノート:

コンポジット・シャーディングでは、LOB の表領域セットをサブパーティション・レベルで指定できません。

5. 重複表の表領域を作成します。

この例では、重複表はサンプルのCustomers-Orders-ProductsスキーマのProducts表です。

```
CREATE TABLESPACE products_tsp datafile size 100m autoextend on next 10M
 maxsize unlimited extent management local uniform size 1m;
```

6. ルート表のシャード表を作成します。

この例では、ルート表はサンプルのCustomers-Orders-ProductsスキーマのCustomers表です。

```
connect app_schema/app_schema_password
alter session enable shard ddl;
CREATE SHARDED TABLE Customers
(
  CustId      VARCHAR2(60) NOT NULL,
  FirstName   VARCHAR2(60),
  LastName    VARCHAR2(60),
  Class       VARCHAR2(10),
  Geo         VARCHAR2(8),
  CustProfile VARCHAR2(4000),
  Passwd      RAW(60),
  CONSTRAINT pk_customers PRIMARY KEY (CustId),
  CONSTRAINT json_customers CHECK (CustProfile IS JSON)
) partitionset by list(GEO)
partition by consistent hash(CustId)
partitions auto
(partitionset america values ('AMERICA') tablespace set tsp_set_1,
partitionset europe values ('EUROPE') tablespace set tsp_set_2
);
```

ノート:

シャード表の列に LOB が含まれる場合は、次に示すように CREATE SHARDED TABLE 文に LOB 表領域セットを含めることができます。

```
CREATE SHARDED TABLE Customers
(
  CustId      VARCHAR2(60)   NOT NULL,
  FirstName   VARCHAR2(60),
  LastName    VARCHAR2(60),
  Class       VARCHAR2(10),
  Geo         VARCHAR2(8)    NOT NULL,
  CustProfile VARCHAR2(4000),
  Passwd      RAW(60),
  image       BLOB,
  CONSTRAINT pk_customers PRIMARY KEY (CustId),
  CONSTRAINT json_customers CHECK (CustProfile IS JSON)
) partitionset by list(GEO)
partition by consistent hash(CustId)
partitions auto
```

```
(partitionset america values ('AMERICA') tablespace set tsp_set_1
 lob(image) store as (tablespace set lobts1),
partitionset europe values ('EUROPE') tablespace set tsp_set_2
 lob(image) store as (tablespace set lobts2));
```

7. 表ファミリの他の表のシャード表を作成します。

   この例では、シャード表はサンプルのCustomers-Orders-ProductsスキーマのOrders表およびLineItems表として作成されます。

   OrderId列に使用される順序を作成します。

```
CREATE SEQUENCE Orders_Seq;
```

   Ordersシャード表を最初に作成します。

```
CREATE SHARDED TABLE Orders
(
  OrderId     INTEGER NOT NULL,
  CustId      VARCHAR2(60) NOT NULL,
  OrderDate   TIMESTAMP NOT NULL,
  SumTotal    NUMBER(19,4),
  Status      CHAR(4),
  constraint  pk_orders primary key (CustId, OrderId),
  constraint  fk_orders_parent foreign key (CustId)
    references Customers on delete cascade
) partition by reference (fk_orders_parent);
```

   LineItemsのシャード表を作成します。

```
CREATE SHARDED TABLE LineItems
(
  OrderId     INTEGER NOT NULL,
  CustId      VARCHAR2(60) NOT NULL,
  ProductId   INTEGER NOT NULL,
  Price       NUMBER(19,4),
  Qty         NUMBER,
  constraint  pk_items primary key (CustId, OrderId, ProductId),
  constraint  fk_items_parent foreign key (CustId, OrderId)
    references Orders on delete cascade
) partition by reference (fk_items_parent);
```

8. 必要な重複表を作成します。

   この例では、Products表は重複しているオブジェクトです。

```
CREATE DUPLICATED TABLE Products
(
  ProductId  INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  Name       VARCHAR2(128),
  DescrUri   VARCHAR2(128),
  LastPrice  NUMBER(19,4)
) tablespace products_tsp;
```

次に、DDL実行を監視し、表領域セット、表およびチャンクがすべてのシャードで正しく作成されたことを確認する必要があります。

DDL実行の監視およびオブジェクト作成の検証

GDSCTLおよびSQLを使用してDDL実行を監視し、DDLがすべてのシャードに伝播されていることを確認できます。

DDL実行のモニター

シャードへのDDLの伝播のステータスを確認するには、GDSCTLのshow ddlコマンドおよびconfig shardコマンドを使用できます。

シャード・カタログでSQL*Plusを使用してDDLを実行した場合、SQL*Plusではすべてのシャードの実行ステータスが返されないため、この確認は必須です。

show ddlコマンドの出力は切り捨てられることがあります。出力内容の完全なテキストを表示するには、シャード・カタログでSELECT ddl_text FROM gsmadmin_internal.ddl_requestsを実行します。

シャード・ディレクタ・ホストから次のコマンドを実行します。

```
GDSCTL> show ddl
id    DDL Text                                 Failed shards
--    --------                                 -------------
5     grant connect, resource to app_schema
6     grant dba to app_schema
7     grant execute on dbms_crypto to app_s...
8     CREATE TABLESPACE SET  TSP_SET_1 usin...
9     CREATE TABLESPACE products_tsp datafi...
10    CREATE SHARDED TABLE Customers (   Cu...
11    CREATE SHARDED TABLE Orders (   Order...
12    CREATE SEQUENCE Orders_Seq;
13    CREATE SHARDED TABLE LineItems (   Or...
14    CREATE MATERIALIZED VIEW "APP_SCHEMA"...
```

ここに示すように、構成内の各シャードでconfig shardコマンドを実行し、コマンド出力の「Last Failed DDL」行を書き留めます。

```
GDSCTL> config shard -shard sh1
Name: sh1
Shard Group: primary_shardgroup
Status: Ok
State: Deployed
Region: region1
Connection string: shard_host_1:1521/sh1_host:dedicated
SCAN address:
ONS remote port: 0
Disk Threshold, ms: 20
CPU Threshold, %: 75
Version: 18.0.0.0
Last Failed DDL:
DDL Error: ---
Failed DDL id:
Availability: ONLINE
Supported services
------------------------
Name                                          Preferred Status
----                                          --------- ------
oltp_ro_srvc                                  Yes       Enabled
```

```
oltp_rw_srvc                                Yes        Enabled
```

表領域セット作成の検証

シャード表ファミリのために作成した表領域セットの表領域、および重複表のために作成した表領域が、すべてのシャードに作成されたことを確認します。

C006TSP_SET_1を介してC001TSP_SET_1として次に示す表領域セットの表領域の数は、シャード・データベース構成がデプロイされたときにGDSCTL create shardcatalogコマンドで指定されたチャンクの数に基づきます。

重複Products表領域をPRODUCTS_TSPとして次に示します。

次に示すように、構成内のすべてのシャードでSELECT TABLESPACE_NAMEを実行します。

```
$ sqlplus / as sysdba
SQL> select TABLESPACE_NAME, BYTES/1024/1024 MB from sys.dba_data_files
 order by tablespace_name;
TABLESPACE_NAME                     MB
------------------------ ----------
C001TSP_SET_1           100
C002TSP_SET_1           100
C003TSP_SET_1                        100
C004TSP_SET_1                        100
C005TSP_SET_1                        100
C006TSP_SET_1                        100
PRODUCTS_TSP            100
SYSAUX                  650
SYSTEM                  890
SYS_SHARD_TS                         100
TSP_SET_1                                100
TABLESPACE_NAME                     MB
------------------------ ----------
UNDOTBS1                                  105
USERS                                               5
13 rows selected.
```

チャンクの作成および分散の検証

チャンクおよびチャンク表領域がすべてのシャードに作成されたことを確認します。

ここに示すようにGDSCTL config chunksコマンドを実行し、各シャードのチャンクIDの範囲をノートにとります。

```
GDSCTL> config chunks
Chunks
------------------------
Database                    From      To
--------                    ----      --
sh1                         1         6
sh2                         1         6
sh3                         7         12
sh4                         7         12
```

次に示すように、構成内の各シャードで次のSQL文を実行します。

```
SQL> show parameter db_unique_name
NAME             TYPE        VALUE
```

```
---------------- ----------- ------------------------------
db_unique_name   string      sh1
SQL> select table_name, partition_name, tablespace_name
 from dba_tab_partitions
 where tablespace_name like 'C%TSP_SET_1'
 order by tablespace_name;
TABLE_NAME       PARTITION_NAME   TABLESPACE_NAME
---------------- ---------------- --------------------
ORDERS           CUSTOMERS_P1     C001TSP_SET_1
CUSTOMERS        CUSTOMERS_P1     C001TSP_SET_1
LINEITEMS        CUSTOMERS_P1     C001TSP_SET_1
CUSTOMERS        CUSTOMERS_P2     C002TSP_SET_1
LINEITEMS        CUSTOMERS_P2     C002TSP_SET_1
ORDERS           CUSTOMERS_P2     C002TSP_SET_1
CUSTOMERS        CUSTOMERS_P3     C003TSP_SET_1
ORDERS           CUSTOMERS_P3     C003TSP_SET_1
LINEITEMS        CUSTOMERS_P3     C003TSP_SET_1
ORDERS           CUSTOMERS_P4     C004TSP_SET_1
CUSTOMERS        CUSTOMERS_P4     C004TSP_SET_1
TABLE_NAME       PARTITION_NAME   TABLESPACE_NAME
---------------- ---------------- --------------------
LINEITEMS        CUSTOMERS_P4     C004TSP_SET_1
CUSTOMERS        CUSTOMERS_P5     C005TSP_SET_1
LINEITEMS        CUSTOMERS_P5     C005TSP_SET_1
ORDERS           CUSTOMERS_P5     C005TSP_SET_1
CUSTOMERS        CUSTOMERS_P6     C006TSP_SET_1
LINEITEMS        CUSTOMERS_P6     C006TSP_SET_1
ORDERS           CUSTOMERS_P6     C006TSP_SET_1
18 rows selected.
```

シャード・カタログ・データベースに接続し、次に示すようにチャンクが均一に分散されていることを確認します。

```
$ sqlplus / as sysdba
SQL> SELECT a.name Shard, COUNT(b.chunk_number) Number_of_Chunks
  FROM gsmadmin_internal.database a, gsmadmin_internal.chunk_loc b
  WHERE a.database_num=b.database_num
  GROUP BY a.name
  ORDER BY a.name;
SHARD                          NUMBER_OF_CHUNKS
------------------------------ ----------------
sh1                                           6
sh2                                           6
sh3                                           6
sh4                                           6
```

表の作成の検証

シャード表および重複表が作成されたことを確認するには、シャード・カタログ・データベースおよび各シャードにアプリケーション・スキーマ・ユーザーとしてログインし、次のapp_schemaユーザーの例に示すように、データベース・シャードの表を問い合せます。

```
$ sqlplus app_schema/app_schema_password
Connected.
SQL> select table_name from user_tables;
TABLE_NAME
--------------------------------------------------------------------------------
CUSTOMERS
ORDERS
```

```
LINEITEMS
PRODUCTS
4 rows selected.
```

DDL実行の失敗およびリカバリの例

次の例は、DDLを発行して実行ステータスを監視するステップ、およびエラーが発生した場合の対処方法を示しています。

シャードでDDLが失敗した場合、その失敗が解決されてGDSCTLのrecover shardコマンドが実行されるまで、そのシャードに対するその後のすべてのDDLはブロックされます。

これらのGDSCTLコマンドを実行するには、GSM_ADMIN権限が必要です。

次の例では、SQL*Plusを使用してDDLが発行されたが、GDSCTL SQLコマンドを使用した場合に同じステータス・チェックおよび修正処理が適用される場合を示します。

例5-1 シャード・カタログでのDDL実行エラー

この例では、ユーザーがCREATE USERコマンドで入力ミスをしています。

```
SQL> alter session enable shard ddl;
Session altered.
SQL> CREATE USER example_user IDENTRIFIED BY out_standing1;
CREATE USER example_user IDENTRIFIED BY out_Standing1
                         *
ERROR at line 1:
ORA-00922: missing or invalid option
```

シャード・カタログでDDLの実行が失敗し、すべてのシャードでDDLが実行されなかったことがGDSCTLのshow ddlコマンドに予想どおり表示されます。

```
GDSCTL> show ddl
id      DDL Text                          Failed shards
--      --------                          -------------
```

その後、ユーザーが正しいスペリングでコマンドを実行し直します。同じセッションを使用しているため、alter session enable shard ddlを再度実行する必要はありません。

```
SQL> CREATE USER example_user IDENTIFIED BY out_Standing1;
User created.
```

今度は、DDLがシャード・カタログ・データベースで正常に実行され、オンラインのシャードで失敗していないことがshow ddlに示されます。

```
GDSCTL> show ddl
id      DDL Text                                      Failed shards
--      --------                                      -------------
1       create user example_user identified by *****
```

ノート:

DDL の実行時に停止していたシャードの場合、DDL はそのシャードがアクセス可能になったときに自動的に適用されます。

例5-2 シャードで修正のための対処を実行することによって、シャードでのエラーをリカバリする

この例では、ユーザーはシステム管理のシャード表のための表領域セットを作成しようとします。ただし、1つのシャードのデータファイル・ディレクトリが書込み可能ではないため、DDLはカタログでは正常に実行されますが、そのシャードでは失敗します。

```
SQL> connect example_user/out_Standing1
Connected
SQL> create tablespace set tbsset;
Tablespace created.
```

ユーザーexample_userはSDBユーザーとして作成され、SHARD DDLがデフォルトで有効にされるため、alter session enable shard ddlを実行する必要はありません。

GDSCTLのshow ddlを使用してステータスを確認します。

```
GDSCTL> show ddl
id      DDL Text                                  Failed shards
--      --------                                  -------------
1       create user example_user identified by *****
2       create tablespace set tbsset              shard01
```

このコマンド出力は、シャードshard01でDDLが失敗したことを示しています。GDSCTLのconfig shardコマンドを実行して、詳細情報を取得します。

```
GDSCTL> config shard -shard shard01
Conversion = ':'Name: shard01
Shard Group: dbs1
Status: Ok
State: Deployed
Region: east
Connection string: (DESCRIPTION=(ADDRESS=(HOST=shard01-host)(PORT=1521)(PROTOCOL=tcp))
(CONNECT_DATA=(SID=shard01)))
SCAN address:
ONS remote port: 0
Disk Threshold, ms: 20
CPU Threshold, %: 75
Version: 18.0.0.0
Failed DDL: create tablespace set tbsset
DDL Error: ORA-02585: create tablepsace set failure, one of its tablespaces not created
ORA-01119: error in creating database file ¥'/ade/b/3667445372/oracle/rdbms/dbs/
SHARD01/datafile/o1_mf_tbsset_%u_.dbf¥'
ORA-27040: file create error, unable to create file
Linux-x86_64 Error: 13: Permission denied
Additional information: 1 ¥(ngsmoci_execute¥)
Failed DDL id: 2
Availability: ONLINE
```

「Failed DDL:」で始まるテキストは問題を示しています。それを解決するには、ユーザーはシャード・データベースのホストにログインして、そのディレクトリを書込み可能にする必要があります。

ディレクトリの権限を表示します。

```
cd $ORACLE_HOME/rdbms/dbs
 ls -l ../ | grep dbs
dr-xr-xr-x  4 oracle dba    102400 Jul 20 15:41 dbs/
```

ディレクトリを書込み可能に変更します。

```
chmod +w .
ls -l ../ | grep dbs
drwxrwxr-x  4 oracle dba    102400 Jul 20 15:41 dbs/
```

GDSCTL コンソールに戻って、recover shardコマンドを発行します。

```
GDSCTL> recover shard -shard shard01
```

ステータスを再度確認します。

```
GDSCTL> show ddl
id      DDL Text                                  Failed shards
--      --------                                  -------------
1       create user example_user identified by *****
2       create tablespace set tbsset
GDSCTL> config shard -shard shard01
Conversion = ':'Name: shard01
Shard Group: dbs1
Status: Ok
State: Deployed
Region: east
Connection string: (DESCRIPTION=(ADDRESS=(HOST=shard01-host)(PORT=1521)(PROTOCOL=tcp))
(CONNECT_DATA=(SID=shard01)))
SCAN address:
ONS remote port: 0
Disk Threshold, ms: 20
CPU Threshold, %: 75
Version: 18.0.0.0
Last Failed DDL:
DDL Error: ---
DDL id:
Availability: ONLINE
```

この出力に示されているように、失敗したDDLエラーが表示されなくなりました。

例5-3 他のすべてのシャードで修正のための対処を実行することによって、シャードでのエラーをリカバリする

この例では、ユーザーは別の表領域セットtbs_setを作成しようとしますが、同じ名前の既存のローカル表領域がすでに存在するため、シャードでDDLが失敗します。

シャード・カタログで次のコマンドを実行します。

```
SQL> create tablespace set tbs_set;
Tablespace created.
```

GDSCTLのshow ddlコマンドを使用してステータスを確認します。

```
GDSCTL> show ddl
id      DDL Text                                  Failed shards
--      --------                                  -------------
```

```
1       create user example_user identified by *****
2       create tablespace set tbsset
3       create tablespace set tbs_set                 shard01
GDSCTL> config shard -shard shard01
Conversion = ':'Name: shard01
.....
Failed DDL: create tablespace set tbs_set
DDL Error: ORA-02585: create tablespace set failure, one of its tablespaces not created
ORA-01543: tablespace ¥'TBS_SET¥' already exists ¥(ngsmoci_execute¥)
```

この問題を解決するには、ローカル・データベース管理者としてshard01にログインし、表領域TBS_SETを削除して、GDSCTLのrecover shard -shard shard01を実行します。ただし、この表領域を維持して、かわりに名前が競合している新しく作成した表領域セットを削除し、別の名前(tbsset2など)を持つ別の表領域セットを作成するとします。次の例では、シャード・カタログで実行する方法を示します。

```
SQL> drop tablespace set tbs_set;
SQL> create tablespace set tbs_set2;
```

GDSCTLを使用してステータスを確認します。

```
GDSCTL> show ddl
id      DDL Text                                  Failed shards
--      --------                                  -------------
1       create user example_user identified by *****
2       create tablespace set tbsset
3       create tablespace set tbs_set             shard01
4       drop tablespace set tbs_set
5       create tablespace set tbsset2
```

DDL 3がshard01で失敗したため、DDL 4および5はshard01で実行されなかったことがわかります。このシャードがシャード・カタログと整合性を持つようにするには、GDSCTLのrecover shardコマンドを実行する必要があります。ただし、DDL 3は再び失敗し、実際には表領域セットtbs_setをもう作成するつもりがないため、このシャードでDDL 3を実行することは意味がありません。DDL 3をスキップするために、-ignore_firstオプションを指定してrecover shardを実行します。

```
GDSCTL> recover shard -shard shard01 -ignore_first
GSM Errors: dbs1 shard01:ORA-00959: tablespace ¥'TBS_SET¥' does not exist
 (ngsmoci_execute)
GDSCTL> show ddl
id      DDL Text                                  Failed shards
--      --------                                  -------------
1       create user sidney identified by *****
2       create tablespace set tbsset
3       create tablespace set tbs_set
4       drop tablespace set tbs_set               shard01
5       create tablespace set tbsset2
```

DDL 3がスキップされたため、今度はDDL 3で失敗しません。ただし、次のDDL (4 - drop tablespace set tbs_set)が適用され、削除する表領域セットがシャードに存在しないため、エラーとなります。

-ignore_firstオプションは最初のDDLのみをスキップするため、recover shardを再び実行してdrop文もスキップする必要があります。

```
GDSCTL> recover shard -shard shard01 -ignore_first
```

```
GDSCTL> show ddl
id      DDL Text                                 Failed shards
--      --------                                 -------------
1       create user sidney identified by *****
2       create tablespace set tbsset
3       create tablespace set tbs_set
4       drop tablespace set tbs_set
5       create tablespace set tbsset2
```

失敗は表示されなくなり、すべてのDDLがシャードに正常に適用されました。

-ignore_firstオプションを指定してrecover shardを実行した場合、失敗したDDLは増分デプロイメントで無視とマークされます。このため、新しいシャードがSDBに追加されたとき、DDL番号3および4はスキップされ、DDL番号1、2および5のみが適用されます。

シャード間の一意の順序番号の生成

Oracle Shardingを使用すると、主キー以外の列に対してシャード間でグローバルに一意の順序番号を生成できます。これはシャード・データベースで処理されます。

customer_idがシャーディング・キーの場合、顧客は大抵、主キー以外の列(order_idなど)に対して一意のIDを生成する必要があります。この場合、この機能を使用してシャード間で一意の順序番号を生成できますが、アプリケーション内の主キー以外の特定の列のグローバルな一意性を管理する必要はありません。

この機能は、新しいオブジェクトSHARDED SEQUENCEでサポートされています。シャード順序はシャード・カタログに作成されますが、各シャード上にインスタンスがあります。各インスタンスは、他のシャードで使用される範囲とオーバーラップしない範囲に属する、単調に増加する番号を生成します。したがって、生成されたすべての番号はグローバルに一意になります。

たとえば、シャード順序を使用すると、顧客IDでシャーディングされた表に対して一意の順序番号を生成できます。顧客IDをキーとして使用してシャードへの接続を確立するアプリケーションでは、シャード順序のローカル・インスタンスを使用して、グローバルに一意の順序番号を生成できます。

シャード順序で生成された番号は、このシャードに挿入される新しい行のシャーディング・キーとしてすぐに使用できません。これは、キー値が別のシャードに属している場合に、挿入によってエラーが発生するためです。新しい行を挿入するには、まずアプリケーションでシャーディング・キーの値を生成し、それを使用して適切なシャードに接続する必要があります。シャーディング・キーの新しい値を生成する一般的な方法は、シャード・カタログで通常の(シャーディングされていない)順序を使用することです。

単一のシャーディング・キー・ジェネレータがボトルネックになっている場合、シャード順序をこの目的で使用できます。この場合、アプリケーションは(シャーディング・キーを指定せずにグローバル・サービスを使用して)ランダムなシャードに接続し、シャード順序から一意のキー値を取得してから、キー値を使用して適切なシャードに接続する必要があります。

この機能をサポートするために、次のCREATE文の構文に示すように、新しいSEQUENCEオブジェクト句、SHARDおよびNOSHARDがSEQUENCEオブジェクトのDDL構文に含まれます。

```
CREATE | ALTER SEQUENCE [ schema. ]sequence
   [ { INCREMENT BY | START WITH } integer
   | { MAXVALUE integer | NOMAXVALUE }
   | { MINVALUE integer | NOMINVALUE }
```

```
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
  | { SCALE {EXTEND | NOEXTEND} | NOSCALE}
  | { SHARD {EXTEND | NOEXTEND} | NOSHARD}
  ]
```

NOSHARDは、順序のデフォルトです。SHARD句を指定すると、このプロパティは順序オブジェクトのディクショナリ表に登録され、DBA_SEQUENCES、USER_SEQUENCESおよびALL_SEQUENCESビューを使用して表示されます。

SHARDを指定すると、EXTENDおよびNOEXTEND句によってシャード順序の動作が定義されます。EXTENDを指定すると、生成された順序値はすべての長さ(x+y)になります。xは順序値の先頭に付加されるサイズ4のSHARDオフセット(最大シャード数の幅に対応、つまり1000)の長さ、yは順序MAXVALUE/MINVALUEの最大桁数です。

SHARD句のデフォルト設定はNOEXTENDです。NOEXTENDが設定されていると、生成される順序値の幅は最大でも順序内の数字の最大数(MAXVALUE/MINVALUE)です。この設定は、固定幅の列を移入するために順序が使用される、既存のアプリケーションとの統合に役立ちます。SHARD NOEXTENDを指定した順序でNEXTVALを呼び出すと、生成された値に順序のMAXVALUE/MINVALUEよりも多くの桁数の表現が必要な場合は、ユーザー・エラーがスローされます。

SCALE句もSHARD句で指定されている場合、順序によって、グローバルに一意である複数のインスタンスおよびセッションに対してシャード内でスケーラブルな値が生成されます。EXTENDをSHARDキーワードとSCALEキーワードの両方で指定する場合、生成された順序値はすべての長さ(x+y+z)になります。xは先頭に付加されるサイズ4のSHARDオフセットの長さ、yはスケーラブルなオフセットの長さ(デフォルトは6)、zは順序MAXVALUE/MINVALUEの最大桁数です。

ノート:

SHARD 句を使用する場合は、順序に ORDER を指定しないでください。SHARD を使用すると、グローバルに順序付けされていない値が生成されます。ORDER が必要な場合は、各ノードでローカルに順序を作成します。

SHARD キーワードは、CACHE および NOCACHE モードの操作と組み合せて使用できます。

関連項目:

Oracle Database SQL言語リファレンス

6 シャード・データベースへの移行

既存の非シャード・データベースからシャード・データベースへの移行は、スキーマの移行とデータの移行の2つのフェーズで構成されます。Oracle Shardingには、既存のデータベース・スキーマおよびデータをシャード・データベースに移行するためのガイドラインがあります。

データベースの移行には、次の方法をお薦めします。

Oracle Data Pumpを使用したシャード・データベースへの移行

次のトピックで説明する例およびガイドラインを使用して、Oracle Data Pumpエクスポート・ユーティリティでソース・データベースからDDL定義およびデータを抽出し、データベース・エクスポート・ファイルに対してデータ・ポンプ・インポート・ユーティリティを使用してターゲット・シャード・データベースに移入できます。

シャード・データベースのスキーマをすでに作成している場合は、データ移行のトピックにすぐに移動できます。

シャード・データベースへのスキーマの移行

非シャード・データベースからシャード・データベースへの移行には、いくつかのスキーマ変更が必要です。少なくとも、キーワードSHARDEDまたはDUPLICATEDをCREATE TABLE文に追加する必要があります。場合によっては、表のパーティション化も変更するか、シェーディング・キーが追加された列を変更する必要があります。

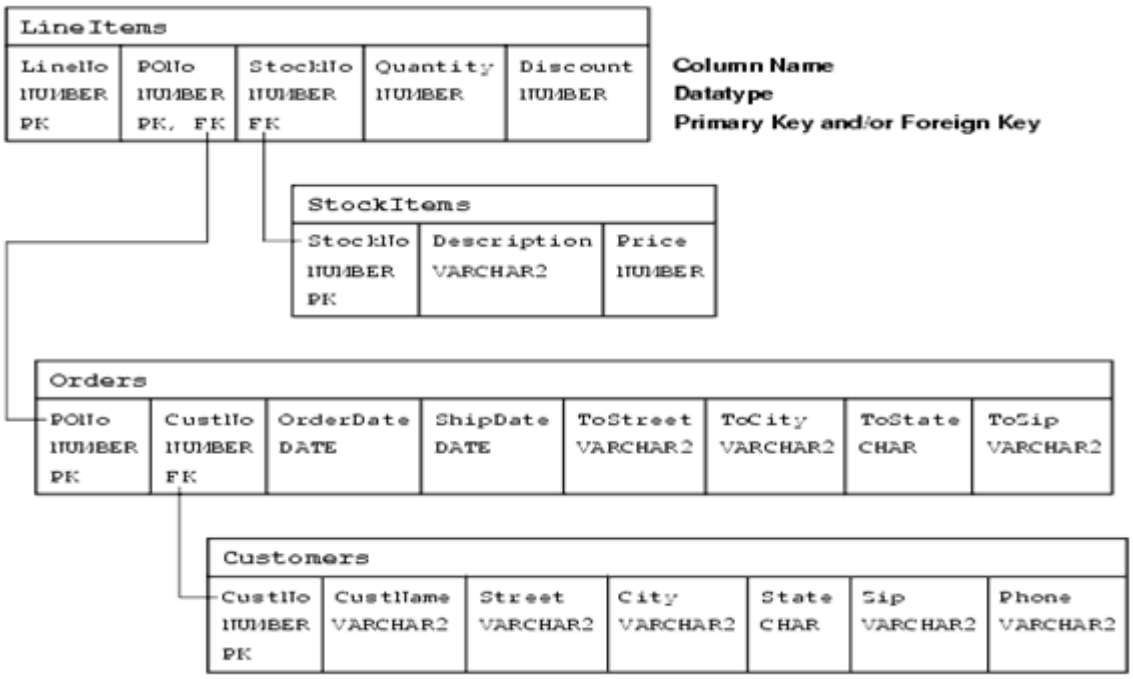
シャード・データベース・スキーマを適切に設計するには、非シャード・データベースのスキーマおよびワークロードを分析し、次の決定を行う必要があります。

- シャーディングする表と複製する表
- 表ファミリ内のシャード表間の親子関係
- シャード表で使用されるシャーディング方法
- シャーディング・キーとして何を使用するか

これらの決定が簡単でない場合は、シャーディング・アドバイザを使用して決定できます。シャーディング・アドバイザは、Oracle Sharding環境への移行を検討している非シャードOracle Databaseに対して実行するツールです。

非シャードからシャード・データベースへのスキーマおよびデータの移行を示すために、次の図に示すサンプル・データ・モデルを使用します。

図6-1 スキーマの移行例のデータ・モデル



データ・モデルはCustomers、Orders、StockItemsおよびLineItemsの4つの表で構成され、データ・モデルでは次の主キー制約が施行されます。

- Customer.(CustNo)
- Orders.(PONo)
- StockItems.(StockNo)
- LineItems.(LineNo, PONo)

このデータ・モデルでは、次の参照整合性制約が定義されます。

- Customers.CustNo -> Orders.CustNo
- Orders.PONo -> LineItems.PONo
- StockItems.StockNo -> LineItems.StockNo

次のDDL文は、非シャード・スキーマ定義の例を作成します。

```
CREATE TABLE Customers (
 CustNo     NUMBER(3) NOT NULL,
 CusName    VARCHAR2(30) NOT NULL,
 Street     VARCHAR2(20) NOT NULL,
 City       VARCHAR2(20) NOT NULL,
 State      CHAR(2) NOT NULL,
 Zip        VARCHAR2(10) NOT NULL,
 Phone      VARCHAR2(12),
 PRIMARY KEY (CustNo)
);
CREATE TABLE Orders (
 PoNo       NUMBER(5),
 CustNo     NUMBER(3) REFERENCES Customers,
 OrderDate  DATE,
 ShipDate   DATE,
 ToStreet   VARCHAR2(20),
 ToCity     VARCHAR2(20),
 ToState    CHAR(2),
 ToZip      VARCHAR2(10),
 PRIMARY KEY (PoNo)
```

```
);
CREATE TABLE LineItems (
 LineNo      NUMBER(2),
 PoNo        NUMBER(5) REFERENCES Orders,
 StockNo     NUMBER(4) REFERENCES StockItems,
 Quantity    NUMBER(2),
 Discount    NUMBER(4,2),
 PRIMARY KEY (LineNo, PoNo)
);
CREATE TABLE StockItems (
 StockNo     NUMBER(4) PRIMARY KEY,
 Description VARCHAR2(20),
 Price       NUMBER(6,2)
);
```

サンプル・スキーマの移行

例として、前述のサンプル・スキーマをシャード・データベースに移行するには、次のステップを実行します。

1. ソース・データベースのエクスポート・ディレクトリへのアクセスを取得します。

   データベース管理者は、次に示すように、データベース・ユーザーにデータベース・エクスポート・ディレクトリへのアクセスに必要な権限を付与する必要があります。

   ```
   CREATE OR REPLACE DIRECTORY expdir AS  '/some/directory' ;
   GRANT READ, WRITE ON DIRECTORY expdir TO uname;
   GRANT EXP_FULL_DATABASE TO uname;
   ```

   全データベース・エクスポートでは、データベース管理者がユーザーunameにEXP_FULL_DATABASEロールを付与する必要があります。表レベルのエクスポートには、追加のロールは必要ありません。

2. ソース・データベースからDDL定義を抽出します。

   DDL文を抽出する便利な方法は、データ・ポンプ抽出ファイルを作成することです。この例に示すように、エクスポートできるのはメタデータのみ、または移行する表のセットを含むスキーマの一部のみです。

   ```
   expdp uname/pwd directory=EXPDIR dumpfile=sample_mdt.dmp logfile=sample_mdt.log
   INCLUDE=TABLE:¥"IN ¥( ¥'CUSTOMERS¥', ¥'ORDERS¥', ¥'STOCKITEMS¥', ¥'LINEITEMS¥' ¥) ¥"
   CONTENT=METADATA_ONLY FLASHBACK_TIME=SYSTIMESTAMP
   ```

   その後、このデータベース・エクスポート・ファイルに対してデータ・ポンプ・インポート・ユーティリティを使用します。

   ```
   impdp uname/pwd@orignode directory=expdir dumpfile=sample_mdt.dmp sqlfile=sample_ddl.sql
   ```

   この例では、impdpコマンドは、実際にはダンプ・ファイルの内容をインポートしません。そのかわりに、sqlfileパラメータによって、sample_ddl.sqlという名前のスクリプトの作成がトリガーされます。このスクリプトには、エクスポート・ダンプ・ファイル内のすべてのDDLが格納されています。

   このようにエクスポートをトリミングして、時間がかかることがあるデータ・ポンプ・プロセスなしに、データベース・メタデータの一貫したイメージを効率的に取得します。それでも、テキスト形式でDDL文を取得して、シャード・データベース・スキーマ設計で要求されるようにDDLを変更する必要があります。

3. シャード・データベースの抽出されたDDL文を変更します。

   前述のサンプル・スキーマの場合、シャード・データベースに対応するDDL文は、次のようになります。これは、システム

管理シャーディングの例です。

```
CREATE SHARDED TABLE Customers (
 CustNo     NUMBER(3) NOT NULL,
 CusName    VARCHAR2(30) NOT NULL,
 Street     VARCHAR2(20) NOT NULL,
 City       VARCHAR2(20) NOT NULL,
 State      CHAR(2) NOT NULL,
 Zip        VARCHAR2(10) NOT NULL,
 Phone      VARCHAR2(12),
 CONSTRAINT RootPK PRIMARY KEY (CustNo)
)
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
CREATE SHARDED TABLE Orders (
 PoNo       NUMBER(5) NOT NULL,
 CustNo     NUMBER(3) NOT NULL,
 OrderDate  DATE,
 ShipDate   DATE,
 ToStreet   VARCHAR2(20),
 ToCity     VARCHAR2(20),
 ToState    CHAR(2),
 ToZip      VARCHAR2(10),
 CONSTRAINT OrderPK PRIMARY KEY (CustNo, PoNo),
 CONSTRAINT CustFK Foreign Key (CustNo) REFERENCES Customers (CustNo)
)
PARTITION BY REFERENCE (CustFK)
;
CREATE SHARDED TABLE LineItems (
 LineNo      NUMBER(2) NOT NULL,
 PoNo        NUMBER(5) NOT NULL,
 CustNo      NUMBER(3) NOT NULL,
 StockNo     NUMBER(4) NOT NULL,
 Quantity    NUMBER(2),
 Discount    NUMBER(4,2),
 CONSTRAINT LinePK PRIMARY KEY (CustNo, LineNo, PoNo),
 CONSTRAINT LineFK FOREIGN KEY (CustNo, PoNo) REFERENCES Orders (CustNo, PoNo)
)
PARTITION BY REFERENCE (LineFK)
;
CREATE DUPLICATED TABLE StockItems (
 StockNo     NUMBER(4) PRIMARY KEY,
 Description VARCHAR2(20),
 Price       NUMBER(6,2)
);
```

シャード・データベースのスキーマに関するいくつかの観測事項を次に示します。

- Customers - Orders: LineItemsは、Customersをルート表として、子表が参照によってパーティション化されたSHARDED表の表ファミリを形成します。StockItemsはDUPLICATED表です。
- シャーディング・キーとしてCustNoが選択されます。したがって、この列は表ファミリのすべての表に含める必要があります。非シャード・データベースでは、LineItems表にCustNo列はありませんでしたが、表のシャード・バージョンに含まれていました。シャーディング・キー列は、シャード表のすべての主キー制約および外部キー制約にも存在する必要があります。

- StockItemsが重複表になりました。重複表のマスター・コピーはシャード・カタログ・データベースに存在します。したがって、StockItems表を参照するLineItems表の外部キー制約は強制できず、削除されます。

4. 変更したDDLをターゲット・データベースに対して実行します。

   シャード・カタログ・データベースに接続して実行します

   ```
   ALTER SESSION ENABLE SHARD DDL;
   ```

   次に、前述のDDLを実行してシャード表および重複表を作成します。

   データのロード前に、GDSCTL VALIDATEコマンドを使用してシャーディング構成を検証することをお薦めします。

   ```
   gdsctl> validate
   ```

   不整合やエラーが見つかった場合は、GDSCTLコマンドのSHOW DDLおよびRECOVERを使用して問題を修正する必要があります。検証に成功すると、シャード・データベースはデータ・ロードの準備が整っていることになります。

シャード・データベースへのデータの移行

非シャード・データベースからシャード・データベースに移行するには、ソース・データベースの非シャード表からターゲット・データベースのシャード表および重複表にデータを移動する必要があります。

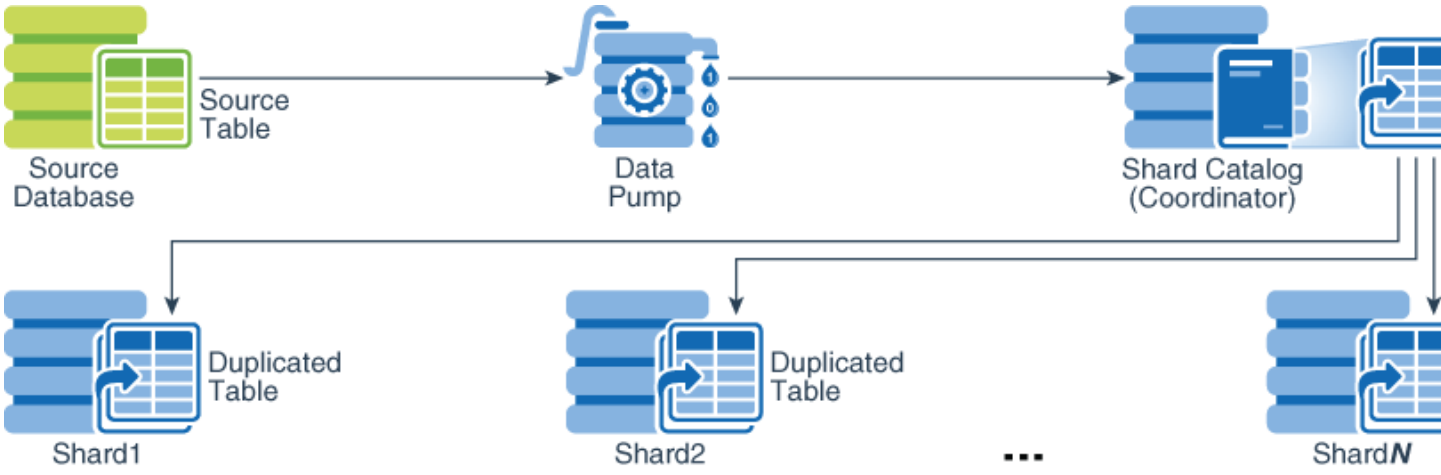
非シャード表から重複表へのデータの移動は簡単ですが、非シャード表からシャード表へのデータの移動には特に注意が必要です。

重複表へのデータのロード

Data Pump、SQL Loader、プレーンSQLなどの既存のデータベース・ツールを使用して、重複表にデータをロードできます。データはシャード・カタログ・データベースにロードする必要があります。その後、全シャードに自動的にレプリケートされます。

重複表の内容はマテリアライズド・ビューを使用してデータベース・シャードに完全にレプリケートされるため、重複表のロードにかかる時間は同じデータを通常の表にロードするよりも長くなる場合があります。

図6-2 重複表のロード



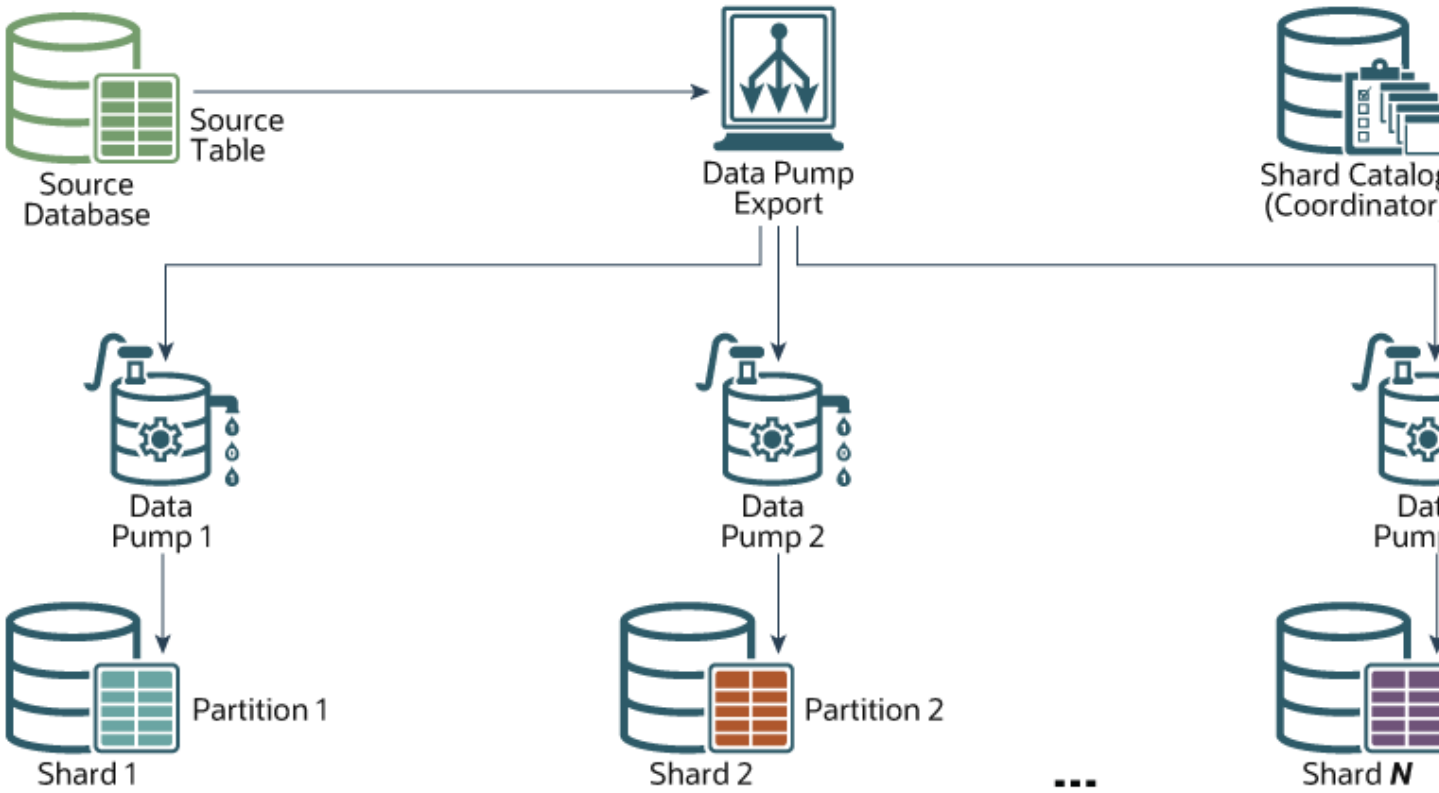
シャード表へのデータのロード

シャード表をロードする場合、各データベース・シャードにはデータ・セットの個別のサブセットが含まれるため、ロード中に各表のデータをシャード間で分割(パーティション化)する必要があります。

データベース・シャードのすべてにサブセットのデータをロードするには、Oracle Data Pumpユーティリティを使用できます。ソース・データベースのデータは、Data Pumpダンプ・ファイルにエクスポートできます。その後、同じダンプ・ファイルを使用して、各シャードで同時にData Pumpインポートを実行できます。

ダンプ・ファイルは、すべてのシャードからアクセス可能な共有記憶域に配置するか、各シャードのローカル記憶域にコピーできます。個々のシャードにインポートする場合、Data Pumpインポートは現在のシャードに属していない行を無視します。

図6-3 データベース・シャードへのシャード表の直接ロード



すべてのシャードがパラレルにロードされるため、データをシャードに直接ロードする方がはるかに高速です。線形スケーラビリティも提供されます。シャード・データベース内のシャードが多いほど、データ収集率が高くなります。

サンプル・スキーマ・データのロード

例として、次のステップでは、サンプル・スキーマ・データを非シャードからシャード・データベースに移動する方法を示します。構文の例は、前のトピックで紹介したサンプルのCustomers-Orders-LineItems-StockItemsスキーマに基づいています。

1. データベース表からデータをエクスポートします。

```
expdp uname/pwd@non_sharded_db directory=expdir dumpfile=original_tables.dmp
logfile=original_tables.log SCHEMAS=UNAME INCLUDE=TABLE:¥"IN ¥( ¥'CUSTOMERS¥', ¥'ORDERS¥',
¥'STOCKITEMS¥' ) ¥" FLASHBACK_TIME=SYSTIMESTAMP CONTENT=DATA_ONLY
```

ソース表(非シャード・データベース内)がパーティション化されている場合は、非パーティション化形式

(data_options=group_partition_table_data)でダンプ・ファイルにエクスポートします。

たとえば、Orders表がソース・データベースのパーティション表である場合は、次のようにエクスポートします。

```
$ cat ordexp.par
directory=expdir
logfile=ordexp.log
dumpfile=ord_%U.dmp
tables=ORDERS
parallel=8
COMPRESSION=ALL
content=data_only
DATA_OPTIONS=GROUP_PARTITION_TABLE_DATA
$ expdp user/password parfile=ordexp.par
```

SHARDEDおよびDUPLICATED表はターゲット・データベースにすでに作成されているため、表の内容(DATA_ONLY)のみをエクスポートします。

Data Pumpエクスポート・ユーティリティ・ファイルは、表ごとに一貫性があります。エクスポート内のすべての表の一貫性を同じ時点で保つ場合は、前述の例に示すように、FLASHBACK_SCNまたはFLASHBACK_TIMEパラメータを使用する必要があります。「特定」の時点での一貫性があるデータベース・エクスポート・ファイルを用意することをお薦めします。

2. ターゲットのデータベース・ノードでエクスポート・ファイル(original_tables.dmp)にアクセスできるようにしてから、シャード・データベースへのデータのインポートを開始します。

   このファイル(並列エクスポートの場合は複数のファイル)は、ターゲット・データベース・システムに移動することも、ネットワーク経由で共有することもできます。

3. すべてのターゲット・データベース(シャード・カタログおよびシャード)をインポート用に準備します。

   データベース管理者は、次に示すように、データベース・ユーザーにデータベース・インポート・ディレクトリへのアクセスに必要な権限を付与する必要があります。

```
CREATE OR REPLACE DIRECTORY expdir AS  '/some/directory' ;
GRANT READ, WRITE ON DIRECTORY expdir TO uname;
GRANT IMP_FULL_DATABASE TO uname;
```

4. シャード・カタログを使用してDUPLICATED表(StockItems)をロードします。

   次に、インポート・コマンドの例を示します。

```
impdp uname/pwd@catnode:1521/ctlg directory=expdir dumpfile=original_tables.dmp
logfile=imp_dup.log tables=StockItems content=DATA_ONLY
```

5. シャードにSHARDED表を直接ロードします。

   エクスポートされたSHARDED表(Customers、Orders)をロードする最適な方法は、各シャード(shrd1,2、…、N)でデータ・ポンプを直接実行することです。次に、最初のシャードでのインポート・コマンドの例を示します。

```
impdp uname/pwd@shrdnode:1521/shrd1 directory=expdir DUMPFILE=original_tables.dmp
LOGFILE=imp_shd1.log TABLES=" Customers, Orders, LineItems"  CONTENT=DATA_ONLY
```

   他のすべてのシャードで、このステップを繰り返します。すべてのシャードのデータのロードに同じダンプ・ファイル(original_tables.dmp)が使用されることに注意してください。Data Pumpインポートでは、現在のシャードに属さない行は無視されます。この操作は、すべてのシャードでパラレルに実行できます。

   並列性を持つ非常に大きなパーティション表への高速ロードの利点を得るには、データ・ポンプ・パラメータ

DATA_OPTIONSに値_FORCE_PARALLEL_DMLを含める必要があります(パッチ31891464が必要です)。

```
$ cat ordimp.par
directory=expdir
logfile=ordimp.log
dumpfile=ord_%U.dmp
tables=ORDERS
parallel=8
content=data_only
DATA_OPTIONS=_force_parallel_dml
$ impdp user/password parfile=ordimp.par
```

パッチ31891464がない場合は、次の例に示すように、DATA PUMPタイプの外部表を使用してデータを移行することもできます。

a. ソース・データベースでエクスポートします。

```
CREATE TABLE ORDERS_EXT
 ORGANIZATION EXTERNAL
    ( TYPE ORACLE_DATAPUMP
      DEFAULT DIRECTORY "expdir"
      ACCESS PARAMETERS ( DEBUG = (3 , 33489664))
      LOCATION ('ord1.dat',
                'ord2.dat',
                'ord3.dat',
                'ord4.dat')
    )
PARALLEL 8
REJECT LIMIT UNLIMITED
AS SELECT * FROM user.ORDERS;
```

b. 各ターゲット・シャードにインポートします。

```
CREATE TABLE ORDERS_EXT
 ORGANIZATION EXTERNAL
    ( TYPE ORACLE_DATAPUMP
      DEFAULT DIRECTORY "expdir"
      ACCESS PARAMETERS ( DEBUG = (3 , 33489664))
      LOCATION ('ord1.dat',
                'ord2.dat',
                'ord3.dat',
                'ord4.dat')
    )
PARALLEL 8
REJECT LIMIT UNLIMITED
;
INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL(a,12) pq_distribute(a, random) */ INTO
"user"."ORDERS" a
SELECT /*+ full(b) parallel(b,12) pq_distribute(b, random)*/
*
FROM "ORDERS_EXT"
WHERE <predicate*>;
Commit;
```

(*) WHERE句の述語はシャーディング方法によって異なります。たとえば、範囲別のユーザー定義シャーディングの場合、特定のシャードのCustNoの範囲に基づきます。システム管理(コンシステント・ハッシュ・ベース)シャーディングの場合は、外部表を使用したシャード・データベースへのデータのロードのユースケースを参照し

てください。

ノート:

expdp および impdp コマンドで PARALLEL パラメータを使用すると、Data Pump の実行速度を上げることができます。エクス
の%U ワイルド・カードと組み合せて使用して、複数のダンプ・ファイルを作成できるようにする必要があります。

```
expdp uname/pwd@orignode SCHEMAS=uname directory=expdir dumpfile=samp_%U.dmp logfile=samp.log FLASHBA
```

前述のコマンドでは、4 つのパラレル・ワーカーが使用され、接尾辞_01、_02、_03 および_04 を持つ 4 つのダンプ・ファイル
入力ファイルを参照できるようになります。

シャーディング・キーのないデータの移行

例として、次のステップは、シャーディング・キーを含まないソース表からシャード表にデータを移行する方法を示しています。

前のトピックのデータ・ポンプ・エクスポートおよびインポート・コマンドの例には、LineItems表は含まれていません。これは、非シャード・データベースのこの表にシャーディング・キー列(CustNo)が含まれていないためです。ただし、この列は表のシャード・バージョンで必要です。

非シャード・バージョンとシャード・バージョンの表のスキーマが一致しないため、次のステップに示すように、LineItemsのデータ移行は異なる方法で処理する必要があります。

1. ソースの非シャードのデータベースで、この列の値を生成するための列およびSQL式が欠落している一時ビューを作成します。

   ```
   CREATE OR REPLACE VIEW Lineitems_View AS
     SELECT l.*,
           (SELECT o.CustNo From Orders o WHERE l.PoNo=o.PoNo) CustNo
   FROM LineItems l;
   ```

   これにより、Orders表との外部キー関係に基づいて列CustNoが移入されたビューLineItems_Viewが作成されます。

2. データ・ポンプ・エクスポート・ユーティリティのVIEWS_AS_TABLESオプションを使用して、新しいビューをエクスポートします。

   ```
   expdp uname/pwd@non_sharded_db directory=expdir DUMPFILE=original_tables_vat.dmp
   LOGFILE=original_tables_vat.log FLASHBACK_TIME=SYSTIMESTAMP CONTENT=DATA_ONLY
   TABLES=Uname.Customers,Uname.Orders,Uname.StockItems  VIEWS_AS_TABLES=Uname.LineItems_
   ```

3. 個々のシャード(shrd1、shrd2、..、shrdN)でデータ・ポンプ・インポートを直接実行して、データをシャード表にインポートします。

   次に、最初のシャードでインポートを実行する例を示します。

   ```
   impdp uname/pwd@shrdnode:1521/shrd1 directory=expdir DUMPFILE=original_tables_vat.dmp
   LOGFILE=imp_shd_vat1.log CONTENT=DATA_ONLY
   TABLES=Uname.Customers,Uname.Orders,Uname.LineItems_View VIEWS_AS_TABLES=Uname.LineItems_View
   REMAP_TABLE=Lineitems_View:Lineitems
   ```

   この例では、impdpツールのVIEWS_AS_TABLESオプションを使用して、エクスポート操作中に表としてエクスポートされた

ビューLineItems_Viewをインポートします。また、パラメータREMAP_TABLEを使用して、このデータを実際に元の表LineItemsに挿入する必要があることを示します。

外部表を使用したシャード・データベースへのデータのロード

次のトピックの例およびガイドラインを使用して、外部表を作成し、外部表からシャード表または重複表にデータをロードすることで、シャード・データベースにデータをロードできます。

このデータ・ロード方法は、ロードするデータがCSVファイルなどの外部ファイルに存在する場合に便利です。

外部表は、CREATE TABLE文でORGANIZATION EXTERNALキーワードを使用して定義できます。この表は、シャードまたは重複ではなく、各シャードに対してローカルである必要があります。シャード表または重複表へのデータのロードには、外部表からの単純なINSERT … SELECT文と、シャード表のデータのサブセットのみをフィルタ処理する条件が含まれます。

ファイルのアクセス時間とサイズに基づいて、ファイルを別のホストに保存することもできます。たとえば、シャード・カタログ・ホスト上の重複表のファイルをコピーし、すべてのシャードからアクセス可能なネットワーク共有上のシャード表のファイルを保持します。ロードを高速化するために、シャード表ファイルのコピーを各シャードに保持することもできます。

外部表の詳細は、*Oracle Database*ユーティリティの外部表に関する項を参照してください。

重複表へのデータのロード

重複表のデータはシャード・カタログに存在するため、重複表へのデータのロードもシャード・カタログで実行されます。ロードが完了すると、データは自動的にシャードにレプリケートされます。

次の表が重複表として定義されているとします。

```
CREATE DUPLICATED TABLE StockItems (
 StockNo     NUMBER(4) PRIMARY KEY,
 Description VARCHAR2(20),
 Price       NUMBER(6,2)
);
```

StockItems表へのデータのロードには、次のステップが必要です。

1. データ・ファイルを含むディレクトリを指すディレクトリ・オブジェクトを作成し、このディレクトリのシャード・ユーザーにアクセス権を付与します。

   ```
   CREATE OR REPLACE DIRECTORY shard_dir AS '/path/to/datafile';
   GRANT ALL on DIRECTORY shard_dir TO uname;
   ```

2. シャード・カタログに対してローカルで、重複表と同じ列を持つ外部表を作成します。

   シャード・カタログで、次のコマンドを実行します。

   ```
   ALTER SESSION DISABLE SHARD DDL;
   CREATE TABLE StockItems_Ext (
    StockNo     NUMBER(4) NOT NULL,
    Description VARCHAR2(20),
    Price       NUMBER(6,2)
   )
   ORGANIZATION EXTERNAL
   (TYPE ORACLE_LOADER DEFAULT DIRECTORY shard_dir
           ACCESS PARAMETERS
   ```

```
                (FIELDS TERMINATED BY ' |'  (
                  StockNo,
                  Description,
                  Price)
       )LOCATION (' StockItems.dat' )
);
```

この例では、重複表のデータ・ファイルの名前はStockItems.datで、列値は'|'文字で区切られます。

3. 外部表から重複表にデータを挿入します。

```
INSERT INTO StockItems  (SELECT * FROM StockItems_Ext);
```

APPENDやPARALLEL (並列度)などのオプティマイザ・ヒントを使用して、システム・リソースに応じてロードを高速化することもできます。たとえば:

```
ALTER SESSION ENABLE PARALLEL DML;
INSERT /*+ APPEND PARALLEL */ INTO StockItems
  (SELECT * FROM StockItems_Ext);
```

または

```
ALTER SESSION ENABLE PARALLEL DML;
INSERT /*+ APPEND PARALLEL(24) */ INTO StockItems
  (SELECT * FROM StockItems_Ext);
```

4. 挿入操作をコミットします。

```
COMMIT;
```

5. 外部表を削除します。

```
DROP TABLE StockItems_Ext;
```

重複表ごとにこれらのステップを繰り返します。

シャード表へのデータのロード

シャード表のデータはシャード間でパーティション化されるため、シャード表へのデータのロードは個々のシャードで実行する必要があります。ロードは、ソース・データ・ファイルが共有されている場合でも、すべてのシャードで同時に実行できます。

ロードのプロセスは重複表のロードと似ていますが、現在のシャードに属していない行をフィルタで除外するための追加のフィルタがINSERT … SELECT文にあります。

たとえば、次のように作成されたシャード表があるとします。

```
CREATE SHARDED TABLE Customers (
 CustNo     NUMBER(3) NOT NULL,
 CusName    VARCHAR2(30) NOT NULL,
 Street     VARCHAR2(20) NOT NULL,
 City       VARCHAR2(20) NOT NULL,
 State      CHAR(2) NOT NULL,
 Zip        VARCHAR2(10) NOT NULL,
 Phone      VARCHAR2(12),
 CONSTRAINT RootPK PRIMARY KEY (CustNo)
)
```

```
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

この表にデータをロードするには、各シャードで次のステップを実行します。

1. 重複表と同じ方法でディレクトリ・オブジェクトを作成します。
2. Customers表の外部表を作成します。

```
ALTER SESSION DISABLE SHARD DDL;
CREATE TABLE Customers_Ext (
 CustNo     NUMBER(3) NOT NULL,
 CusName    VARCHAR2(30) NOT NULL,
 Street     VARCHAR2(20) NOT NULL,
 City       VARCHAR2(20) NOT NULL,
 State      CHAR(2) NOT NULL,
 Zip        VARCHAR2(10) NOT NULL,
 Phone      VARCHAR2(12)
)
ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER DEFAULT DIRECTORY shard_dir
        ACCESS PARAMETERS
        (FIELDS TERMINATED BY ' |'  (
          CustNo, CusName, Street, City, State, Zip, Phone)
        )LOCATION (' Customers.dat' )
 );
```

3. 外部表からシャード表にデータを挿入します。

```
ALTER SESSION ENABLE PARALLEL DML;
INSERT /*+ APPEND PARALLEL(24) */ INTO Customers
 (SELECT * FROM Customers_Ext WHERE
        SHARD_CHUNK_ID(' UNAME.CUSTOMERS' , CUSTNO) IS NOT NULL
  );
```

演算子SHARD_CHUNK_IDを使用して、現在のシャードに属する行をフィルタします。この演算子は、指定されたシャーディング・キー値に対して有効なチャンク番号を戻します。この演算子のパラメータは、ルート表名(この場合はUNAME.CUSTOMERS)およびシャーディング・キー列の値です。値が現在のシャードに属していない場合、この演算子はNULLを戻します。

この演算子は、現在のリリース(Oracle Database 21c)で導入されています。ご使用のバージョンでこの演算子を使用できない場合は、システム管理シャーディングの場合に次のようにinsert文を変更する必要があります。

```
INSERT /*+ APPEND PARALLEL(24) */ INTO Customers c
 (SELECT * FROM Customers_Ext WHERE
        EXISTS (SELECT chunk_number FROM gsmadmin_internal.chunks
                        WHERE ora_hash(c.CustNo)>= low_key
                          AND ora_hash c.CustNo)< high_key)
  );
```

この問合せでは、ユーザーの内部シャーディング・メタデータを問い合せて、挿入する行の適格性を決定します。

4. 挿入操作をコミットします。

```
COMMIT;
```

5. 外部表を削除します。

```
DROP TABLE Customers_Ext;
```

外部キー制約を維持するには、ルート表から開始して表ファミリ階層を降順に、シャード表ごとに前述のステップを繰り返します。

Oracle GoldenGateを使用したシャード・データベースと非シャード・データベース間のデータのレプリケート

Oracle GoldenGateを使用して、非シャード・データベースからシャード・データベースにデータを移行できます。

Oracle GoldenGateを使用した非シャード・データベースからシャード・データベースへのデータの移行は、2つのフェーズで実行されます。

ソース・データベースでの抽出

- ソース・データベースのすべての表が、ソース・データベースで単一の抽出を使用して抽出されます。

ターゲット・データベースのレプリケーション

- シャード表へのレプリケーションはシャードで実行され、重複表へのレプリケーションはシャード・カタログで実行されます。

Oracle GoldenGateレプリケーションの前提条件

シャード・データベースのOracle GoldenGateレプリケーションを試行する前に、ソース・データベースとターゲット・データベースおよびOracle GoldenGate環境がこれらの前提条件を満たしていることを確認してください。

前提

1. 非シャード・データベースからシャード・データベースに移行される表は、すでにシャード表および重複表に分類されているものとします。
2. シャード表に移行するすべての表のシャーディング・キーはすでに識別されています。
3. シャード表と重複表は、ターゲット・シャード・データベースに事前作成されています。
4. Oracle GoldenGateソフトウェアは、ソース・システムとターゲット・システムにすでにインストールされています。

ソース・データベースおよびターゲット・データベース

- Oracle Databaseバージョン: マルチテナント・アーキテクチャの19c (19.15.0.0.0)以降
- ターゲット・データベースのシャーディング・タイプ: システム管理

Oracle GoldenGate構成

Oracle GoldenGateバージョン: ハブ構成の19c Classicアーキテクチャ

非シャード・データベースからシャード・データベースへのデータのレプリケート

環境の例

以降のステップの例では、次のトポロジを使用します

システム/オブジェクト	ソース環境	ターゲット環境

システム/オブジェクト	ソース環境	ターゲット環境
CDB 名	srccdb	sdbcdb
PDB 名	srcpdb	シャード: sdbpdb1、sdbpdb2、sdbpdb3 シャード・カタログ: scpdb
アプリケーション・スキーマ	app_schema	app_schema
シャード表	顧客、注文、品目	顧客、注文、品目
重複表	製品	製品

概要ステップ

1) ソース・データベースで抽出を作成し、ソース表からトランザクションを取得して起動します。

2) expdpおよびflashback_scnを使用して、初期ロード用にソース・データベースからデータを取得します。

3) impdpを使用して、ターゲット・シャード上のシャード表への初期ロードを実行します。

4) impdpを使用して、ターゲット・シャード・カタログで重複表への初期ロードを実行します。

5) シャード表をレプリケートするターゲット・シャードの数と同じ数のレプリカを作成します。

6) 重複表をレプリケートするために、シャード・カタログに1つのレプリカを作成します。

7) ターゲット・シャードでat csnを使用してレプリカを開始します

8) シャード・カタログでat csnを使用してレプリカを開始します

9) ソース表からターゲット表へのデータ・レプリケーションを検証します。

1. ソース(非シャード)・データベースの構成

    a. ソース・データベースで抽出を作成し、ソース表からトランザクションを取得して起動します。

```
$ ggsci

GGSCI > dblogin useridalias ggadmin
GGSCI > add extract extnshd, integrated tranlog, begin now
GGSCI > register extract extnshd, database container (SRCPDB)
GGSCI > add exttrail ./dirdat/tr, extract extnshd

Add the following parameters in extract parameter file

GGSCI > edit params extnshd

extract extnshd
useridalias  ggadmin
```

```
TranlogOptions IntegratedParams (max_sga_size 256)
extTrail ./dirdat/tr
DiscardFile ./dirrpt/extnshd.dsc, Append Megabytes 50
REPORTCOUNT EVERY 2 HOURS, RATE
Table SRCPDB.app_schema.customers;
Table SRCPDB.app_schema.orders;
Table SRCPDB.app_schema.lineitems;
Table SRCPDB.app_schema.products;

GGSCI> start extract extnshd
```

b. expdpを使用して、初期ロード用にソース・データベースからデータを取得します。

```
SQL> select current_scn from v$database;

$ expdp app_schema/xxxxx@SRCPDB flashback_scn=current_scn_from_previous_step
 directory=DATA_PUMP_DIR dumpfile=app_schema_exp.dmp
 logfile=app_schema_exp.log
```

2. ターゲット(シャード)・データベースを構成します。

a. impdpを使用して、ターゲット・シャード・データベースおよびシャード・カタログで初期ロードを実行します。

```
Import into shards
$ impdp app_schema/xxxxx@SDBPDB1 directory=DATA_PUMP_DIR dumpfile=app_schema_exp.dmp
 logfile=app_schema_imp.log tables=CUSTOMERS,ORDERS,LINEITEMS, CONTENT=DATA_ONLY
$ impdp app_schema/xxxxx@SDBPDB2 directory=DATA_PUMP_DIR dumpfile=app_schema_exp.dmp
 logfile=app_schema_imp.log tables=CUSTOMERS,ORDERS,LINEITEMS, CONTENT=DATA_ONLY
$ impdp app_schema/xxxxx@SDBPDB3 directory=DATA_PUMP_DIR dumpfile=app_schema_exp.dmp
 logfile=app_schema_imp.log tables=CUSTOMERS,ORDERS,LINEITEMS, CONTENT=DATA_ONLY

Import into shard catalog
$ impdp app_schema/xxxxx@SCPDB directory=DATA_PUMP_DIR dumpfile=app_schema_exp.dmp
 logfile=app_schema_imp.log tables=PRODUCTS CONTENT=DATA_ONLY
```

b. ターゲット・データベースに(シャード数と同じ)レプリカを作成します。

```
Replicat for sharded tables on Shard 1
======================================
GGSCI > dblogin useridalias ggadmin_shd1
GGSCI > add replicat repshd1, INTEGRATED, exttrail ./dirdat/tr
 CHECKPOINTTABLE ggadmin.GGCHKPT

Add the following parameters in replicat for shard1

GGSCI > edit params repshd1

replicat repshd1
useridalias ggadmin_shd1
HANDLECOLLISIONS
SOURCECATALOG SDBPDB1
MAP NSHDPDB.APP_SCHEMA.CUSTOMERS , target APP_SCHEMA.CUSTOMERS, &
SQLEXEC (ID chunklookup1, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup1.COUNT = 1);

MAP NSHDPDB.APP_SCHEMA.ORDERS, target APP_SCHEMA.ORDERS, &
```

```
SQLEXEC (ID chunklookup2, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup2.COUNT = 1);

MAP NSHDPDB.APP_SCHEMA.LINEITEMS, target APP_SCHEMA.LINEITEMS, &
SQLEXEC (ID chunklookup3, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup3.COUNT = 1);

Replicat for sharded tables on Shard 2
======================================

GGSCI > dblogin useridalias ggadmin_shd2
GGSCI > add replicat repshd2, INTEGRATED, exttrail ./dirdat/tr
 CHECKPOINTTABLE ggadmin.GGCHKPT

Add the following parameters in replicat for shard2

GGSCI > edit params repshd2

replicat repshd2
useridalias ggadmin_shd2
HANDLECOLLISIONS
SOURCECATALOG SDBPDB2
MAP NSHDPDB.APP_SCHEMA.CUSTOMERS , target APP_SCHEMA.CUSTOMERS, &
SQLEXEC (ID chunklookup1, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup1.COUNT = 1);

MAP NSHDPDB.APP_SCHEMA.ORDERS, target APP_SCHEMA.ORDERS, &
SQLEXEC (ID chunklookup2, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup2.COUNT = 1);


MAP NSHDPDB.APP_SCHEMA.LINEITEMS, target APP_SCHEMA.LINEITEMS, &
SQLEXEC (ID chunklookup3, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup3.COUNT = 1);

Replicat for sharded tables on Shard 3
======================================

GGSCI > dblogin useridalias ggadmin_shd3
GGSCI > add replicat repshd3, INTEGRATED, exttrail ./dirdat/tr
 CHECKPOINTTABLE ggadmin.GGCHKPT

Add the following parameters in replicat for shard3
```

```
GGSCI > edit params repshd3

replicat repshd3
useridalias ggadmin_shd3
HANDLECOLLISIONS
SOURCECATALOG SDBPDB3
MAP NSHDPDB.APP_SCHEMA.CUSTOMERS , target APP_SCHEMA.CUSTOMERS, &
SQLEXEC (ID chunklookup1, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup1.COUNT = 1);

MAP NSHDPDB.APP_SCHEMA.ORDERS, target APP_SCHEMA.ORDERS, &
SQLEXEC (ID chunklookup2, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup2.COUNT = 1);


MAP NSHDPDB.APP_SCHEMA.LINEITEMS, target APP_SCHEMA.LINEITEMS, &
SQLEXEC (ID chunklookup3, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(:CODE_IN_PARAM) >= low_key and ora_hash(:CODE_IN_PARAM) < high_key',&
PARAMS (CODE_IN_PARAM = CUSTID),
BEFOREFILTER), &
FILTER (chunklookup3.COUNT = 1);

#### NOTE ####
1. Remove Handlecollisions parameter and restart replicats after deltas
   have been applied on target shards.
2. If sharding key column is of  number datatype, please use below sqlexec
   filter which has to_number in ora_hash function.

SQLEXEC (ID chunklookup, QUERY 'select count(*) count FROM gsmadmin_internal.chunks
 WHERE ora_hash(to_number(:CODE_IN_PARAM)) >= low_key
   and ora_hash(to_number(:CODE_IN_PARAM)) < high_key',&


Replicat for duplicate tables on Catalog
========================================

GGSCI > dblogin useridalias ggadmin_cat
GGSCI > add replicat repcat, INTEGRATED, exttrail ./dirdat/tr
 CHECKPOINTTABLE ggadmin.GGCHKPT

Add the following parameters in replicat for shard1
GGSCI > edit params repcat

replicat repcat
useridalias ggadmin_cat
HANDLECOLLISIONS
SOURCECATALOG SCPDB
map NSHDPDB.APP_SCHEMA.PRODUCTS, target APP_SCHEMA.PRODUCTS;
```

c. ターゲット・シャードでatcsnを使用してレプリカを開始します。

```
GGSCI> start replicat repshd1, atcsn <SCN captured on source>
GGSCI> start replicat repshd2, atcsn <SCN captured on source>
GGSCI> start replicat repshd3, atcsn <SCN captured on source>
GGSCI> start replicat repcat, atcsn  <SCN captured on source>
GGSCI > info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     EXTNSHD     00:00:00      00:00:05
REPLICAT    RUNNING     REPCAT      00:00:00      00:00:00
REPLICAT    RUNNING     REPSHD1     00:00:00      00:00:03
REPLICAT    RUNNING     REPSHD2     00:00:00      00:00:06
REPLICAT    RUNNING     REPSHD3     00:09:01      00:00:09
```

3. ソース表からターゲット表へのデータ・レプリケーションを検証します。

   非シャード表からシャードに行がレプリケートされることを検証するには、たとえば、ソース表に9000行あり、3つのターゲット・シャードがある場合、各シャードに約3000行を分散する必要があります。

7 問合せおよびDMLの実行

シャード・データベースでは、問合せおよびDMLをシャードにルーティングして、シャーディング・キーの有無に関係なく実行できます。アプリケーションによってキーが提供される場合、データベース・リクエストはシャードに直接ルーティングされますが、キーが指定されない場合、リクエストはシャード・カタログによって処理され、実行に必要なシャードに送信されます。

データベース・リクエストのシャードへのルーティング方法

Oracle Shardingでは、シャーディング・キーがリクエストに指定されているかどうかに応じて、データベース問合せおよびDMLリクエストは2種類の主な方法でシャードにルーティングされます。

この2つのルーティング方法は、直接ルーティングおよびプロキシ・ルーティングと呼ばれます。

直接ルーティング

データベース・リクエストでシャーディング・キーを指定することで、シャードに直接接続して問合せおよびDMLを実行できます。直接ルーティングは、パフォーマンスを向上させるためにシャードにアクセスする場合に推奨される方法で、特に利点があります。

プロキシ・ルーティング

アプリケーションでは、複数のシャードからのデータを必要とする問合せや、シャーディング・キーを指定していない問合せを直接ルーティングできません。こうした問合せには、アプリケーションとシャードの間でリクエストをルーティングするためのプロキシが必要になります。プロキシ・ルーティングは、シャード・カタログ問合せコーディネータによって処理されます。

問合せおよびDMLのシャードへの直接ルーティング

シャーディング・キーを提供する場合、アプリケーションはリクエストをシャードに直接ルーティングできます。直接ルーティング・メカニズムでは、リクエストはルーティングされたシャードに属するデータの問合せおよび操作のみが可能です。

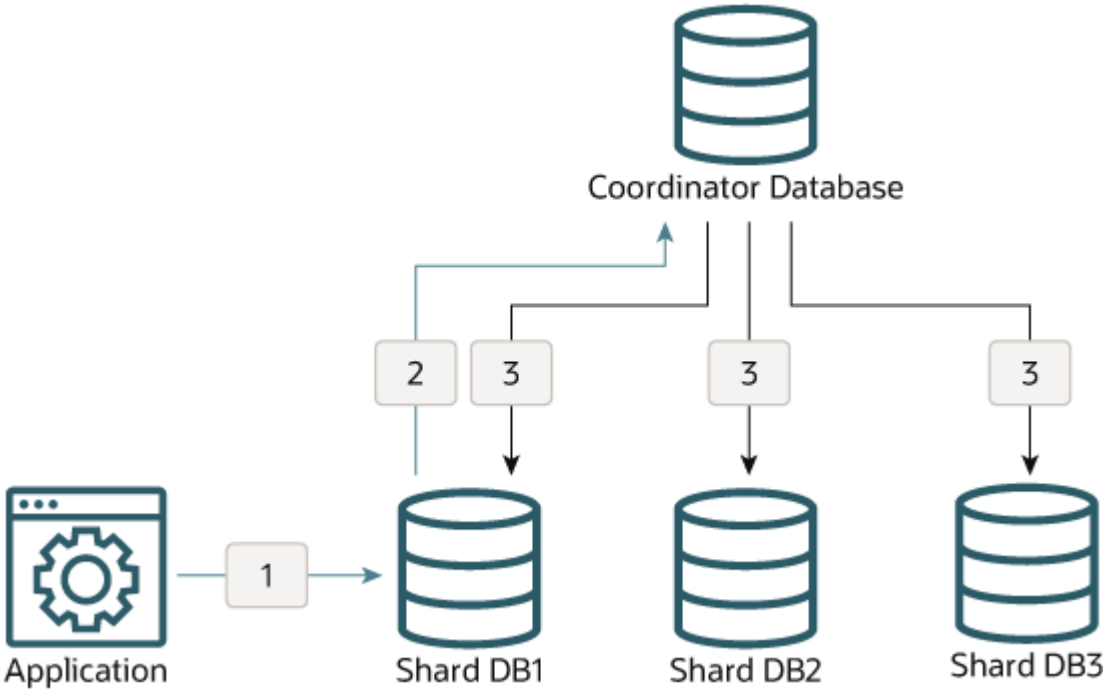
シャード上のデータへの直接アクセスには、いくつかの利点があります。

- パフォーマンスの向上: 全体的に、アプリケーションでは、シャード・カタログを介してリクエストをシャードに間接的にルーティングするよりもパフォーマンスが向上します(プロキシによる)。直接ルーティングでは、リクエストおよび結果がコーディネータ・データベースを通過する必要はありません。
- シャードの地理的分散に対応します。アプリケーションは、リージョン内でローカライズされたシャード内のデータにアクセスできます。
- ロード・バランシングが容易: チャンク移動を使用してシャード間でデータを移動することで、シャード間でのアプリケーション・リクエストのロード・バランシングを簡単に実行できます。
- すべてのタイプの問合せをサポートします。
    - シャード表に対するSELECT、INSERTおよびUPDATE: これらの問合せの有効範囲は、アクセスされるシャードに属するデータです。
    - 重複表に対するSELECT、INSERTおよびUPDATE: これらの問合せの有効範囲は、重複表内のすべてのデータです。重複表のマスター・コピーはコーディネータ・データベースに存在するため、重複表のDMLはコーディネータ・データベースに再ルーティングされます。

次の図は、シャードへの直接ルーティングを使用した重複表に対するDMLを示しています。

1. アプリケーションは、シャードの1つ(シャードDB1)にDMLリクエストを直接送信します。
2. DMLは、シャードDB1からコーディネータ・データベースに転送され、マスター重複表で実行されます。
3. コーディネータ・データベースのリフレッシュ・メカニズムが定期的に実行され、すべてのシャードで重複表のインスタンスが更新されます。

図7-1 直接ルーティングを使用した重複表のDML



直接ルーティングの詳細は、クライアント・アプリケーション・リクエストのルーティングを参照してください。

直接ルーティング用アプリケーションの開発の詳細は、シャード・データベースのアプリケーションの開発を参照してください。

プロキシによる問合せおよびDMLのルーティング

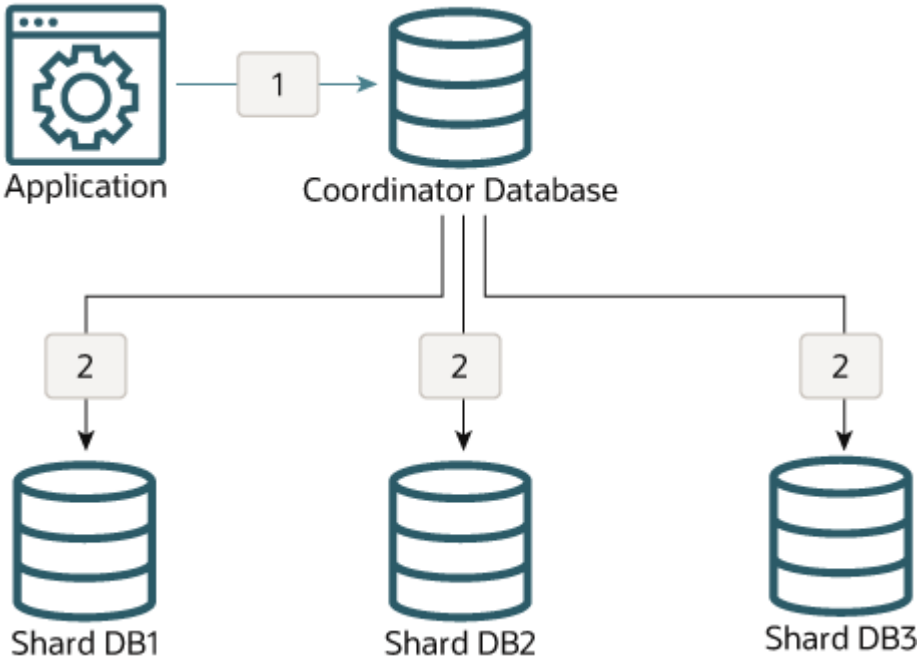
シャード・カタログ問合せコーディネータをプロキシとして使用すると、Oracle Shardingでは、シャーディング・キーを指定しない問合せおよびDMLのリクエスト・ルーティングを処理できます。

コーディネータをプロキシとして使用することで、Oracle Shardingでは、問合せを実行するシャードを指定しなくても、任意のデータベース・アプリケーションがSQL文を実行できる柔軟性が提供されます。

次の図は、プロキシ・ルーティングを使用した重複表に対するDMLを示しています。

1. DMLリクエストがアプリケーションからコーディネータ・データベースに送信され、マスター重複表で実行されます。
2. コーディネータ・データベースのリフレッシュ・メカニズムが定期的に実行され、すべてのシャードで重複表のインスタンスが更新されます。

図7-2 プロキシ・ルーティングを使用した重複表のDML



コーディネータの詳細は、問合せ処理と問合せコーディネータを参照してください。

この章の残りのトピックでは、プロキシによるデータベース・リクエストのルーティングおよび処理について説明します。

問合せコーディネータへの接続

Oracle Sharding問合せコーディネータ(シャード・カタログのコンポーネント)には、シャード・トポロジのメタデータが含まれており、シャード・データベースの問合せ処理をサポートします。

マルチシャード問合せを実行するには、シャード・カタログ・データベースでGDS$CATALOGサービスを使用してコーディネータに接続します。

```
sqlplus app_schema/app_schema@shardcatvm:1521/GDS¥$CATALOG.oradbcloud
```

コーディネータの詳細は、問合せ処理と問合せコーディネータを参照してください

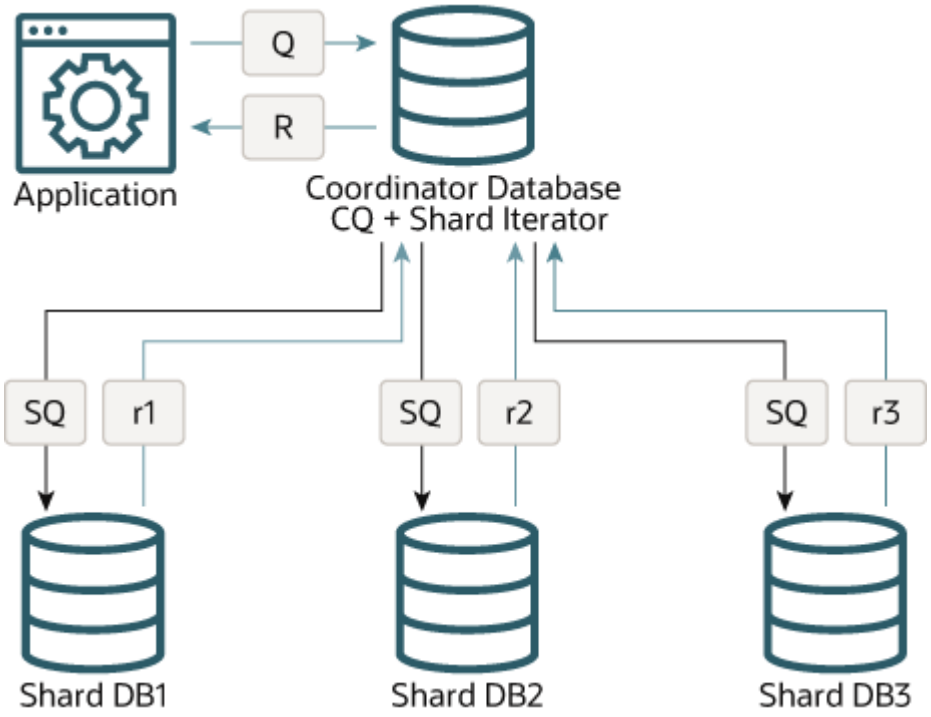
問合せコーディネータ操作

シャード・カタログのSQLコンパイラは、関連するシャードを自動的に識別し、関係するすべてのシャード間で問合せの実行を調整します。コーディネータとシャードの間の通信のために、データベース・リンクが使用されます。

次の図に示すように、おおまかに言うと、コーディネータは各受信問合せQをコーディネータ問合せ(CQ)とシャード問合せ(SQ)の2つの問合せにリライトします。SQ (シャード問合せ)は各参加シャードで実行されるキューの一部、CQ (コーディネータ問合せ)はコーディネータ・シャードで実行されるキューの一部です。

問合せQはCQ (Shard_Iterator (SQ))にリライトされます。ここで、**Shard_Iterator**はシャードに接続してSQを実行する演算子です。パラレルまたはシリアルで実行できます。

図7-3 問合せコーディネータ操作



次の例では、集計問合せQ1をQ1'にリライトしています。

```
Q1 : SELECT COUNT(*) FROM customers
Q1' : SELECT SUM(sc) FROM (Shard_Iterator(SELECT COUNT(*) sc FROM s1 (i) ))
```

この処理には2つの主要な要素があります。

1. 関連するシャードが識別されます。
2. 問合せが分散形式にリライトされ、関連するシャードで繰り返されます。

コーディネータ・データベースでの問合せのコンパイル中に、問合せコンパイラはシャーディング・キーに対する述語を分析し、関与するシャード(つまり、問合せで参照されるシャード表の行があるシャード)を識別するために使用できる述語を抽出します。残りのシャードは除外されたシャードと呼ばれます。

関与するシャードが1つのみ識別された場合は、問合せ全体がそのシャードにルーティングされて実行されます。これは単一シャード問合せと呼ばれます。

関与する複数のシャードがある場合、その問合せはマルチシャード問合せと呼ばれ、リライトされます。リライト処理では、問合せによって計算される式および問合せの形態が考慮されます。

単一シャード問合せのための問合せ処理

単一シャード問合せは、1つのシャードでのみデータをスキャンする必要があり、他のシャードでデータを検索する必要がない問合せです。

単一シャード問合せは、特定のシャードに接続してそのシャードに問合せを発行するクライアントに似ています。このシナリオでは、問合せ全体が関与する単一のシャードで実行され、コーディネータは処理された行をクライアントに返すだけです。コーディネータの計画は、リモート・マップ・カーソルに似ています。

たとえば、次の問合せは、顧客123のデータが1つのシャードのみにあるため、そのシャードにすべてマップされます。

```
SELECT count(*) FROM customers c, orders o WHERE c.custno = o.custno and c.custno = 123;
```

この問合せには、問合せのコンパイル時(リテラル)または問合せ開始時(バインド)に判明する1つのシャードにのみマップするシャード・キーの条件が含まれています。問合せは該当するシャードですべて実行されます。

単一シャード問合せでは次のものがサポートされます。

- 等価およびINリスト(Area = 'West'など)
- リテラル、バインド、またはリテラルとバインドの式を含む次のような条件

  ```
  Area = :bind
  Area = CASE :bind <10 THEN 'West' ELSE 'East' END
  ```

- SELECT、UPDATE、DELETE、INSERT、FOR UPDATEおよびMERGE。UPSERTはサポートされていません。

マルチシャード問合せのための問合せ処理

マルチシャード問合せは、複数のシャードでデータをスキャンする必要がある問合せであり、各シャードでの処理は他のシャードから独立しています。

マルチシャード問合せは、複数のシャードにマップされ、結果をクライアントに送信する前に、コーディネータが処理する必要がある場合があります。たとえば、次の問合せは各顧客によって発注されたオーダー数を取得します。

```
SELECT count(*), c.custno FROM customers c, orders o WHERE c.custno = o.custno
 GROUP BY c.custno;
```

この問合せは、コーディネータによって次のように変換されます。

```
SELECT sum(count_col), custno FROM (SELECT count(*) count_col, c.custno
 FROM customers c, orders o
 WHERE c.custno = o.custno GROUP BY c.custno) GROUP BY custno;
```

インライン問合せブロックは、リモート・マップ問合せブロックのように各シャードにマップされます。コーディネータは、すべてのシャードからの結果セットに対して、さらなる集計およびGROUP BYを実行します。各シャードでの実行の単位はインライン問合せブロックです。

マルチシャード問合せとグローバルな読取り一貫性

マルチシャード問合せでは、すべてのシャードで最も大きい共通SCNで問合せを発行することによって、グローバルな読込み一貫性(CR)を維持する必要があります。一貫性レベルを設定する方法の詳細は、マルチシャード問合せでの一貫性レベルの指定を参照してください。

マルチシャード問合せでのヒントの受け渡し

コーディネータに対する元の問合せで指定したヒントは、シャードに伝播されます。

マルチシャード問合せでの実行速度低下のトレースとトラブルシューティング

クエリー・リライトおよびシャード・プルーニングをトレースするには、コーディネータに対してトレース・イベントshard_sqlを設定します。観測される一般的なパフォーマンス問題の1つは、シャーディングの一定の制限により、GROUP BYがシャードにプッシュされ

ない場合です。考えられるすべての操作がシャードにプッシュされ、シャードからの結果を統合するためのコーディネータでの処理が最低限であるかどうかを確認します。

マルチシャード問合せでの一貫性レベルの指定

シャード間でマルチシャード問合せを実行するときに、初期化パラメータMULTISHARD_QUERY_DATA_CONSISTENCYを使用して様々な一貫性レベルを設定できます。

マルチシャード問合せでは、様々な一貫性レベルを指定できます。たとえば、一部の問合せでシャード間のSCN同期のコストを回避する必要がある場合は、それらのシャードをグローバルに分散できます。別のユース・ケースとして、レプリケーション用のスタンバイを使用している場合は、プライマリとそのスタンバイから結果がフェッチされる可能性があるため、マルチシャード問合せで少し古いデータが許容されます。

デフォルトのモードは厳密な一貫性であり、すべてのシャード間でSCN同期が実行されます。他のモードでは、SCN同期はスキップされます。delayed_standby_allowedレベルでは、ロード・バランシングなどの要因に応じてスタンバイからもデータをフェッチでき、古いデータを含めることができます。

このパラメータは、システム・レベルまたはセッション・レベルで設定できます。

関連項目:

MULTISHARD_QUERY_DATA_CONSISTENCYの使用方法の詳細は、Oracle Databaseリファレンスを参照してください。

サポートされる問合せ構成と問合せ形態の例

Oracle Shardingでは、単一シャード問合せとマルチシャード問合せの形態がサポートされていますが、いくつかの制限事項があります。

Oracle Shardingでの問合せコンストラクトに関する制限事項は、次のとおりです。

- CONNECT BYを使用する問合せ CONNECT BYを使用する問合せはサポートされません。
- MODEL句 MODEL句はサポートされていません。
- WHERE句のユーザー定義のPL/SQL マルチシャード問合せでは、ユーザー定義のPL/SQLはSELECT句でのみ使用できます。WHERE句で指定された場合は、エラーがスローされます。
- XLATEおよびXML問合せタイプ XLATEおよびXML問合せタイプの列はサポートされません。
- オブジェクト型 オブジェクト型をSELECTリスト、WHERE句などに含めることはできますが、オブジェクト型のカスタム・コンストラクタおよびメンバー関数をWHERE句に含めることはできません。

  さらに、重複表の場合、NOT FINAL型(つまりNOT FINALキーワードを指定して作成されたオブジェクト型)は、列のデータ型として使用できません。シャード表の場合、NOT FINAL型を列のデータ型として使用できますが、キーワードNOT SUBSTITUTABLE AT ALL LEVELSを指定して列を作成する必要があります。

ノート:

重複表のみに関連する問合せは、コーディネータで実行されます。

次のトピックでは、Oracle Shardingでサポートされる問合せ形態の例をいくつか示します。

シャード表のみに対する問合せ

単一表の問合せの場合、問合せにはシャードを限定するシャーディング・キーに対する等価フィルタを指定できます。結合問合せの場合は、すべての表がシャーディング・キーに対する等価フィルタを使用して結合される必要があります。

次の例は、シャード表のみが関与する問合せを示しています。

例7-1 内部結合

```
SELECT … FROM s1 INNER JOIN s2 ON s1.sk=s2.sk
WHERE any_filter(s1) AND any_filter(s2)
```

例7-2 左外部結合

```
SELECT … FROM s1 LEFT OUTER JOIN s2 ON s1.sk=s2.sk
```

例7-3 右側外部結合

```
SELECT … FROM s1 RIGHT OUTER JOIN s2 ON s1.sk=s2.sk
```

例7-4 完全外部結合

```
SELECT … FROM s1 FULL OUTER JOIN s2 ON s1.sk=s2.sk
WHERE any_filter(s1) AND any_filter(s2)
```

シャード表および重複表の両方が関係する問合せ

シャード表と重複表が関係する問合せは、シャーディング・キーに対する述語に応じて、単一シャード問合せまたはマルチシャード問合せになります。唯一の違いは、問合せに非シャード表が含まれていることです。

ノート:

シャード表と重複表の結合では、任意の列で、任意の比較演算子(= < > <= >=)または任意の結合式を使用できます。

例7-5 内部結合

```
SELECT … FROM s1 INNER JOIN r1 ON any_join_condition(s1,r1)
WHERE any_filter(s1) AND any_filter(r1)
```

例7-6 左外部結合または右外部結合

この場合、シャード表はLEFT OUTER JOINの最初の表です。

```
SELECT … FROM s1 LEFT OUTER JOIN r1 ON any_join_condition(s1,r1)
WHERE any_filter(s1) AND any_filter(r1)
```

```
SELECT … FROM r1 LEFT OUTER JOIN s1 ON any_join_condition(s1,s2)
AND any_filter(r1) AND filter_one_shard(s1)
```

この場合、シャード表はRIGHT OUTER JOINの2番目の表です。

```
SELECT … FROM r1 RIGHT OUTER JOIN s1 ON any_join_condition(s1,r1)
WHERE any_filter(s1) AND any_filter(r1)
SELECT … FROM s1 RIGHT OUTER JOIN r1 ON any_join_condition(s1,s2)
AND filter_one_shard(s1) AND any_filter(r1)
```

場合によっては、重複表がLEFT OUTER JOINの最初の表であるか、シャード表が最初の表でシャーディング・キーに対するフィルタ述語に基づいて単一のシャードにマップされます。

```
SELECT … FROM r1 LEFT OUTER JOIN s1 ON any_join_condition(s1,s2)
AND any_filter(r1) AND any_filter(s1)
```

場合によっては、重複表がRIGHT OUTER JOINの2番目の表であるか、シャード表が2番目の表でシャーディング・キーに対するフィルタ述語に基づいて単一のシャードにマップされます。

```
SELECT … FROM s1 RIGHT OUTER JOIN r1 ON any_join_condition(s1,s2)
AND any_filter (s1) AND any_filter(r1)
```

例7-7 完全外部結合

```
SELECT … FROM s1 FULL OUTER JOIN r1 ON s1.sk=s2.sk
WHERE any_filter(s1) AND any_filter(s2)
```

この場合、シャード表で複数のシャードへのアクセスが必要となります。

```
SELECT … FROM s1 FULL OUTER JOIN r1 ON s1.non_sk=s2.non_sk
WHERE any_filter(s1) AND any_filter(s2)
```

例7-8 セミ結合(EXISTS)

```
SELECT … FROM s1 EXISTS
(SELECT 1 FROM r1 WHERE r1.anykey=s1.anykey)
SELECT … FROM r1 EXISTS
(SELECT 1 FROM s1 WHERE r1.anykey=s1.anykey and filter_one_shard(s1))
```

この場合、シャード表は複数シャードの関与を必要とする副問合せにあります。

```
SELECT … FROM r1 EXISTS
(SELECT 1 FROM s1 WHERE r1.anykey=s1.anykey)
```

例7-9 アンチ結合(NOT EXISTS)

```
SELECT … FROM s1 NOT EXISTS
(SELECT 1 FROM r1 WHERE r1.anykey=s1.anykey)
```

この場合、シャード表は副問合せにあります。

```
SELECT … FROM r1 NOT EXISTS
(SELECT 1 FROM s1 WHERE r1.anykey=s1.anykey
```

Oracle Shardingでサポートされる集計関数

Oracle Shardingでは、次の集計はプロキシ・ルーティングによってサポートされます。

- COUNT
- SUM
- MIN
- MAX
- AVG

ユーザー定義型を使用した問合せ

ユーザー定義のSQLオブジェクト型とSQLコレクション型は、ユーザー定義型と呼ばれます。Oracle Shardingでは、ユーザー定義型を使用した問合せがサポートされます。

例7-10 ユーザー定義型を持つ表の作成

次の例では、全シャード・タイプおよびタイプ本体を作成し、そのタイプを参照するシャード表を作成します。

```
ALTER SESSION ENABLE SHARD DDL;
CREATE OR REPLACE TYPE person_typ AS OBJECT (
    first_name   VARCHAR2(20),
    last_name    VARCHAR2(25),
    email        VARCHAR2(25),
    phone        VARCHAR2(20),
    MEMBER FUNCTION details (
    self IN person_typ
    ) RETURN VARCHAR2
);
/
CREATE OR REPLACE TYPE BODY person_typ AS
    MEMBER FUNCTION details (
    self IN person_typ
    ) RETURN VARCHAR2 IS
        result VARCHAR2(100);
    BEGIN
        result := first_name || ' ' || last_name || ' ' || email || ' ' || phone;
        RETURN result;
    END;
END;
/
CREATE SHARDED TABLE Employees
( Employee_id      NUMBER NOT NULL
, person      person_typ
, signup_date DATE NOT NULL
, CONSTRAINT RootPK PRIMARY KEY(CustNo)
)
PARTITION BY CONSISTENT HASH (CustNo)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

例7-11 型コンストラクタを使用したデータの挿入

```
INSERT INTO Employees values ( 1, person_typ('John', 'Doe', 'jdoe@example.com', '123-456-7890'),
to_date('24 Jun 2020', 'dd Mon YYYY'));
```

例7-12 ユーザー定義型の列のマルチシャード問合せ

```
SELECT e.person FROM Employees e;
SELECT e.person.first_name, e.person.last_name FROM Employees e;
SELECT e.person.details() FROM Employee e where e.person.first_name = 'John' ;

SELECT signup_date from Employees e where e.person = person_typ('John', 'Doe', 'jdoe@example.com',
'123-456-7890' );
```

プロキシ・ルーティング用の実行計画

マルチシャード問合せでは、各シャードで別個の実行計画が生成されます。この実行計画はデータのサイズに応じて最適化され、シャードで使用可能なリソースが計算されます。

SQLフラグメントの実行計画を参照するために、個々のシャードに接続する必要はありません。
dbms_xplan.display_cursor()で提供されるインタフェースは、シャードで実行されたSQLセグメントの計画をコーディネータに表示します。また、[V/X]$SHARD_SQLは、マルチシャード問合せのシャードSQLフラグメントをターゲットのシャード・データベースに一意にマップします。

dbms_xplan.display_cursor()のSQLセグメント・インタフェース

2つのインタフェースを使用して、シャードで実行されたSQLセグメントの計画を表示できます。これらのインタフェースは、引数としてシャードIDを取り、指定されたシャードの計画を表示します。ALL_SHARDS形式では、すべてのシャードの計画が表示されます。

シャードのすべての計画を出力するには、次に示すようにformat値ALL_SHARDSを使用します。

```
select * from table(dbms_xplan.display_cursor(sql_id=>:sqlid,
                                              cursor_child_no=>:childno,
                                              format=>'BASIC +ALL_SHARDS ',
                                              shard_ids=>shard_ids))
```

シャードの計画を選択して出力するには、display_cursor()関数内でシャードIDを渡します。複数のシャードの計画を出力する場合は、次に示すようにshard_idsパラメータでシャードIDを含む数値の配列を渡します。

```
select * from table(dbms_xplan.display_cursor(sql_id=>:sqlid,
                                               cursor_child_no=>:childno,
                                               format=>'BASIC',
                                               shard_ids=>ids))
```

1つのシャードの計画を返すには、次に示すようにshard_idパラメータにシャードIDを直接渡します。

```
select * from table(dbms_xplan.display_cursor(sql_id=>:sqlid,
                                              cursor_child_no=>:childno,
                                              format=>'BASIC',
                                              shard_id=>1))
```

V$SQL_SHARD

V$SQL_SHARDは、マルチシャード問合せのシャードSQLフラグメントをターゲットのシャード・データベースに一意にマップします。こ

のビューは、特定のマルチシャード問合せのシャードSQLフラグメントごとにアクセスされたシャードのリストがシャード・コーディネータ・データベースに格納されている場合にのみ関連します。マルチシャード問合せを実行するたびに、異なるシャードのセットに対してシャードSQLフラグメントを実行できるため、実行のたびにシャードIDが更新されます。このビューには、各リモート・ノードのシャードSQLフラグメントのSQL IDと、シャードSQLフラグメントが実行されたシャードIDが保持されています。

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
 SQL_ID                                            VARCHAR2(13)
 CHILD_NUMBER                                      NUMBER
 NODE_ID                                           NUMBER
 SHARD_SQL_ID                                      VARCHAR2(13)
 SHARD_ID                                          NUMBER
 SHARD_CHILD_NUMBER                                NUMBER
```

- SQL ID - コーディネータ上のマルチシャード問合せのSQL ID
- CHILD_NUMBER - コーディネータ上のマルチシャード問合せのカーソル子番号
- NODE - マルチシャード問合せのシャードSQLフラグメントのリモート・ノードのID
- SHARD_SQL_ID - 指定されたリモート・ノードIDに対するシャードSQLフラグメントのSQL ID
- SHARD_ID - シャードSQLフラグメントが実行されたシャードのID
- SHARD_CHILD_NUMBER - シャードに対するシャードSQLフラグメントのカーソル子番号(デフォルトは0)

シャード・データベースに対するマルチシャード問合せと実行計画の例を次に示します。

```
SQL> select count(*) from departments a where exists (select distinct department_id
 from departments b where b.department_id=60);
-----------------------------------------------
| Id  | Operation          | Name             |
-----------------------------------------------
|   0 | SELECT STATEMENT   |                  |
|   1 |  SORT AGGREGATE    |                  |
|   2 |   FILTER           |                  |
|   3 |    VIEW            | VW_SHARD_377C5901 |
|   4 |     SHARD ITERATOR |                  |
|   5 |      REMOTE        |                  |
|   6 |    VIEW            | VW_SHARD_EEC581E4 |
|   7 |     SHARD ITERATOR |                  |
|   8 |      REMOTE        |                  |
-----------------------------------------------
```

V$SQL_SHARDビューに対するSQL_IDの問合せ。

```
SQL> Select * from v$sql_shard where SQL_ID =  '1m024z033271u' ;
SQL_ID        NODE_ID   SHARD_SQL_ID  SHARD_ID
------------- -------  -------------- --------
1m024z033271u       5   5z386yz9suujt        1
1m024z033271u       5   5z386yz9suujt       11
1m024z033271u       5   5z386yz9suujt       21
1m024z033271u       8   8f50ctj1a2tbs       11
```

関連項目:

Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

Oracle Databaseデータベース・リファレンス

サポートされているDMLと例

Oracleシャーディングの DMLは、重複表またはシャード表のいずれかをターゲットにできます。ターゲットが重複表の場合、DMLに制限はありません。

シャード表をターゲットとするDML (主に挿入、更新および削除)は、次のようにできます。

- ターゲット表のみが参照される単純なDML
- 他の表を参照するDML
- MERGE文

ターゲット表のみが参照される単純なDML

次に、サポートされているDMLの例をいくつか示します。

例7-13 すべての行の更新

```
UPDATE employees SET salary = salary *1.1;
```

例7-14 単一行の挿入

```
INSERT INTO employees VALUES (102494, 'Jane Doe, ...
    );
```

例7-15 単一行の削除

```
DELETE employees WHERE employee_id = 103678;
```

他の表を参照するDML

シャード表のDMLは、他のシャード表、重複表またはローカル表を参照できます。

例7-16 重複表を参照するDML

この例では、employeesはシャード表で、ref_jobsは重複表です。

```
DELETE employees
            WHERE job_id IN (SELECT job_id FROM ref_jobs
                             WHERE job_id = 'SA_REP');
```

例7-17 別のシャード表を参照するDML

```
UPDATE departments SET department_name = 'ABC '
            WHERE department_id IN (SELECT department_id
                                    FROM employees
                                    WHERE salary < 10000);
```

例7-18 ローカル表からのINSERT AS SELECT

```
INSERT INTO employees SELECT * FROM local_employees;
```

例7-19 1つのシャードに影響するDML

DML文は、1つのシャードのみに影響することも、複数のシャードに影響することもあります。たとえば、ここに示すDELETE文は、WHERE句のシャーディング・キー(employee_id)に述語があるため、1つのシャードにのみ影響します。

```
DELETE employees WHERE employee_id = 103678;
```

例7-20 複数のシャードに影響するDML

次の文は、WHERE句がないため、EMPLOYEES表のすべての行に影響します。

```
UPDATE employees SET salary = salary *1.1;
```

このUPDATE文をすべてのシャードで実行するために、シャード・コーディネータがすべてのプライマリ・データベースに対して反復処理を行い、UPDATE文のリモート実行を呼び出します。コーディネータは分散トランザクションを開始し、2フェーズ・コミットを実行して分散トランザクションの一貫性を保証します。インダウト・トランザクションがある場合は、手動でリカバリする必要があります。

MERGE文の例

MERGE文で、シャード表または重複表をターゲットにできます。MERGE操作自体をシャードにプッシュできるかぎり、マージは許可されます。

例7-21 シャード表employeesをターゲット表とするMERGE文

この例では、employee_id列がシャーディング・キーで、ソース問合せの結合述語がシャーディング・キーにあるため、MERGE文は実行されるすべてのシャードにプッシュされます。

```
MERGE INTO employees D
   USING (SELECT employee_id, salary, department_id FROM employees
   WHERE department_id = 80) S
   ON (D.employee_id = S.employee_id)
   WHEN MATCHED THEN UPDATE SET D.salary = D.salary + S.salary*.01
     DELETE WHERE (S.salary > 8000)
   WHEN NOT MATCHED THEN INSERT (D.employee_id, D.salary)
     VALUES (S.employee_id, S.salary*0.1)
     WHERE (S.salary <= 8000);
```

例7-22 重複表をターゲット表とするMERGE文

この例では、ターゲット表は重複表ref_employeesです。ソース問合せはシャード表employeesを参照し、結合述語はシャーディング・キーemployee_idにあります。

```
MERGE INTO ref_employees D
   USING (SELECT employee_id, salary, department_id FROM employees
   WHERE department_id = 80) S
   ON (D.employee_id = S.employee_id)
   WHEN MATCHED THEN UPDATE SET D.salary = D.salary + S.salary*.01
     DELETE WHERE (S.salary > 8000)
   WHEN NOT MATCHED THEN INSERT (D.employee_id, D.salary)
     VALUES (S.employee_id, S.salary*0.1)
      WHERE (S.salary <= 8000);
```

マルチシャードDMLのサポートの制限事項

次のDML機能は、Oracle ShardingのマルチシャードDMLではサポートされません。

- パラレルDML パラレルDMLはマルチシャードDMLではサポートされません。マルチシャードDMLでは、DMLは常に一度に1つのシャードに対して(シリアルに)実行されます。
- エラー・ロギング DMLのERROR LOG句は、マルチシャードDMLではサポートされません。この場合、ユーザー・エラーが発生します。
- 配列DML 配列DMLはマルチシャードDMLではサポートされません。この場合、ORA-2681が発生します。
- RETURNING句RETURNING INTO句は通常の分散DMLでサポートされないため、Oracle Shardingでサポートされません。マルチシャードDMLでRETURNING INTO句の使用を試みると、ORA-22816が発生します。
- MERGEおよびUPSERT MERGE文はOracle Shardingで部分的にサポートされます。つまり、単一のシャードのみに影響するMERGE文はサポートされます。MERGE文で複数のシャードの変更が必要となる場合は、ORAエラーが発生します。
- 複数表へのINSERT 複数表への挿入はデータベース・リンクでサポートされないため、Oracle Shardingでサポートされません。
- 更新可能な結合ビュー 更新可能な結合ビューでシャード表がシャーディング・キーで結合されている場合、ORA-1779がスローされます。このエラーの理由は、シャード表に定義されている主キーが内部列SYS_HASHVAL + シャーディング・キーの組合せであるため、更新可能な結合ビューにSYS_HASHVALを指定できないからです。この制限により、キー保存表を設定できないため、ORA-1779が発生します。
- トリガー

シャード表のオプティマイザ統計の収集

コーディネータ・データベースからシャード表の統計を収集できます。

統計プリファレンス・パラメータCOORDINATOR_TRIGGER_SHARDがすべてのシャードでTRUEに設定されている場合、コーディネータ・データベースはシャードで収集された統計をインポートできます。

PL/SQLプロシージャDBMS_STATS.GATHER_SCHEMA_STATS()およびDBMS_STATS.GATHER_TABLE_STATS()は、シャードおよびコーディネータ・データベース内のシャード表および重複表に関する統計を収集します。
REPORT_GATHER_TABLE_STATS関数も参照してください。

手動統計収集

1. すべてのシャードでCOORDINATOR_TRIGGER_SHARDをTRUEに設定します。

   このステップは、1回のみシャードで実行されます。たとえば、sharduserという名前のスキーマがあるとします。

   ```
   connect / as sysdba
   EXECUTE DBMS_STATS.SET_SCHEMA_PREFS('SHARDUSER','COORDINATOR_TRIGGER_SHARD','TRUE');
   ```

2. シャード全体の統計を収集します。

   ユーザーは全シャード・ユーザーであり、ディクショナリ表にアクセスする権限を持っている必要があります。

   a. シャードで、次を実行します。

      ```
      connect sharduser/password
      EXEC DBMS_STATS.GATHER_SCHEMA_STATS(ownname => 'SHARDUSER', options => 'GATHER');
      ```

b. すべてのシャードが完了したら、集計統計をプルするためにコーディネータで次を実行します。

```
connect sharduser/password
EXEC DBMS_STATS.GATHER_SCHEMA_STATS(ownname => 'SHARDUSER', options => 'GATHER');
```

c. すべてのシャードの統計を確認します。

```
connect sharduser/password
ALTER SESSION SET nls_date_format='DD-MON-YYYY HH24:MI:SS';
  col TABLE_NAME form a40
  set pagesize 200 linesize 200
SELECT TABLE_NAME, NUM_ROWS, sharded, duplicated, last_analyzed
  FROM user_tables
  WHERE table_name not like 'MLOG%' and table_name not like 'RUPD%'
  and table_name not like 'USLOG%';
```

自動統計収集

1. すべてのシャードでCOORDINATOR_TRIGGER_SHARDをTRUEに設定します。

   このステップは、1回のみシャードで実行されます。たとえば、sharduserという名前のスキーマがあるとします。

```
connect / as sysdba
EXECUTE DBMS_STATS.SET_SCHEMA_PREFS('SHARDUSER','COORDINATOR_TRIGGER_SHARD','TRUE');
```

2. シャードおよびコーディネータ・データベースで集計された統計をプルするジョブをスケジュールします。

   ユーザーは全シャード・ユーザーであり、ディクショナリ表にアクセスする権限を持っている必要があります。

   シャードで次のジョブを開始します。

```
connect sharduser/password
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name => 'Gather_Stats_2',
   job_type => 'PLSQL_BLOCK',
   job_action => 'BEGIN DBMS_STATS.GATHER_SCHEMA_STATS(ownname => ''DEMO'', options =>
''GATHER''); END;',
   start_date => SYSDATE,
   repeat_interval =>
'freq=daily;byday=MON,TUE,WED,THU,FRI,SAT,SUN;byhour=14;byminute=10;bysecond=00',
   end_date => NULL,
   enabled => TRUE,
   comments => 'Gather table statistics');
END;
/
```

   すべてのシャードのジョブが終了したら、コーディネータで次のジョブを開始します。

```
connect sharduser/password
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name             => 'Gather_Stats_2',
   job_type             => 'PLSQL_BLOCK',
   job_action           => 'BEGIN DBMS_STATS.GATHER_SCHEMA_STATS(ownname => ''DEMO'', options
=> ''GATHER''); END;',
   start_date           =>  SYSDATE,
   repeat_interval      =>
'freq=daily;byday=MON,TUE,WED,THU,FRI,SAT,SUN;byhour=15;byminute=10;bysecond=00',
   end_date             =>  NULL,
```

```
   enabled              =>  TRUE,
   comments             => 'Gather table statistics');
END;
/
```

8 シャード・データベースのアプリケーションの開発

シャードへの直接ルーティング

Oracleクライアントおよび接続プールは、高パフォーマンスのデータ依存ルーティングのために接続文字列に指定されたシャーディング・キーを認識できます。接続レイヤーのシャード・ルーティング・キャッシュは、データが常駐するシャードにデータベース・リクエストを直接ルーティングするために使用されます。

シャードへの直接(キーベース)ルーティングでは、必要なトランザクションに関連するデータが含まれている単一の関連するシャードに対して、シャーディング・キーを使用して接続が確立されます。

接続のチェック時にユーザー・セッション・レベルでデータベース接続リクエストをルーティングするために、シャーディング・キーが使用されます。コンポジット・シャーディング方法では、シャーディング・キーおよびスーパー・シャーディング・キーの両方が必要となります。直接(キーベース)ルーティングでは、シャーディング・キー(またはスーパー・シャーディング・キー)を接続の一部として渡す必要があります。この情報に基づいて、指定されたシャーディング・キーまたはスーパー・シャーディング・キーに関係するデータが含まれている関連するシャードに対して、接続が確立されます。

シャードとのセッションが確立されると、すべてのSQL問合せおよびDMLが該当するシャードのスコープ内でサポートされ、実行されます。このルーティングは高速であり、シャード内トランザクションを実行するすべてのOLTPワークロードで使用されます。最高のパフォーマンスおよび可用性を必要とするすべてのOLTPワークロードに直接ルーティングを使用することをお薦めします。

Oracle Shardingをサポートするために、Oracle接続プールおよびドライバに対して重要な機能強化が行われました。JDBC、UCP (Universal Connection Pool)、OCI (OCI Session Pool)およびODP.NET (Oracle Data Provider for .NET)は、接続作成中にシャーディング・キーを渡すためのAPIを提供しています。Apache Tomcat、IBM Websphere、Oracle WebLogic ServerおよびJBOSSでは、JDBC/UCPのサポートを活用してシャーディングを使用できます。PHP、Python、PerlおよびNode.jsではOCIのサポートを活用できます。

シャード・トポロジ・キャッシュは、シャードとシャーディング・キーの範囲のマッピングです。Oracle統合接続プールは、このシャード・トポロジ・キャッシュをそのメモリー内に保持します。特定のシャードへの最初の接続時(プールの初期化時またはプールが新しいシャードに接続するとき)に、シャーディング・キーの範囲のマッピングがシャードから収集されて、シャード・トポロジ・キャッシュが動的に作成されます。

シャード・トポロジをキャッシュすると、シャードへの高速なパスが作成され、シャードへの接続を作成するプロセスが迅速になります。シャーディング・キーを使用して接続リクエストが実行されると、接続プールはこの特定のシャーディング・キーが存在する対応するシャードを検索します(トポロジ・キャッシュから)。一致する接続がプールで利用できる場合、プールは内部接続選択アルゴリズムを適用することによって、シャードへの接続を返します。

キャッシュされたトポロジ・マップに存在する特定のシャーディング・キーへのデータベース接続リクエストは、そのシャードに直接送られます(つまり、シャード・ディレクタがバイパスされます)。また、接続プールは、SDBからRLB通知をサブスクライブし、実行時ロード・バランシング・アドバイザに基づいて最適な接続を分配します。接続が確立されると、クライアントはシャードに対するトランザクションを直接実行します。指定したシャーディング・キーのすべてのトランザクションが実行されたら、アプリケーションはその接続をプールに返し、別のキーの接続を取得する必要があります。

一致する接続がプールにない場合は、シャーディング・キーとともに接続リクエストをシャード・ディレクタに転送することによって、新しい接続が作成されます。

プールが初期化され、シャード・トポロジ・キャッシュがすべてのシャードに基づいて作成されると、シャード・ディレクタが停止しても直接ルーティングに影響しなくなります。

既存のアプリケーションのシャーディングに対する適合性

シャード・アーキテクチャの利点を享受するには、シャーディングを使用する予定がない既存のアプリケーションで、一定のレベルの再設計が必要となります。

シャーディング・キーを渡すのみで済む簡単な場合や、シャード・データベースで必要となるデータおよびワークロードの水平パーティション化を行うことができない場合があります。

電子商取引、モバイル、ソーシャル・メディアなど、多くの顧客対応Webアプリケーションはシャーディングに適切です。そのようなアプリケーションには、適切に定義されたデータ・モデルおよびデータ分散方法(ハッシュ、範囲、リストまたはコンポジット)があり、主にシャーディング・キーを使用してデータにアクセスします。シャーディング・キーの例として顧客ID、アカウント番号、country_idなどがあります。通常、アプリケーションでは、シャーディングが十分に機能するためにデータの部分的な非正規化も必要となります。

シャーディング・キーの単一値に関連付けられたデータにアクセスするトランザクションは、シャード・データベースの主なユースケースです。たとえば、顧客のレコード、サブスクライバのドキュメント、財務トランザクション、電子商取引トランザクションなどの検索および更新などです。同じ値のシャーディング・キーを持つ、シャード・スキーマ内のすべての行は同じシャード上にあることが保証されるため、そのようなトランザクションは常に単一のシャードで最高のパフォーマンスで実行され、最高レベルの一貫性となります。

マルチシャード操作はサポートされますが、パフォーマンスおよび一貫性のレベルは低くなります。そのようなトランザクションには単純な集計、レポートなどが含まれ、単一シャードのトランザクションが優位を占めるワークロードに比べて、シャード・アプリケーションで重要度の低い役割を果たします。

直接ルーティングをサポートするシャーディングAPI

Oracle接続プールおよびドライバはOracle Shardingをサポートしています。

JDBC、UCP、OCIおよびOracle Data Provider for .NET (ODP.NET)は、接続チェックの一部としてシャーディング・キーを認識します。Apache Tomcat、WebsphereおよびWebLogicはシャーディングのためのUCPサポートを活用し、PHP、Python、PerlおよびNode.jsはOCIサポートを活用します。

Oracle Sharding対応のOracle JDBC API

Oracle Java Database Connectivity (JDBC)には、Oracle Sharding構成内のデータベース・シャードに接続するために使用できるAPIがあります。

JDBCドライバは指定されたシャーディング・キーおよびスーパー・シャーディング・キーを認識し、データが含まれている該当シャードに接続します。シャードに対する接続が確立されると、DML、SQL問合せなどのサポートされているデータベース操作は通常ど

おりに実行されます。

シャード対応アプリケーションは、データベース・シャーディングAPIを使用してシャーディング・キーを指定することで特定のシャードに接続します。

- OracleShardingKeyインタフェースは、現在のオブジェクトがOracleのシャード・データベースで使用されるOracleシャーディング・キーを表していることを示します。
- OracleShardingKeyBuilderインタフェースは、サポートされている様々なデータ型のサブキーを持つ複合シャーディング・キーを作成します。このインタフェースでは、新しいJDK 8ビルダー・パターンを使用してシャーディング・キーを作成します。
- OracleConnectionBuilderインタフェースは、ユーザー名とパスワード以外の追加パラメータを使用して、接続オブジェクトを構築します。
- OracleDataSourceクラスにより、createConnectionBuilderメソッドとcreateShardingKeyBuliderメソッドはデータベース・シャードのサポートが可能になります。
- OracleXADataSourceクラスにより、createConnectionBuilderメソッドはデータベース・シャードのサポートが可能になります。
- OracleConnectionクラスにより、setShardingKeyIfValidメソッドとsetShardingKeyメソッドはデータベース・シャードのサポートが可能になります。
- OracleXAConnectionクラスにより、setShardingKeyIfValidメソッドとsetShardingKeyメソッドはデータベース・シャードのサポートが可能になります。

詳細と例は、『*Oracle Database JDBC*開発者ガイド』を参照してください。

例8-1 JDBCを使用したシャード対応アプリケーション・コードの例

次のコードでは、JDBCシャーディングAPIの使用方法を示します

```
OracleDataSource ods = new OracleDataSource();

ods.setURL("jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(HOST=myhost)(PORT=1521)(PROTOCOL=tcp))(CONNECT_DA
TA=(SERVICE_NAME=myorcldbservicename)))");
   ods.setUser("hr");
   ods.setPassword("hr");

  // Employee name is the sharding Key in this example.
  // Build the Sharding Key using employee name as shown below.
   OracleShardingKey employeeNameShardKey = ods.createShardingKeyBuilder()
                                              .subkey("Mary", JDBCType.VARCHAR)// First Name
                                              .subkey("Claire", JDBCType.VARCHAR)// Last Name
                                              .build();
   OracleShardingKey locationSuperShardKey = ods.createShardingKeyBuilder() // Building a super
sharding key using location as the key
                                               .subkey("US", JDBCType.VARCHAR)
                                               .build();
   OracleConnection connection = ods.createConnectionBuilder()
                                    .shardingKey(employeeNameShardKey)
                                    .superShardingKey(locationSuperShardKey)
                                    .build();
```

関連項目

- 『Oracle Database JDBC開発者ガイド』のデータベース・シャーディングに対するJDBCサポートに関する項

Oracle Sharding対応のOracle Call Interface

Oracle Call Interface (OCI)には、Oracle Sharding構成内のデータベース・シャードに接続するために使用できるインタフェースがあります。

チャンクを対象に読取りまたは書込みを行う要求を作成するには、接続開始ステップでそのチャンクを格納している適切なデータベース(シャード)にアプリケーションをルーティングする必要があります。このルーティングを行うには、データ・キーを使用します。データ・キーにより、特定のチャンクへのルーティング(シャーディング・キーを指定)またはチャンクのグループへのルーティング(スーパー・シャーディング・キーを指定)が可能になります。

操作対象のチャンクを含む適切なシャードに接続するためには、アプリケーションでキーを指定してから、シャードされたOracle Databaseへの接続(スタンドアロン接続またはOCIセッション・プールから取得された接続)を取得する必要があります。OCIセッション・プールでは、プールから接続をチェックアウトする前にデータ・キーを指定する必要があります。

大まかに言うと、シャーディング・キーとシャード・グループ・キーを構成して、基礎となる接続でセッションを取得するには、次のステップを実行する必要があります。

1. シャーディング・キー記述子を割り当てるため、OCIDescriptorAlloc()をコールし、シャーディング・キーを構成するためにOCI_DTYPE_SHARDING_KEYとして記述子型パラメータを指定します。
2. シャード・グループ・キー記述子を割り当てるため、OCIDescriptorAlloc()をコールし、シャード・グループ・キーを構成するためにOCI_DTYPE_SHARDING_KEYとして記述子型パラメータを指定します。
3. シャーディング・キーおよびシャード・グループ・キー情報が含まれる前のステップの初期化済認証ハンドルを使用してOCISessionGet()をコールし、シャーディング・キーで指定されたシャードおよびチャンクと、シャード・グループ・キーで指定されたチャンクのグループに対するデータベース接続を取得します。

OCIセッション・プールへの接続、スタンドアロン接続、およびカスタム・プール接続の作成の詳細は、『*Oracle Call Interface* プログラマーズ・ガイド』を参照してください。

関連項目

- 『Oracle Call Interfaceプログラマーズ・ガイド』のシャードを使用するためのOCIインタフェースに関する項

Oracle Sharding対応のOracle Universal Connection Pool API

Oracle Universal Connection Pool (UCP)には、Oracle Sharding構成内のデータベース・シャードに接続するために使用できるAPIがあります。

シャード対応のアプリケーションは、createShardingKeyBuilderおよびcreateConnectionBuilderという拡張シャーディングAPIを使用してシャーディング・キーを指定することによって、特定のシャードへの接続を取得します。

大まかに言うと、アプリケーションがシャード・データベースと連携するようにするには、次のステップに従う必要があります。

1. シャード・ディレクタおよびグローバル・サービスが反映されるようにURLを更新します。
2. 次のプール・パラメータをプール・レベルおよびシャード・レベルで設定します。
    - setInitialPoolSizeでは、UCPの開始時に作成する接続の初期数を設定します。
    - setMinPoolSizeでは、実行時にプールで維持する接続の最小数を設定します。

- setMaxPoolSizeでは、接続プールに許容される接続の最大数を設定します
- setMaxConnectionsPerShardでは、シャードごとの最大接続数を設定します

3. createShardingKeyBuilderを使用してシャーディング・キー・オブジェクトを作成します。

4. createConnectionBuilderを使用して接続を確立します。

5. 特定のシャードのスコープ内でトランザクションを実行します。

例8-2 UCPシャーディングAPIを使用した接続の確立

シャーディング・キーを作成し、UCPシャーディングAPIコールを使用して接続を確立する方法を示すコード片を次に示します。

```
...
PoolDataSource pds =
     PoolDataSourceFactory.getPoolDataSource();

  // Set Connection Pool properties
pds.setURL(DB_URL);
pds.setUser("hr");
pds.setPassword("****");
pds.setInitialPoolSize(10);
pds.setMinPoolSize(20);
pds.setMaxPoolSize(30);

// build the sharding key object
OracleShardingKey shardingKey =
    pds.createShardingKeyBuilder()
      .subkey("mary.smith@example.com", OracleType.VARCHAR2)
      .build();
  // Get an UCP connection for a shard
Connection conn =
    pds.createConnectionBuilder()
     .shardingKey(shardingKey)
     .build();
...
```

例8-3 UCP接続プールを使用したシャード対応アプリケーション・コードの例

この例では、プール設定をプール・レベルおよびシャード・レベルで定義しています。

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import oracle.jdbc.OracleShardingKey;
import oracle.jdbc.OracleType;
import oracle.jdbc.pool.OracleDataSource;
import oracle.ucp.jdbc.PoolDataSource;
import oracle.ucp.jdbc.PoolDataSourceFactory;
public class MaxConnPerShard
{
  public static void main(String[] args) throws SQLException
  {
    String url = "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(HOST=shard-dir1)(PORT=3216)
 (PROTOCOL=tcp))(CONNECT_DATA=(SERVICE_NAME=shsvc.shpool.oradbcloud)(REGION=east)))";
    String user="testuser1", pwd = "testuser1";
```

```
    int maxPerShard = 100, initPoolSize = 20;
    PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
    pds.setConnectionFactoryClassName(OracleDataSource.class.getName());
    pds.setURL(url);
    pds.setUser(user);
    pds.setPassword(pwd);
    pds.setConnectionPoolName("testpool");
    pds.setInitialPoolSize(initPoolSize);
    // set max connection per shard
    pds.setMaxConnectionsPerShard(maxPerShard);
    System.out.println("Max-connections per shard is: "+pds.getMaxConnectionsPerShard());

    // build the sharding key object
    int shardingKeyVal = 123;
    OracleShardingKey sdkey = pds.createShardingKeyBuilder()
        .subkey(shardingKeyVal, OracleType.NUMBER)
        .build();
    // try to build maxPerShard connections with the sharding key
    Connection[] conns = new Connection[maxPerShard];
    for (int i=0; i<maxPerShard; i++)
    {
      conns[i] = pds.createConnectionBuilder()
          .shardingKey(sdkey)
          .build();

Statement stmt = conns[i].createStatement();
      ResultSet rs = stmt.executeQuery("select sys_context('userenv', 'instance_name'),
       sys_context('userenv', 'chunk_id') from dual");
      while (rs.next()) {
        System.out.println((i+1)+" - inst:"+rs.getString(1)+", chunk:"+rs.getString(2));
      }
      rs.close();
      stmt.close();
    }
    System.out.println("Try to build "+(maxPerShard+1)+" connection ...");
    try {
      Connection conn = pds.createConnectionBuilder()
          .shardingKey(sdkey)
          .build();
      Statement stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery("select sys_context('userenv', 'instance_name'),
       sys_context('userenv', 'chunk_id') from dual");
      while (rs.next()) {
        System.out.println((maxPerShard+1)+" - inst:"+rs.getString(1)+",
         chunk:"+rs.getString(2));
      }
      rs.close();
      stmt.close();
      System.out.println("Problem!!! could not build connection as max-connections per
        shard exceeded");
      conn.close();
    } catch (SQLException e) {
      System.out.println("Max-connections per shard met, could not build connection
        any more, expected exception: "+e.getMessage());
    }
    for (int i=0; i<conns.length; i++)
    {
      conns[i].close();
```

```
    }
  }
}
```

関連項目

- 『Oracle Universal Connection Pool開発者ガイド』のデータベース・シャーディング・サポートのためのUCP APIに関する項

Oracle Sharding対応のOracle Data Provider for .NET API

Oracle Data Provider for .NET (ODP.NET)には、Oracle Sharding構成内のデータベース・シャードに接続するために使用できるAPIがあります。

ODP.NET APIを使用すると、シャード対応アプリケーションはOracleConnectionクラスのSetShardingKey(OracleShardingKey shardingKey, OracleShardingKey superShardingKey)インスタンス・メソッドなどのAPIでシャーディング・キーおよびスーパー・シャーディング・キーを指定することで、特定のシャードへの接続を取得します。

大まかに言うと、.NETアプリケーションがシャード・データベースと連携するようにするには、次のステップを実行する必要があります。

1. ODP.NET管理対象外ドライバを使用します。

   シャーディングは、ODP.NET接続プールを使用する場合もODP.NET接続プールを使用しない場合もサポートされます。各プールは、シャード・データベースの異なるシャードへの接続を保持できます。

2. OracleShardingKeyクラスを使用して、シャーディング・キー、およびスーパー・シャーディング・キーの別のインスタンスを設定します。

3. ODP.NETが指定したシャーディング・キーおよびスーパー・シャーディング・キーを使用する接続を返すことができるように、OracleConnection.Open()を呼び出す前に、OracleConnection.SetShardingKey()メソッドを呼び出します。

   これらのキーは、OracleConnectionがクローズ状態のときに設定する必要があります。そうしないと、例外がスローされます。

例8-4 ODP.NETを使用したシャード対応アプリケーション・コードの例

```
using System;
using Oracle.DataAccess.Client;

class Sharding
{
  static void Main()
  {
    OracleConnection con = new OracleConnection
      ("user id=hr;password=hr;Data Source=orcl;");
    //Setting a shard key
    OracleShardingKey shardingKey = new OracleShardingKey(OracleDbType.Int32, 123);
    //Setting a second shard key value for a composite key
    shardingKey.SetShardingKey(OracleDbType.Varchar2, "gold");
    //Creating and setting the super shard key
    OracleShardingKey superShardingKey = new OracleShardingKey();
    superShardingKey.SetShardingKey(OracleDbType.Int32, 1000);

    //Setting super sharding key and sharding key on the connection
```

```
    con.SetShardingKey(shardingKey, superShardingKey);
    con.Open();
    //perform SQL query
  }
}
```

関連項目

- Oracle Data Provider for .NET開発者ガイド for Microsoft Windowsのデータベース・シャーディングに関する項

9 シャード・データベースの管理

Oracle Shardingは、シャード・データベースの管理のためのツールおよびいくつかの自動化を提供しています。

次のトピックでは、シャード・データベースの管理についてさらに詳しく説明します。

シャーディング対応スタックの管理

この項では、シャード・データベース構成内のコンポーネントの起動および停止について説明します。次の項目が含まれます。

シャーディング対応タックの起動

シャーディング対応スタックの推奨される起動順序は次のとおりです。

- シャード・カタログ・データベースとローカル・リスナーを起動します。
- シャード・ディレクタ(GSM)を起動します。
- シャード・データベースとローカル・リスナーを起動します。
- グローバル・サービスを起動します。
- 接続プールとクライアントを起動します。

シャーディング対応スタックの停止

シャーディング対応スタックの推奨される停止手順は次のとおりです。

- 接続プールとクライアントを停止します。
- グローバル・サービスを停止します。
- シャード・データベースとローカル・リスナーを停止します。
- シャード・ディレクタ(GSM)を停止します。
- シャード・カタログ・データベースとローカル・リスナーを停止します。

Oracle Shardingデータベース・ユーザーの管理

この項では、Oracle Shardingに固有のデータベース・ユーザーについて説明します。次の項目が含まれます。

GSMUSERアカウントについて

GSMUSERアカウントは、GDSCTLおよびグローバル・サービス・マネージャがGDS構成のデータベースに接続するために使用されます。

GSMUSERは、デフォルトでOracleデータベースに存在します。Oracle Sharding構成では、このアカウントはプール・データベースのかわりにシャードへの接続に使用され、アカウントがロック解除された後、SYSDGおよびSYSBACKUPシステム権限の両方を付与する必要があります。

GSMUSERアカウントに指定されたパスワードは、gdsctl add shardコマンドで使用されます。新しいシャードでGSMUSERにSYSDGおよびSYSBACKUPを付与しないと、gdsctl add shardがORA-1031: 権限が不足していますのエラーで失敗します。

gdsctl create shardコマンドを使用してDatabase Configuration Assistant (DBCA)で新しいシャードを作成すると、デプロイメント・プロセス中にGSMUSERアカウントに自動的にSYSDG権限とSYSBACKUP権限が付与され、ランダム・パスワードが割り当てられます。GSMUSERアカウントは対話的にログインする必要がないため、管理者がパスワードの値を認識する必要はありません。ただし、シャードでalter user SQLコマンドをgdsctl modify shard -pwdコマンドと組み合せて使用すると、必要に応じてパスワードを変更できます。

関連項目:

Global Data Services概要および管理ガイドのシャードの追加

GSMROOTUSERアカウントについて

GSMROOTUSERは、プラガブル・データベース(PDB)シャードが存在する場合にのみ使用されるOracle Shardingに固有のデータベース・アカウントです。アカウントは、GDSCTLおよびグローバル・サービス・マネージャがコンテナ・データベース(CDB)のルート・コンテナに接続して管理タスクを実行するために使用されます。

PDBシャードが使用されていない場合、GSMROOTUSERユーザーはロック解除されず、どのデータベースでもパスワードを割り当てられません。ただし、PDBシャードを含むシャード構成では、gdsctl add cdbコマンドを実行する前に、GSMROOTUSERをロック解除し、SYSDGおよびSYSBACKUP権限を付与する必要があります。GSMROOTUSERアカウントのパスワードは、CDBのルート・コンテナ内でalter user SQLコマンドをgdsctl modify cdb -pwdコマンドと組み合せて使用することで、デプロイメント後に必要に応じて変更できます。

関連項目:

Global Data Services概要および管理ガイドのcdbの追加

シャード・データベースのバックアップおよびリカバリ

シャードは個別のOracleデータベースでホストされるため、Oracle Maximum Availabilityのベスト・プラクティスを使用してシャードを個別にバックアップおよびリストアできます。

SDBの高可用性のためにData GuardおよびOracle Active Data Guardを使用している場合は、プライマリ・データベースまたはスタンバイ・データベースをオフラインにする前に、オブザーバをオフラインにして、ファスト・スタート・フェイルオーバーを無効にします。

災害が発生した場合にシャードをリカバリするための具体的なステップについては、Oracleサポートにお問合せください。

関連項目:

MAAベスト・プラクティス・ホワイト・ペーパーについては、Oracle Maximum Availabilityアーキテクチャを参照してください

シャード・データベース・スキーマの変更

シャード・データベースの重複表またはシャード表を変更する場合、それらの変更はシャード・カタログ・データベースから行う必要があります。

シャード・データベースに対してDDL操作を実行する前に、次のコマンドを使用してSHARD DDLを有効にします。

```
ALTER SESSION ENABLE SHARD DDL;
```

この文は、DDLによる変更がシャード・データベース内の各シャードに伝播されるようにします。

伝播されるDDLによる変更は、ALTER TABLEなどの操作が含まれる"スキーマ関連"として定義されているコマンドです。各シャードに伝播される他の操作には、ユーザー管理を簡略化するためのCREATE、ALTER、DROPユーザー・コマンド、複数のシャードでの表領域の作成を簡略化するためのTABLESPACE操作などがあります。

GRANTおよびREVOKE操作はシャード・カタログから行うことができ、セッションでSHARD DDLを有効にしている場合は各シャードに伝播されます。より細かい制御が必要な場合は、各シャードに対してコマンドを直接発行できます。

DBMSパッケージ・コール(または同様の操作)などの操作は伝播されません。たとえば、シャード・カタログで統計を収集する操作は各シャードに伝播されません。

表に対するロックを必要とする操作(NOT NULL列の追加など)を実行する場合、DDL操作を実行するには各シャードで表に対するロックを取得する必要があることに留意することが重要です。単一インスタンスにDDLを適用するためのOracleのベスト・プラクティスがシャード環境に適用されます。

シャード・カタログで実行されるマルチシャード問合せは、各シャードのデータベース接続でリモート問合せを発行します。この場合、問合せによってそのシャードからデータを返されるかどうかにかかわらず、ユーザーが各シャードに対する適切な権限を持つようにすることが重要です。

関連項目:

重複表およびシャード表に使用する操作の詳細は、Oracle Database SQL言語リファレンスを参照してください

シャード間のパラメータ設定の伝播

シャード・カタログでシステム・パラメータ設定を構成すると、シャード・データベースのすべてのシャードに自動的に伝播されます。

Oracle Database 19cより前は、シャード・データベースの各シャードでALTER SYSTEMパラメータ設定を構成する必要がありました。Oracle Database 19cでは、Oracle Shardingはシャード・カタログでパラメータを設定できるようにすることで集中管理を提供します。その後、設定はシャード・データベースのすべてのシャードに自動的に伝播されます。

システム・パラメータの伝播は、シャード・カタログに対してENABLE SHARD DDLで行われた場合にのみ発生します。その後、

ALTER文にSHARD=ALLを含めます。

```
SQL>alter session enable shard ddl;
SQL>alter system set enable_ddl_logging=true shard=all;
```

ノート:

enable_goldengate_replication パラメータ設定の伝播はサポートされていません。

非PDBシャードのPDBへの移行

単一インスタンスのレガシー・データベースからOracleマルチテナント・アーキテクチャにシャードを移行する場合は、次のステップを実行してください。

1. 既存の各非PDBシャードをバックアップして、新しいCDBと、その内部にPDBを作成します。
2. CDB内のPDBに、各シャードをリストアします。
3. GDSCTL ADD CDBコマンドを実行して、新しいCDBを追加します。

   ```
   GDSCTL> add cdb -connect cdb_connect_string -pwd gsmrootuser_password
   ```

4. GDSCTL ADD SHARD -REPLACEコマンドに、PDBの接続文字列shard_connect_stringを指定して実行します。これにより、シャーディング・インフラストラクチャにシャードの古い場所を新しいPDBの場所に置き換えるように指示します。

   システム管理またはコンポジットのシャーディングの場合は、次に示すパラメータでADD SHARDを実行します。

   ```
   GDSCTL> add shard -replace db_unique_name_of_non_PDB -connect shard_connect_string -pwd
   gsmuser_password
   -shardgroup shardgroup_name -cdb cdb_name
   ```

   ユーザー定義のシャーディングの場合は、わずかにコマンドの使用方法が異なります。

   ```
   GDSCTL> add shard -replace db_unique_name_of_non_PDB -connect shard_connect_string -pwd
   gsmuser_password
   -shardspace shardspace_name -deploy_as db_mode -cdb cdb_name
   ```

シャード・データベースのソフトウェア・バージョンの管理

この項では、シャード・データベース構成内のソフトウェア・コンポーネントのバージョン管理について説明します。次の項目が含まれます。

シャード・データベースへのパッチ適用およびアップグレード

シャード・データベース環境へのOracleパッチの適用は、単一のシャードまたはすべてのシャードに対して行うことができます。ただし、使用する方法は、その環境で使用されているレプリケーション・オプションおよび適用されるパッチのタイプによって異なります。

Oracle Shardingでは統合パッチ適用を使用してシャード・ディレクタ(GSM)のORACLE_HOMEを更新するため、Oracle Databaseリリース更新をORACLE_HOMEに適用してセキュリティおよびグローバル・データ・サービスの修正を取得する必要があります。

シャード・データベースへのパッチ適用

ほとんどのパッチは、単一のシャードに一度に適用できます。ただし、一部のパッチはすべてのシャードに適用する必要があります。シャード・データベースで使用されているレプリケーション方法に留意して、非シャード・データベースの場合と同様に、Oracleのベスト・プラクティスを使用して単一のシャードにパッチを適用します。Oracleのopatchautoは、複数のシャードに一度にパッチを適用するために使用でき、ローリング形式で行うことができます。Data Guard構成は1つずつ適用され、場合によっては(パッチによって異なります) Standby Firstパッチ適用を使用できます。

Oracle GoldenGateを使用する場合は、必ずシャード領域全体で並列にパッチを適用してください。パッチがマルチシャード問合せ、レプリケーションまたはシャーディング・インフラストラクチャの問題に対処するものである場合は、シャード・データベース内のすべてのシャードに適用する必要があります。

ノート:

Oracle ShardingのOracle GoldenGateレプリケーション・サポート高可用性は、Oracle Database 21cでは非推奨です。

シャード・データベースへのアップグレード

Oracle Sharding環境のアップグレードは、他のOracle Databaseとグローバル・サービス・マネージャの環境のアップグレードと大きな違いはありません。ただし、コンポーネントを特定の順序(最初にシャード・カタログ、次にシャード・ディレクタ、最後にシャードの順)でアップグレードする必要があります。

関連項目:

Oracle OPatchユーザーズ・ガイド

Oracle Database Global Data Services概要および管理ガイド(シャード・ディレクタのアップグレードの詳細)

Oracle Data Guard概要および管理(Oracle Data Guard構成内のパッチ適用およびアップグレードの詳細)

シャード・データベースのコンポーネントのアップグレード

コンポーネントが停止してオンライン状態に戻る際の停止時間を制限し、エラーを回避するためには、シャード・データベースのコンポーネントをアップグレードする順序が重要です。

シャード・データベースのコンポーネントをアップグレードするときは、次の点に注意する必要があります。

- 保留中のMOVE CHUNK操作が進行中の場合は、完了してください。
- 新しいMOVE CHUNK操作を開始しないでください。
- アップグレード・プロセス中に新しいシャードを追加しないでください。

1. 次の点に注意しながら、シャードをアップグレードします。
    - システム管理のシャード・データベースの場合: Data Guard Broker構成内のシャードの各セットをローリン

グ方式でアップグレードします。

- ユーザー定義のシャード・データベースの場合: シャード領域内のシャードの各セットをローリング方式でアップグレードします。
- コンポーネント・シャード・データベースの場合: 特定のシャード領域で、Data Guard Broker構成内のシャードの各セットをローリング方式でアップグレードします。
- プラガブル・データベース(PDB)シャードを含むOracle Database 18cシャード・データベース構成をアップグレードする場合は、Oracle Database 18cからの互換性と移行のPDB固有のアップグレード手順に従います。

2. シャード・カタログ・データベースをアップグレードします。

   最高の結果を得るには、ローリング・データベース・アップグレードを使用してカタログをアップグレードする必要があります(ただし、カタログが使用できない場合、アップグレード中にグローバル・サービスは引き続き使用可能ですが、サービス・フェイルオーバーは発生しません)。

3. GDSCTLクライアントを実行するために使用されるシャード・ディレクタをアップグレードします(グローバル・サービス・マネージャ・サーバーは実行しません)。

   シャード・ディレクタのアップグレードはインプレースで実行する必要がありますが、次のプラットフォームの各ファイルに対するアクセス権が755に更新されていない場合、インプレース・アップグレードによって間違ったエラー・メッセージが表示されます。

   - Linux、Solaris64およびSolaris Sparc64の場合:

     ```
     $ORACLE_HOME/QOpatch/qopiprep.bat
     $ORACLE_HOME/jdk/bin/jcontrol
     $ORACLE_HOME/jdk/jre/bin/jcontrol
     ```

   - AIXの場合:

     ```
     $ORACLE_HOME/QOpatch/qopiprep.bat
     $ORACLE_HOME/jdk/jre/bin/classic/libjvm.a
     $ORACLE_HOME/jdk/bin/policytool
     ```

   - HPIの場合:

     ```
     $ORACLE_HOME/jdk/jre/lib/IA64N/server/Xusage.txt
     $ORACLE_HOME/jdk/jre/bin/jcontrol
     $ORACLE_HOME/QOpatch/qopiprep.bat
     ```

   - Windowsでは、エラー・メッセージは予期されません。

4. すべてのシャード・ディレクタ・サーバーを一度に1つずつ停止、アップグレードおよび再起動します。

   停止時間をゼロにするには、1つ以上のシャード・ディレクタ・サーバーが常に稼働している必要があります。カタログより前のバージョンのシャード・ディレクタ・サーバーは、カタログに変更が発生するまで完全に動作し続けます。

関連項目:

DBMS_ROLLINGを使用してローリング・アップグレードを実行する方法の詳細は、Oracle Data Guard概要および管理を参照してください。

Oracle Data Guard概要および管理(Oracle Data Guard構成内のデータベースのパッチ適用およびアップグレードの詳細)

シャード・データベースのダウングレード

Oracle Shardingはダウングレードをサポートしていません。

シャード・データベース・カタログおよびシャードはダウングレードできません。

Oracle Database 18cからの互換性と移行

特定のCDBに単一のPDBシャードを含むOracle Database 18cインストールからアップグレードする場合、PDBのシャード・カタログ・メタデータを更新する必要があります。

具体的には、18cのPDBシャードの名前はCDBのDB_UNIQUE_NAMEですが、Oracle Database 19cのシャード名はdb_unique_name_of_CDB_pdb_nameです。

カタログ・メタデータを更新してこの新しいネーミング方法を反映し、GSMROOTUSERアカウントについての説明に従って新しいGSMROOTUSERアカウントもサポートするには、シャード・データベース・コンポーネントのアップグレードの説明に従って、アップグレード・プロセス中に次のステップを実行します。

1. PDBシャードを含むCDBをアップグレードした後、GSMROOTUSERアカウントが存在し、ロック解除され、パスワードが割り当てられ、SYSDG、SYSBACKUPおよびgsmrootuser_role権限が付与されていることを確認します。

   SQL*Plusの次のSQL文は、CDBのルート・コンテナ(CDB$ROOT)への接続中にGSMROOTUSERを正常に設定します。

   ```
   SQL> alter session set "_oracle_script"=true;
   Session altered.

   SQL> create user gsmrootuser;
   User created.
   SQL> alter user gsmrootuser identified by new_GSMROOTUSER_password
     account unlock;
   User altered.
   SQL> grant sysdg, sysbackup, gsmrootuser_role to gsmrootuser container=current;
   Grant succeeded.
   SQL> alter session set "_oracle_script"=false;
   Session altered.
   ```

2. カタログ・データベースを目的のOracle Databaseバージョンにアップグレードした後、次のPL/SQLプロシージャを実行して、構成に存在するPDBシャードの新しい名前を反映するようにカタログ・メタデータを更新します。

   このプロシージャは、Oracle Database 18cのPDBシャードごとに実行する必要があります。

   pdb_fixupの最初のパラメータは、PDBシャードを含むCDBのdb_unique_nameの値です。Oracle Database 18cでは、gdsctl config shardで示されるシャード名と同じです。

   2番目のパラメータはシャードPDBのPDB名で、シャードPDBへの接続時にSQL*Plusのshow con_nameによって示されます。

   pdb_fixupプロシージャは、カタログ・メタデータを更新して、PDBシャードの新しいネーミング・メソッドに対応させます。

   ```
   SQL> connect sys/password as sysdba
   ```

```
Connected.
SQL> set serveroutput on
SQL> execute gsmadmin_internal.dbms_gsm_pooladmin.pdb_fixup('cdb1', 'pdb1');
```

3. すべてのシャード・ディレクタを目的のバージョンにアップグレードした後、構成内の各CDBに対して次のGDSCTLコマンドを実行し、各CDBのGSMROOTUSERのパスワードをシャード・ディレクタに通知します。

```
GDSCTL> modify cdb -cdb CDB_name -pwd new_GSMROOTUSER_password
```

Enterprise Manager Cloud ControlによるOracleシャード・データベースの管理

Oracle Enterprise Manager Cloud Controlを使用すると、シャード・データベースとそのコンポーネントを検出、監視および管理できます。

Enterprise Manager Cloud Controlを使用したシャード・データベースの検出、監視および管理の詳細は、次のトピックを参照してください。

- 前提条件: シャード・データベース・メトリックの有効化
- 前提条件: シャードされたデータベース・トポロジの検出
- Enterprise Manager Cloud Controlによるシャード・データベースの監視
- Oracle Enterprise Manager Cloud Controlを使用したシャード・データベース管理の概要
- シャード管理
- チャンク管理
- シャード・ディレクタ管理
- リージョン管理
- シャード領域管理
- シャードグループ管理
- サービス管理

前提条件: シャード・データベース・メトリックの有効化

デフォルトでは、シャード・データベースのパフォーマンス・メトリックは無効になっています。これはEnterprise Manager Cloud コンソールまたはモニタリング・テンプレートから有効にできます。

メトリックを収集する方法は2つあります。これらの方法を実行するには、次の各項で説明するように、様々な設定ステップに従う必要があります。

デフォルトのEnterprise Managerデータベース・メトリックの使用

デフォルトでEnterprise Manager Cloudコンソールの「シャードされたデータベース」ページに表示されるメトリックはデフォルトのデータベース・メトリックであり、メトリック問合せユーザーを作成する必要があります。これは、Enterprise Managerで検出されたシャード・データベースでのみ収集されます。

デフォルトのデータベース・メトリックでは、後述する拡張シャード・データベース・メトリックほど、頻繁にデータが提供されません。

マルチシャード問合せはメトリックの収集に使用されるため、シャード・データベース内のすべてのシャードにアクセスして問合せを

実行できるユーザーも作成する必要があります。

デフォルト・メトリックを使用するには:

1. シャード間の問合せを実行する権限をユーザーに付与します。

```
alter session enable shard ddl;
```

2. すべてのシャードおよびシャード・カタログに新しいメトリック問合せアカウントを手動で作成します。

```
create user SHARD_SYS identified by password;
grant connect, create session, gsmadmin_role to SHARD_SYS;
GRANT ALL PRIVILEGES TO SHARD_SYS; /*Needed to get all the schemas stats*/
GRANT SELECT ANY DICTIONARY TO SHARD_SYS; /*Needed to get all the schemas stats*/
```

3. 同じメトリック問合せアカウント資格証明を使用して、Enterprise Managerのシャード・カタログおよびすべてのシャード・データベースを検出します。

   「前提条件: シャードされたデータベース・トポロジの検出」を参照してください

4. デフォルト・メトリックを有効にするには:

```
$emctl set property
 -sysman_pwd password
 -name oracle.sysman.db.ha.sdb.dd.usesdbmetrics
 -value false
```

拡張シャード・データベース・メトリックの使用

シャード・データベースの拡張メトリックを使用すると、シャードに関する情報をシャード・カタログから収集できるため、Enterprise Manager内のすべてのシャード・データベースを検出して、シャード・データベース・トポロジの完全なメトリックを取得する必要はありません。

拡張メトリックを使用するには:

1. Enterprise Managerでシャード・カタログを検出します。

   「前提条件: シャードされたデータベース・トポロジの検出」を参照してください

2. コンソールまたはモニタリング・テンプレートを使用して、シャード・データベース・メトリックを有効にします。

```
$emctl set property
 -sysman_pwd password
 -name oracle.sysman.db.ha.sdb.dd.usesdbmetrics
 -value true
```

シャード・データベースのコンポーネントの検出

Enterprise Manager Cloud Controlでは、シャード・カタログおよびシャード・データベースを検出し、ガイドされた検出を使用してシャード・ディレクタ、シャード・データベース、シャード領域およびシャードグループを追加できます。

Cloud Controlでシャード・データベースを管理する前提条件として、まずシャード・ディレクタ・ホストおよびシャード・カタログ・データベースを少なくとも検出する必要があります。必要に応じて、シャード・データベース内のすべてのシャードを管理するために、シャード・データベースも検出する必要があります。

シャード・カタログ・データベースおよび各シャードはデータベースそのものであるため、標準のデータベース検出プロシージャを使用できます。

シャードの管理は、データベース検出を使用して個々のシャードを検出する場合のみ可能です。シャードの検出は、シャードなしのシャード・データベース構成にすることもできるため、シャード・データベースの検出ではオプションです。

1. Enterprise Manager Cloud Controlで「設定」、「ターゲットの追加」、「ターゲットの手動追加」の順に選択します。
2. 「ターゲットの手動追加」ページで、「ガイド付きプロセスを使用して非ホスト・ターゲットを追加」パネルで「ガイド付きプロセスを使用した追加」をクリックします。
3. 「ガイド付きプロセスを使用した追加」ダイアログで、「シャードされたデータベース」を探して選択し、「追加」をクリックします。
4. 「シャードされたデータベースの追加: カタログ・データベース」ページで、「カタログ・データベース」の横にある参照アイコンをクリックしてシャード・カタログ・データベースを探します。
5. 「ターゲットの選択」ダイアログで、カタログ・データベースに対応するターゲット名をクリックし、「選択」をクリックします。

   「カタログ・データベース」フィールドと「モニタリング資格証明」フィールドは、存在する場合、自動入力されます。モニタリング資格証明は、シャード・カタログ・データベースに問い合せて構成情報を取得する場合に使用されます。

   モニタリング・ユーザー(通常はDBSNMP)は、GDS_CATALOG_SELECTロールが付与され、シャード・カタログ・リポジトリ表に対して読取り専用権限があります。

   ```
   SQL> grant GDS_CATALOG_SELECT to dbsnmp;
   ```

   「次へ」をクリックして次のステップに進みます。

   「シャードされたデータベースの追加: コンポーネント」ページには、シャード・データベース名、ドメイン名、シャード・データベースで採用されているシャーディング方法、検出されたシャード・ディレクタのリストなど、カタログ・データベースで管理されるシャード・データベースに関する情報が表示されます。

6. モニタリング資格証明をシャード・ディレクタに設定するには、リスト・エントリの右側にあるプラス記号アイコンをクリックします。

   ダイアログが開き、資格証明を設定できます。

   「OK」をクリックしてダイアログを閉じ、「次」をクリックして次のステップに進みます。

7. 「シャードされたデータベースの追加: 確認」ページで、シャード・ディレクタ、シャード領域およびシャードグループがすべて検出されたことを確認します。
8. 「発行」をクリックして、ステップを終了します。

   Enterprise Managerデプロイメント・プロシージャが発行され、「ターゲットの手動追加」ページに戻ります。

   ページ上部に、検出されたすべてのコンポーネントをCloud Controlに追加するために発行されたスクリプトに関する情報が表示されます。

9. リンクをクリックして、シャード・データベースのコンポーネントのプロビジョニング・ステータスを表示します。

   別のブラウザ・ウィンドウで、Cloud Controlの「すべてのターゲット」ページに移動してシャード・データベースのステータスを監視できます。

ターゲット検出プロシージャが終了すると、シャード・データベース・ターゲットがCloud Controlに追加されます。Cloud Controlでシャード・データベースを開き、コンポーネントを監視および管理できます。

Oracle Enterprise Manager Cloud Controlによるシャード・データベースの管理の概要

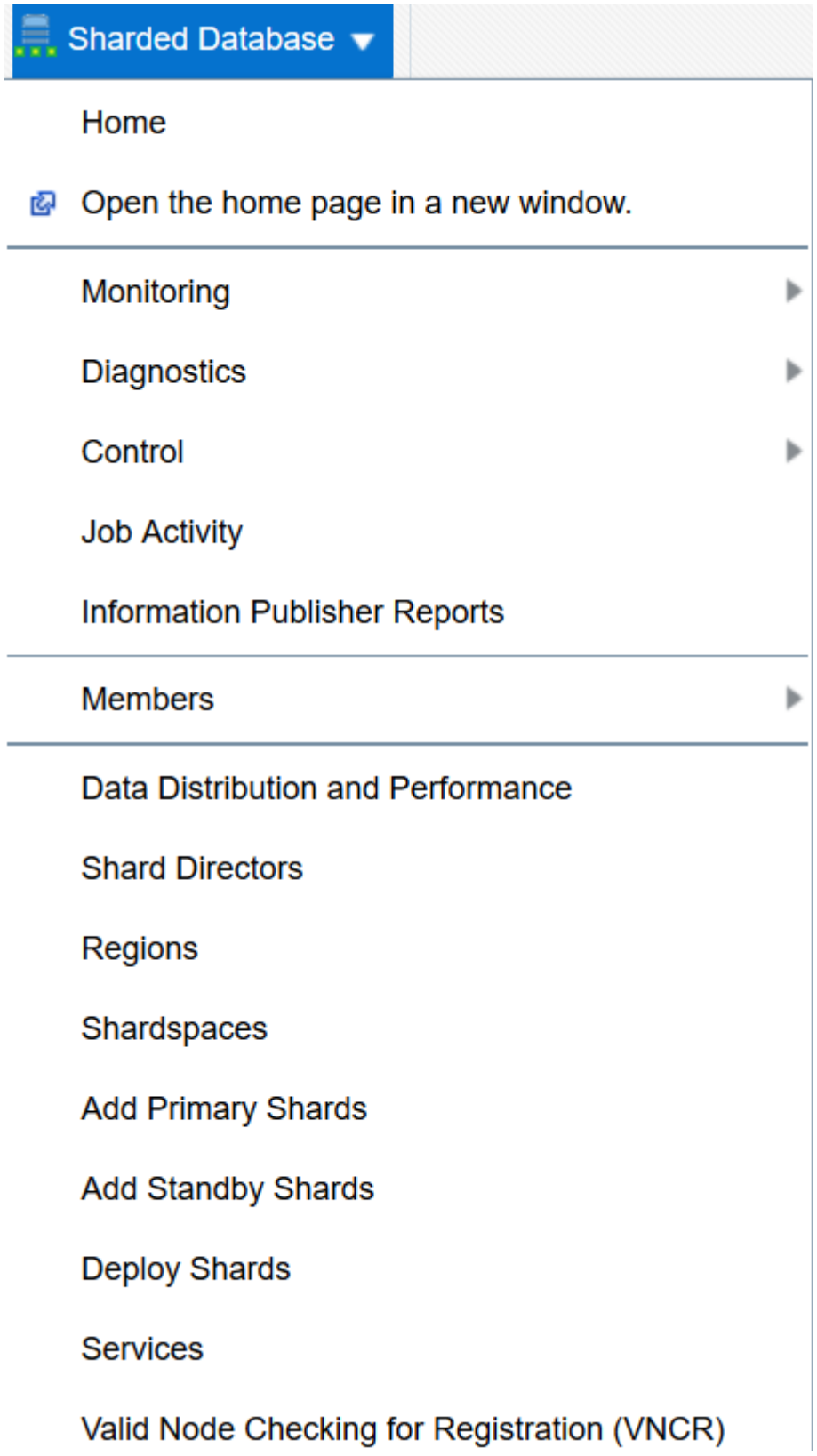
シャード・データベースは、Oracle Enterprise Manager Cloud Controlを使用して構成、デプロイ、監視および管理できます

検出されたシャード・データベース・オブジェクトは、Enterprise Managerの「すべてのターゲット」ページで確認できます。

次に示すのは、「データベース」ターゲット・タイプ・カテゴリのOracle Shardingオブジェクトのシャード・ディレクタおよびシャード・データベースです。

次に、「グループ」、「システムとサービス」ターゲット・タイプ・カテゴリのOracle Shardingオブジェクトのシャードグループおよびシャード領域を示します。

「シャードされたデータベース」ページでは、次に示すように、「プライマリ・シャードの追加」、「スタンバイ・シャードの追加」および「シャードのデプロイ」など、「シャードされたデータベース」メニューからほとんどの管理ツールにアクセスできます。



他のシャード・データベース・オブジェクトの管理ツールは、他のシャード・データベース・オブジェクト・ページのメニューにあります。このメニューは、これらのページにアクセスする必要がある手順で説明されています。

シャード・データベースの監視

Enterprise Manager Cloud ControlまたはGDSCTLを使用して、シャード・データベースを監視できます。

Enterprise Manager Cloud ControlまたはGDSCTLを使用してシャード・データベースを監視するには、次の各トピックを参照してください。

シャード間のシステム・オブジェクトの問合せ

SHARDS()句を使用して、V$ビューとDBA_*ビューからパフォーマンス、診断および監査データを収集するためにOracle提供の表を問い合せることができます。

SQLのSHARDS () 句を使用して、シャード・カタログ・データベースを一元化された診断操作のエントリ・ポイントとして使用できます。SHARDS () 句を使用して、すべてのシャードに対して同じOracle提供オブジェクト(V$、DBA/USER/ALLビュー、ディクショナリ・オブジェクトおよび表など)を問い合せ、集計結果を返すことができます。

次の例に示すように、SELECT文のFROM部分のオブジェクトをSHARDS () 句でラップすることにより、これがローカル・オブジェクトに対する問合せではなく、シャード・データベース構成内のすべてのシャード上のオブジェクトに対する問合せであることを指定します。結果に含まれるすべての行のソースを示すため、マルチシャード問合せの実行時にSHARD_IDという名前の仮想列がSHARDS () でラップされたオブジェクトに自動的に追加されます。同じ列を、問合せをプルーニングするための述語で使用できます。

SHARDS () 句を含む問合せは、シャード・カタログ・データベースに対してのみ実行できます。

例

次の文は、パフォーマンス・ビューを問い合せます。

```
SQL> SELECT shard_id, callspersec FROM SHARDS(v$servicemetric)
 WHEREservice_name LIKE 'oltp%' AND group_id = 10;
```

次の文は、統計情報を収集します。

```
SQL> SELECT table_name, partition_name, blocks, num_rows
 FROM SHARDS(dba_tab_partition) p
 WHERE p.table_owner= :1;
```

次の例の文は、各シャードのSHARD_ID値を見つける方法を示しています。

```
SQL> select ORA_SHARD_ID, INSTANCE_NAME from SHARDS(sys.v_$instance);
    ORA_SHARD_ID INSTANCE_NAME
    ------------ ----------------
               1 sh1
              11 sh2
              21 sh3
              31 sh4
```

次の例の文は、SHARD_IDを使用して問合せをプルーニングする方法を示しています。

```
SQL> select ORA_SHARD_ID, INSTANCE_NAME
 from SHARDS(sys.v_$instance)
 where ORA_SHARD_ID=21;
    ORA_SHARD_ID INSTANCE_NAME
    ------------ ----------------
              21 sh3
```

関連項目:

Oracle Database SQL言語リファレンス(SHARDS () 句の詳細)

GDSCTLによるシャード・データベースの監視

個別のシャード、シャードグループ、シャード領域およびシャード・ディレクタのヘルス・ステータスを取得するために使用できる多数のGDSCTL CONFIGコマンドがあります。

シャードの監視は通常のデータベースの監視と同様であり、標準のOracleのベスト・プラクティスを使用して、単一のシャードの個別の状態を監視してください。ただし、シャード環境全体の状態を監視することも重要です。GDSCTLコマンドは、スクリプトに記述し、スケジューラを使用して定期的に実行することによって、すべてが円滑に実行されていることを確認することもできます。レプリケーションにOracle GoldenGateを使用している場合は、各レプリケーション・ストリームのラグを監視することも重要です。

関連項目:

GDSCTL CONFIGコマンドの使用方法の詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください

Enterprise Manager Cloud Controlによるシャード・データベースの監視

シャード・データベース・ターゲットは、Enterprise Manager Cloud Controlの「すべてのターゲット」ページにあります。

シャード・データベースのコンポーネントを監視するには、まず統計収集を有効にしてから、シャード・データベースを検出する必要があります。詳細は、「前提条件: シャード・データベース・メトリックの有効化」および「シャード・データベース・コンポーネントの検出」を参照してください。

シャード・データベースのホーム・ページ

シャード・データベースのターゲット・ホーム・ページには、シャード・データベースのコンポーネントとそのステータスの概要が表示されます。

サマリー

ページの左上にある「サマリー」ペインには、次の情報が表示されます。

- シャードされたデータベース名: シャードされたデータベース名
- シャードされたデータベースのドメイン名: シャード・データベース・ドメイン名
- カタログ・データベース: シャード・カタログ・データベース名。名前をクリックすると、シャード・カタログ・データベースの詳細を表示できます。
- カタログのバージョン: シャード・カタログのOracle Databaseバージョン
- シャーディング・タイプ: データベースのシャードに使用されるシャーディング方法。これは、システム管理、ユーザー定義またはコンポジットのいずれかです。
- レプリケーション・タイプ: 高可用性に使用されるレプリケーション・テクノロジ。

- シャード・ディレクタ: シャード・ディレクタの数とステータス
- マスター・シャード・ディレクタ: マスター・シャード・ディレクタ名。シャード・ディレクタ名をクリックすると、シャード・ディレクタ(グローバル・サービス・マネージャ)のバージョン、現在のステータス、使用されているポートおよびインシデントなど、マスター・シャード・ディレクタに関する詳細情報を表示できます。

メンバー

ページの右上にある「メンバー」ペインには、シャードされた各データベース・コンポーネントの関連情報の一部が表示されます。

このペインは、各コンポーネントのタブ(シャード領域、シャードグループ、シャード・ディレクタ、シャード、カタログ・データベース、グローバル・サービス)に分かれています。各種コンポーネントの情報を表示するには、タブをクリックします。

- シャード領域:

  シャード領域は、ユーザー定義またはコンポジット・シャーディング方法でシャードされたデータベースに対してのみ表示されます。

  「シャード領域」タブには、シャード領域名、ステータス、チャンク数および保護モードが表示されます。シャード領域名をクリックすると、選択したシャード領域の詳細を表示できます。

  シャード領域名をクリックすると、シャード領域内のシャードグループに関する情報(コンポジット・シャーディング用)およびインシデントなどの詳細を表示できます。

- シャードグループ:

  シャードグループは、システム管理またはコンポジット・シャーディング方法でシャードされたデータベースに対してのみ表示されます。

  「シャードグループ」タブには、シャードグループ名、ステータス、所属するシャード領域、チャンク数、Data Guardロールおよび所属するリージョンが表示されます。

  シャードグループ名をクリックすると、シャードグループ内のシャードに関する情報やインシデントなど、選択したコンポーネントに関する詳細を表示できます。

  システム管理のシャーディング方法を使用してシャードされたデータベースの場合、shardspaceoraは、すべてのシャードグループを含めるためにOracle Shardingによって作成されるシャード領域です。これはOracle Shardingによって管理され、「シャード領域」タブに表示されません。

- シャード・ディレクタ:

  「シャード・ディレクタ」タブには、シャード・ディレクタ名、ステータス、リージョン、ホストおよびOracleホームが表示されます。

  シャード・ディレクタ名をクリックすると、シャード・ディレクタ(グローバル・サービス・マネージャ)のバージョン、現在のステータス、使用されているポートおよびインシデントなど、選択したシャード・ディレクタの詳細を表示できます。

  シャード・ディレクタ・ホストをクリックして、ホスト・システムの詳細を表示することもできます。

- シャード:

  「シャード」タブには、シャード名、Data Guardロール、ターゲット・タイプ、ターゲット・ステータス、所属するシャード領域およびシャードグループ、所属するリージョンおよび状態(デプロイ)が表示されます。

  「名前」列で、プライマリ・シャードを展開して、対応するスタンバイ・シャードの情報を表示できます。

  「デプロイ済」列のアイコンの上にマウスを置くと、デプロイメント・ステータスの詳細を表示できます。シャード名、シャード

領域名およびシャードグループ名をクリックすると、選択したコンポーネントの詳細を表示できます。

- カタログ・データベース

  「カタログ・データベース」タブには、シャード・カタログ・データベースがリストされ、各カタログ・データベースのシャード・カタログ・データベース名、タイプ、ステータスおよびロールが表示されます。

  カタログ・データベース名をクリックすると、データベースの詳細を表示できます。

- グローバル・サービス:

  「グローバル・サービス」タブには、シャード・データベース・グローバル・サービスの名前、ステータスおよびData Guardロールが表示されます。リストの上にはサービスの合計数と、特定のステータスにあるサービスの数を示すアイコンが表示されます。アイコンの上にマウス・ポインタを置くと、ステータス・アイコンの説明を表示できます。

インシデント

「インシデント」ペインには、シャード・データベース環境の様々なコンポーネントに関するメッセージおよび警告が表示されます。このペインの使用方法の詳細は、Cloud Controlオンライン・ヘルプを参照してください。

「シャードされたデータベース」メニュー

左上隅にある「シャードされたデータベース」メニューを使用すると、シャード・データベースのコンポーネントを管理するためのツールにアクセスできます。

ターゲット・ナビゲーション

「ターゲット・ナビゲーション」ペインを使用すると、シャード・データベースのどのコンポーネントの詳細にも簡単にアクセスできるようになります。

「シャードされたデータベース」ホーム・ページの左上隅にあるナビゲーション・ツリー・アイコンをクリックすると、「ターゲット・ナビゲーション」ペインが開きます。このペインには、シャード・データベースで検出されたすべてのコンポーネントがツリー形式で表示されます。

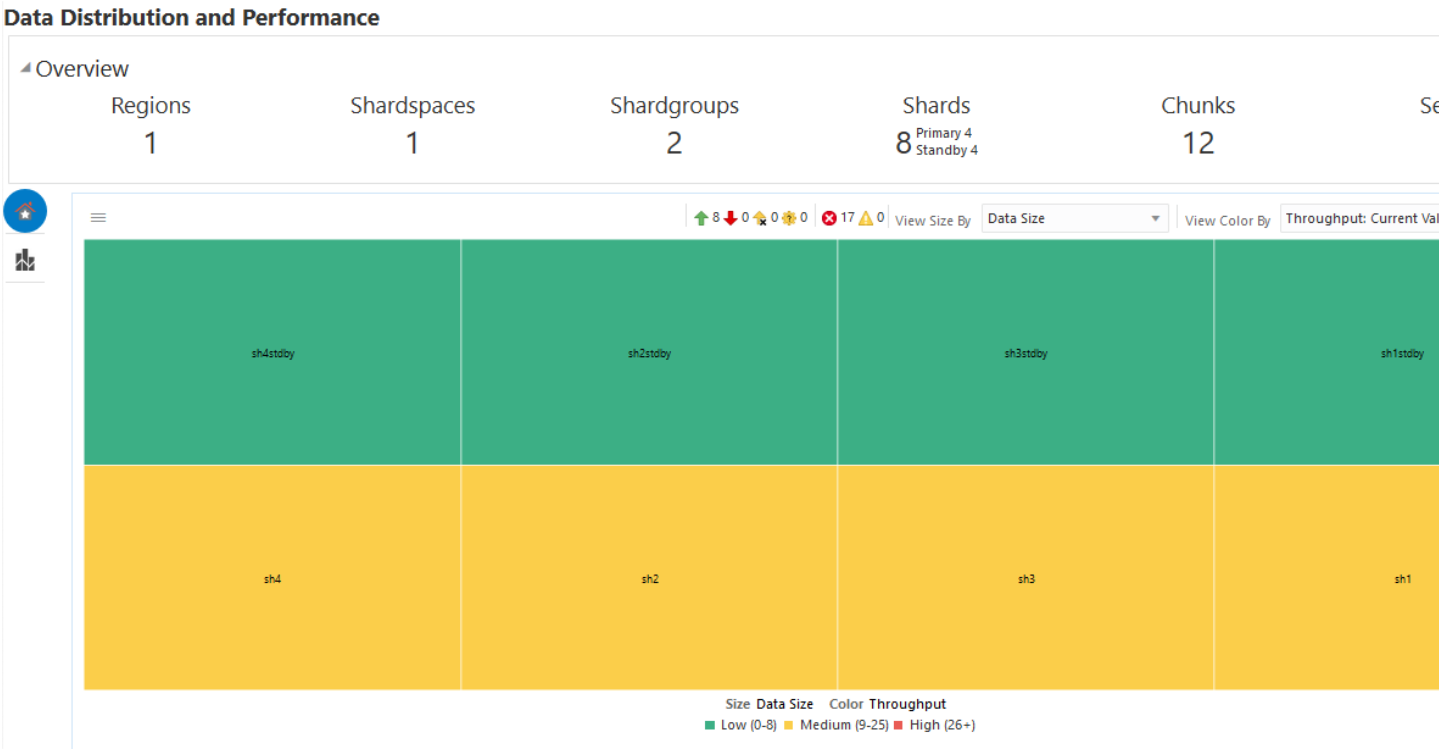
シャード領域を展開すると、その領域内のシャードグループが表示されます。シャードグループを展開すると、そのシャードグループ内のシャードが表示されます。

コンポーネント名をクリックすると、その詳細を表示できます。

「データ分散およびパフォーマンス」ページ

Enterprise Manager Cloud Controlにある「シャードされたデータベース」ページの「データ分散およびパフォーマンス」では、シャード・データベースのデータの包括的ビューと、シャードの実行方法が表示されます。

概要

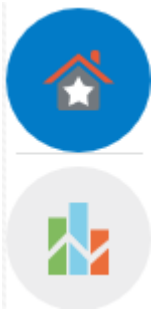


ページ上部の「概要」セクションには、チャートのデータで表されるシャード・データベース構成のリージョン、シャード領域、シャードグループ、シャード(プライマリおよびスタンバイに分かれる)、チャンクおよびサービスの数が表示されます。チャートにフィルタを適用すると、これらの数値が変わります。

データ分散およびパフォーマンスのチャート・ビュー

チャートの左上隅にある2つのアイコンは、チャートを2つのビュー間で切り替えます。

図9-1 「ホーム」アイコンおよび上位シャード・アイコン

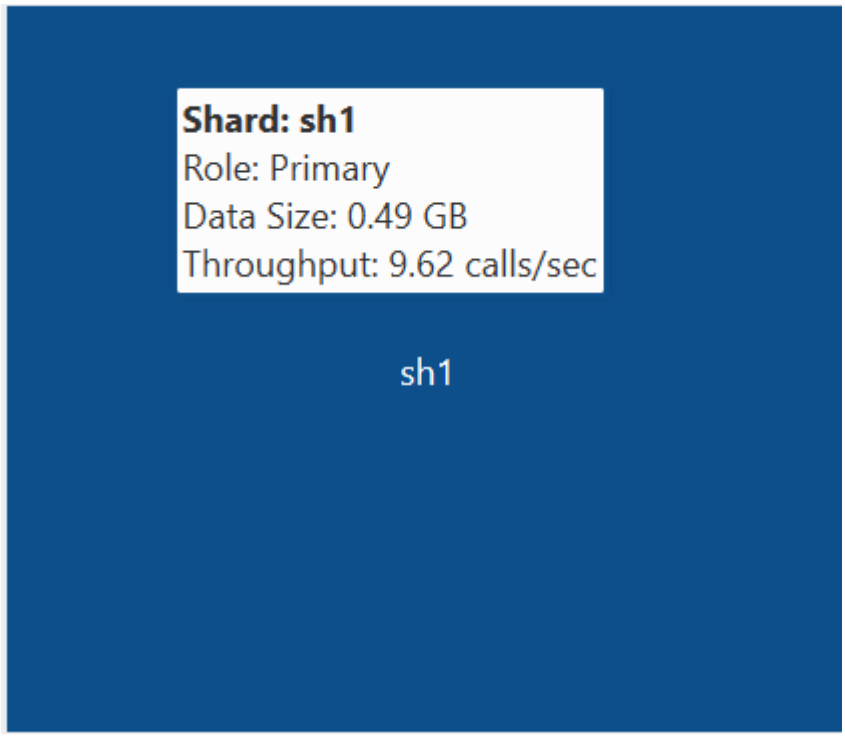


- ホーム: デフォルト・ビューです。「ホーム」には、デフォルトでシャード・データベース内のすべてのシャードのデータが表示されます。次に示すように、チャートをフィルタしたり、表示上のメトリックを変更できます。
- 上位シャード: 特定のメトリックの上位5、10または20のシャードのチャートを示します。

シャード・ブロック

色分けされたチャートには、データがシャード別に表示されます。各シャードはブロックによって示されます。

図9-2 マウス・オーバー・テキストを使用したシャード・ブロック



各ブロックには、シャード名のラベルが付けられます。ブロック上でマウスを移動すると、シャード名、Data Guardロール、シャード内のチャンク数およびサービス時間(ミリ秒/コール)が表示されます。

ノート:

デフォルトのデータベース・メトリックを使用している場合、チャートに未検出のシャードのデータは表示されません。

拡張メトリックを使用している場合は、シャードがシャード・カタログで検出されるため、すべてのシャードのデータが表示されます。

ホーム・ビューのサマリー・アイコン

チャートの上にあるアイコンの行には、次の情報が表示されます。

図9-3 ホーム・ビューのサマリー・アイコン

- 稼働中: (上向きの緑色の矢印)稼働しているシャード・データベース数
- 停止中: (下向きの赤い矢印)停止しているシャード数
- 未モニター: (「X」が付いた黄色の矢印)監視されていないシャードの数。これは、Enterprise Managerで検出されなかったシャードの数です。
- その他: (疑問符「?」が付いた黄色のギア) Enterprise Managerで検出されたが、到達不能なエージェントや可

用性評価エラーなど、ターゲット監視に問題があるシャード・データベース・ターゲット。

- クリティカル: (「X」が付いた赤い円)クリティカル・インシデントの数
- 警告: (感嘆符「!」が付いた黄色の三角形)警告インシデント数

チャート・ビュー・コントロール

各シャードのメトリックを、チャート内のブロックのサイズと色で比較します。

図9-4 チャート・ビュー・コントロール

- ビュー・サイズの基準: 選択したメトリックでブロックのサイズ分散を変更します
- ビューの色の基準: 選択したメトリックでブロックの比較色を変更します

  デフォルトでは、色は明るい青、中位の明るさの青色および濃い青色で、最も明るい色と最も暗い色のカテゴリのしきい値が任意のEnterprise Managerのデフォルトに設定されます。

  「しきい値の構成」(3つのドットを含むボタン)をクリックして、各メトリックの低いカテゴリと高いカテゴリのカスタムしきい値を設定します。カスタムしきい値で構成されたチャートは、緑色=低、黄色=中、赤=高の各色のスペクトルで表示されます。

- ツリー・マップ表ビュー: (チャートの右上隅にある表のボタン)チャートに表示されるデータの表ビューを表示します

フィルタ

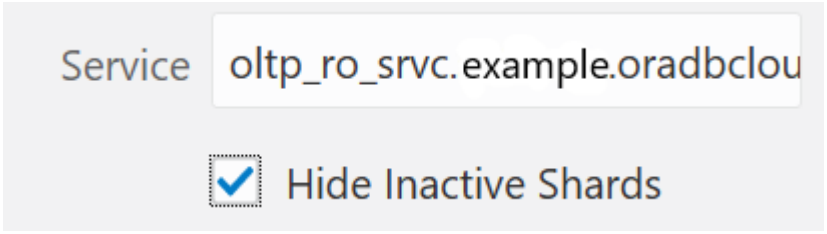
チャートの左上隅にあるハンバーガ・アイコンをクリックして、データにフィルタを適用します。

図9-5 「フィルタ」アイコン

- シャード検索: シャード名でフィルタします。アスタリスク(*)を使用して、一致する名前パターンを持つシャードのグループを選択できます。
- キー検索: シャード・キーの値を入力して、そのキーのデータを含むシャードを表示できます。表示されたチャートで、ブロックを右クリックしてシャード・レベル・データ分散を選択し、特定のシャードにドリルダウンできます。
- SQL ID検索: 問合せのSQL IDによって問合せを実行しているシャードを表示します。これは、カタログ・データベースのV$SQL_SHARDビューにあります。
- ソート基準: デフォルトのタイル・ビューでチャート内のブロックをサイズでソートしたり、棒の順序でソートしたり、上位または下位の5ブロックのみを表示します。
- フィルタ条件: 指定したロール、シャードグループまたはサービスのシャードのみを表示できます。

  非アクティブなシャードを非表示: サービス・フィルタを使用すると、すべてのシャードが表示されますが、サービスが実行

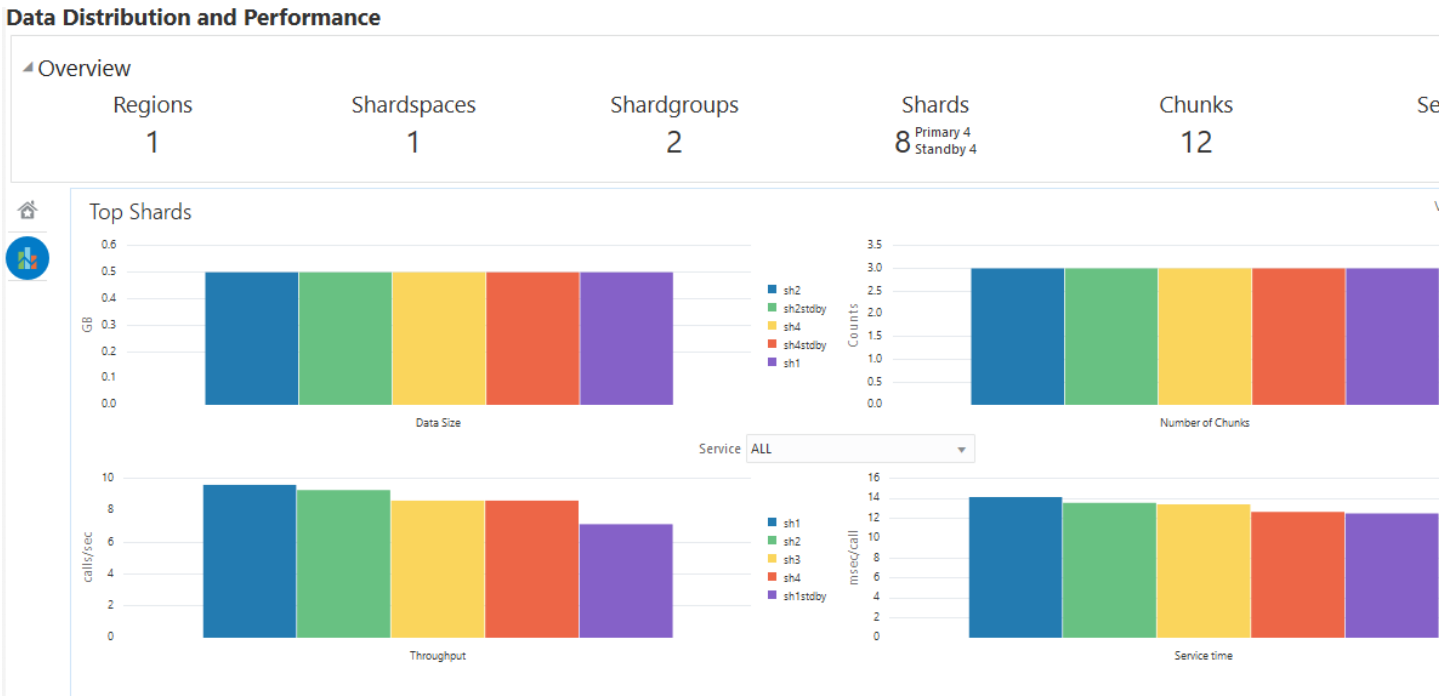
されていないシャードはグレー(非アクティブ)で表示されます。このチェック・ボックスを使用して非アクティブなシャードを非表示にできます。



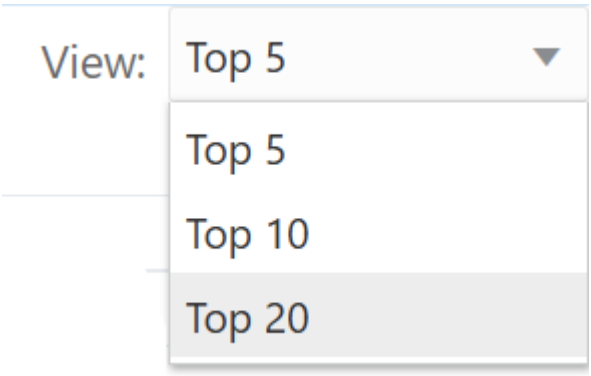
- グループ化: グループの集計の表示を切り替えます。これは、シャードグループの周りにボックス行で示されます。
  - 「シャードグループ」では、グループ上部にシャードグループ・ボックスが表示され、ホバー時にシャードグループに関する集計情報が表示されます。シャードグループ・ベースのデータにドリルダウンできます。
  - 「リージョン」では、グループの上部にリージョン・ボックスが表示され、ホバー時にリージョンに関する集計情報が表示されます。
  - Data Guard集計グループでは、各シャードとそのスタンバイが単一のエンティティとしてグループ化されるため、特定のシャードとそのスタンバイが全体として処理するデータ・セットを確認できます。

上位シャード・ビュー

チャートの左側にある上位シャード・ボタンをクリックして、データ・サイズ、チャンク数、スループットおよびサービス時間が最も大きいシャードのメトリックを含むグラフを表示します。



ビューの右上隅にある「ビュー」リストを使用して、各グラフで上位5、10または20個のシャードを表示します。



シャード管理

Oracle Enterprise Manager Cloud ControlおよびGDSCTLを使用したOracle Shardingデプロイメントのシャードを管理できます。

次のトピックでは、シャード管理の概念およびタスクについて説明します。

シャードの追加について

既存のシャード・データベース環境に新しいシャードを追加することによって、スケール・アウトしたり、フォルト・トレランスを向上させたりできます。

フォルト・トレランスのためには、少数の非常に大きいシャードを使用するより、多数の小さいシャードを使用するほうが効果的です。アプリケーションが長期間使用されてデータ量が増えた場合、1つ以上のシャードをSDBに追加して容量を増やすことができます。

シャード・データベースにシャードを追加するときに、環境がコンシステント・ハッシュによってシャーディングされている場合は、既存のシャードからのチャンクが新しいシャードに自動的に移動されて、シャード環境がリバランスされます。

ユーザー定義シャーディングを使用する場合、新しいシャードにデータを移入するには、GDSCTLのsplit chunkおよびmove chunkコマンドを使用して、既存のシャードから新しいシャードにチャンクを手動で移動する必要がある場合があります。

Oracle Enterprise Manager Cloud Controlを使用して、新しいシャードに移動するか、分割して移動するよい候補となるチャンクを識別できます。

環境にシャードを追加する場合は、スタンバイ・サーバーが準備完了状態であることを確認し、新しいシャードが配置された後に、move chunk操作に関係したすべてのシャードのバックアップを取得します。

再シャーディングおよびホット・スポット回避

シャード数の変化によってトリガーされてデータがシャード間に再分散されるプロセスを再シャーディングと呼びます。自動再シャーディングは、システム管理のシャーディング方法の機能であり、SDBに柔軟なスケーラビリティを提供します。

ときには、SDBのデータを1つのシャードから他のシャードに移行する必要があります。シャード間のデータ移行は次のような場合に必要です。

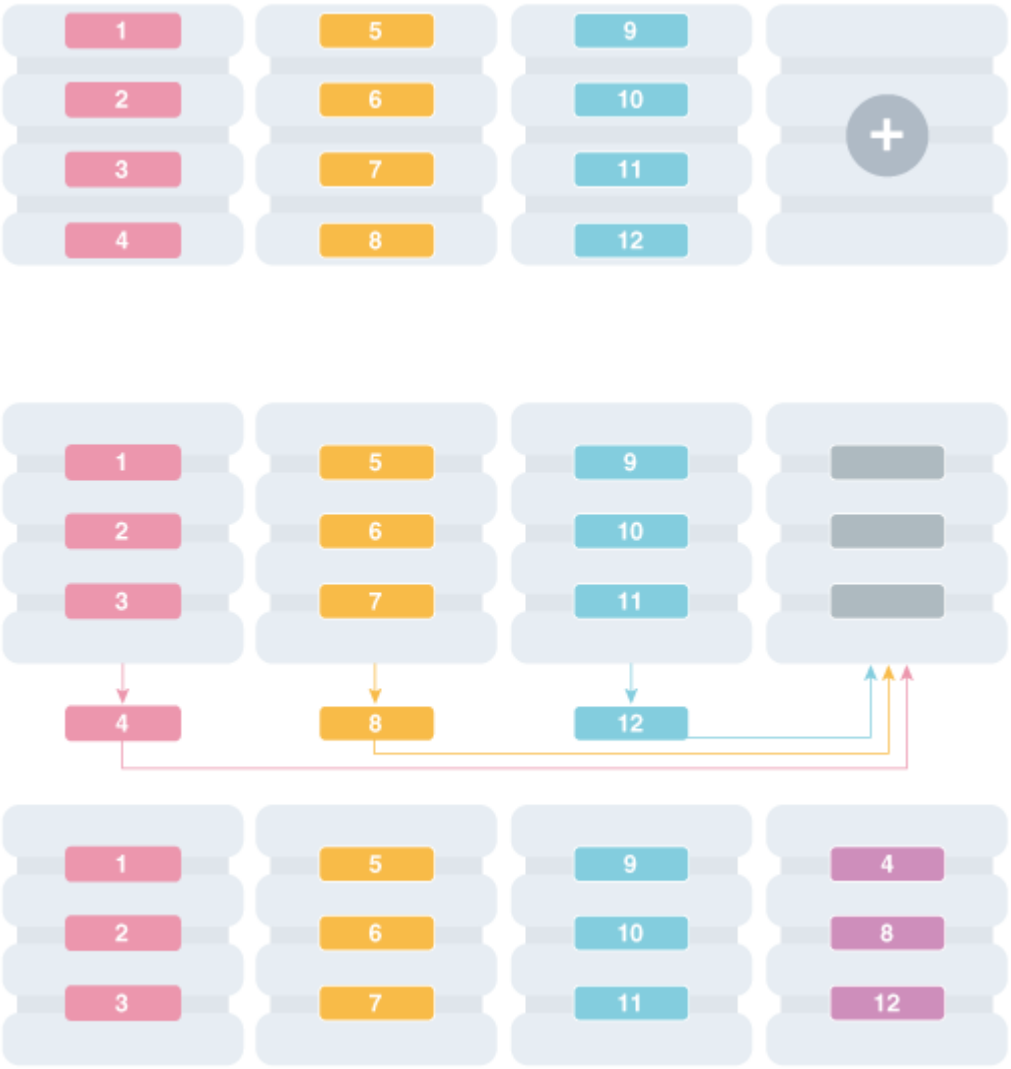
- SDBに対して1つ以上のシャードを追加または削除した場合
- シャード間のデータ分散またはワークロード分散にスキューがある場合

シャード間のデータ移行の単位はチャンクです。チャンクでデータを移行することにより、異なるシャード・データベースの関連するデータが確実に一緒に移動されます。

SDBに対してシャードを追加または削除すると、複数のチャンクが移行されて、シャード間のチャンクとワークロードのバランスのとれた分散が維持されます。

シャーディング方法に応じて、再シャーディングが自動的に行われるか(システム管理)、ユーザーが指示します(コンポジット)。次の図は、3つのシャードを含むSDBにシャードを追加したときの自動再シャーディングの段階を示しています。

図 9-6 SDBの再シャーディング



データまたはワークロードのスキューが発生したときに、シャード数を変更せずに、特定のチャンクを1つのシャードから他へ移動することもできます。この場合、ホット・スポットを回避するために、データベース管理者がチャンクの移動を開始できます。

RMAN増分バックアップ、トランスポータブル表領域およびOracle Notification Serviceテクノロジを使用して、アプリケーションの可用性へのチャンク移行の影響を最小限に抑えます。チャンク移行中、チャンクはオンラインのままです。短時間(数秒間)は、チャンクに格納されたデータが読取り専用アクセスのみになります。

FAN対応クライアントは、ソース・シャードでチャンクが読取り専用になる直前に通知を受信し、チャンク移行が完了して宛先

シャードでチャンクが完全に使用可能になると、再度、通知を受信します。chunk read-onlyイベントを受信すると、クライアントはチャンク移行が完了するまで接続の試行を繰り返すか、ソース・チャンクの読取り専用チャンクにアクセスできます。後者の場合、チャンクに書き込もうとすると、ランタイム・エラーが発生します。

ノート:

シャード・データベースの再シャーディング中にマルチシャード問合せを実行すると、エラーが発生する可能性があるため、マルチシャードのワークロード中に新しいシャードをデプロイしないことをお薦めします。

関連項目:

システム管理のSDBへのシャードの追加

シャーディング方法

プールからのシャードの削除

シャードにあるデータを失うことなく、シャード・データベース環境からシャードを一時的または永続的に削除することが必要となる場合があります。

たとえば、忙しい休日の後にシャード環境がスケール・ダウンされる場合、またはデータ・センター内のサーバーまたはインフラストラクチャを交換する場合、シャードの削除が必要となることがあります。シャードを削除する前に、すべてのチャンクをそのシャードからオンラインの状態に維持される他のシャードに移動する必要があります。それらを移動するときは、すべてのシャード間でデータおよびアクティビティのバランスが維持されるようにします。

シャードを一時的に削除するだけの場合は、保守が完了した後に簡単に識別して元に戻すことができるように、各シャードに移動したチャンクを追跡します。

関連項目:

チャンクの移動について

GDSCTL REMOVE SHARDコマンドの使用方法の詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください

スタンバイ・シャードの追加

Data Guardのスタンバイ・シャードをを Oracle Sharding環境に追加できます。ただし、制限があります。

シャード・データベースのレプリケーション方法としてData Guardを使用する場合、Oracle Shardingでは、プライマリまたは物理スタンバイ・シャードの追加のみサポートします。新しいスタンバイをシャード・データベースに追加する場合、他のタイプのData Guardスタンバイ・データベースはサポートされません。ただし、すでにシャード・データベースの一部であるシャードは、物理スタン

バイからスナップショット・スタンバイに変換できます。物理スタンバイからスナップショット・スタンバイに変換する場合、次のステップに従う必要があります。

1. GDSCTLコマンドSTOP SERVICEを使用して、シャードのすべてのグローバル・サービスを停止します。
2. GDSCTLコマンドDISABLE SERVICEを使用して、シャードのすべてのグローバル・サービスを無効化します。
3. Data Guardドキュメントで説明されている手順を使用して、シャードをスナップショット・スタンバイに変換します。

   この時点で、シャードはシャード・データベースの一部のままですが、シャーディング・キーを使用する接続を受け入れません。

データベースが物理スタンバイに逆変換される場合、グローバル・サービスを有効化して再開できます。シャードはシャード・データベースのアクティブなメンバーになります。

関連項目:

『Oracle Data Guard概要および管理』

Oracle Enterprise Manager Cloud Controlを使用したシャードの管理

Oracle Enterprise Manager Cloud Controlを使用してデータベース・シャードを管理できます

Cloud Controlを使用してシャードを管理するには、最初に検出する必要があります。各データベース・シャードはデータベース自体であるため、標準のCloud Controlデータベース検出プロシージャを使用できます。

次のトピックでは、Oracle Enterprise Manager Cloud Controlを使用したシャード管理について説明します。

シャードの検証

Oracle Shardingデプロイメントに追加する前に、シャードを検証します。

Oracle Shardingデプロイメントに追加する前に、Oracle Enterprise Manager Cloud Controlを使用してシャードを検証できます。デプロイメントした後にシャードを検証して、シャードのその後のライフサイクルでも設定が引き続き有効であることを確認することもできます。たとえば、ソフトウェアのアップグレード後に、既存のシャードを検証して、パラメータおよび構成が正しいことを確認できます。

Cloud Controlを使用してシャードを検証するには、Cloud Controlで監視している既存のターゲットである必要があります。

1. シャードグループ管理ページから、シャードグループ・ターゲット・ページの左上隅にある「シャードグループ」メニューを開き、「シャードの管理」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストからシャードを選択して、「検証」をクリックします。
4. 「OK」をクリックして、シャードを検証することを確認します。
5. ページ上部にある「情報」ボックスのリンクをクリックして、シャードのプロビジョニング・ステータスを表示します。

シャード検証スクリプトが正常に実行されると、出力にレポートされるエラーを確認します。

プライマリ・シャードの追加

Oracle Enterprise Manager Cloud Controlを使用して、プライマリ・シャードをOracle Shardingデプロイメントに追加します。

プライマリ・シャードは、Cloud Controlで監視している既存のターゲットである必要があります。

Oracle Sharding環境に追加する前にシャードを検証することを強くお薦めします。Cloud Controlを使用してシャードを検証するか(「シャードの検証」を参照)、またはSQL*Plusを使用してシャードに対してDBMS_GSM_FIX.validateShardプロシージャを実行します(「シャードの検証」を参照)。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「プライマリ・シャードの追加」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「シャードされたデータベースにすべてのシャードをデプロイ」を選択して、シャード・データベース構成に追加されたすべてのシャードをデプロイします。

   デプロイメント操作は、シャードの構成を検証し、最終構成ステップを実行します。シャードはデプロイ後にのみ使用できます。

4. 「追加」をクリックします。
5. 「シャードの詳細」ダイアログの「データベース」フィールドで、シャードを選択して、「選択」をクリックします。
6. コンポジットOracle Sharding環境で、シャードを追加するシャード領域を選択できます。
7. 「OK」をクリックします
8. 必要に応じてGSMUSER資格証明を入力し、「次」をクリックします。
9. ADD SHARD操作が発生する時間を示し、「次」をクリックします。
   - 即時: 確認後にシャードがプロビジョニングされます
   - 後で: 隣接しているフィールドのカレンダ・ツールを使用して、シャードを追加するタイミングをスケジュールします
10. 追加するシャードの構成を確認し、「送信」をクリックします。
11. ページ上部にある「情報」ボックスのリンクをクリックして、シャードのプロビジョニング・ステータスを表示します。

上記の手順の「シャードされたデータベースにすべてのシャードをデプロイ」を選択しなかった場合、シャードのデプロイ・タスクを使用して、Oracle Shardingデプロイメントにシャードをデプロイします。

スタンバイ・シャードの追加

Oracle Enterprise Manager Cloud Controlを使用して、スタンバイ・シャードをOracle Shardingデプロイメントに追加します。

スタンバイ・シャードは、Cloud Controlで監視している既存のターゲットである必要があります。

Oracle Sharding環境に追加する前にシャードを検証することを強くお薦めします。Cloud Controlを使用してシャードを検証するか(「シャードの検証」を参照)、またはSQL*Plusを使用してシャードに対してDBMS_GSM_FIX.validateShardプロシージャを実行します(「シャードの検証」を参照)。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「スタンバイ・シャードの追加」を選択します。

2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「シャードされたデータベースにすべてのシャードをデプロイ」を選択して、シャード・データベース構成に追加されたすべてのシャードをデプロイします。

   デプロイメント操作は、シャードの構成を検証し、最終構成ステップを実行します。シャードはデプロイ後にのみ使用できます。

4. 新しいシャードが「プライマリ・シャード」リストのスタンバイとして動作するプライマリ・シャードを選択します。
5. 「追加」をクリックします。
6. 「シャードの詳細」ダイアログの「データベース」フィールドで、スタンバイ・シャードを選択します。
7. シャードを追加するシャードグループを選択します。

   選択したプライマリのスタンバイがまだ含まれていないシャードグループのみ表示されます。

8. 「OK」をクリックします
9. 必要に応じてGSMUSER資格証明を入力し、「次」をクリックします。
10. ADD SHARD操作が発生する時間を示し、「次」をクリックします。
    - 即時: 確認後にシャードがプロビジョニングされます
    - 後で: 隣接しているフィールドのカレンダ・ツールを使用して、シャードを追加するタイミングをスケジュールします
11. 追加するシャードの構成を確認し、「送信」をクリックします。
12. ページ上部にある「情報」ボックスのリンクをクリックして、シャードのプロビジョニング・ステータスを表示します。

上記の手順の「シャードされたデータベースにすべてのシャードをデプロイ」を選択しなかった場合、シャードのデプロイ・タスクを使用して、Oracle Shardingデプロイメントにシャードをデプロイします。

シャードのデプロイ

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Sharding環境に追加されているシャードをデプロイします。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャードのデプロイ」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「リバランスの実行」チェック・ボックスを選択して、シャードのデプロイ後に自動的にシャード間のデータを再配分します。

   チャンクをシャードに手動で移動する場合、このボックスの選択を解除します。

4. 「送信」をクリックします
5. ページ上部にある「情報」ボックスのリンクをクリックして、シャードのプロビジョニング・ステータスを表示します。

GDSCTLを使用したシャードの管理

GDSCTLコマンドライン・ユーティリティを使用して、Oracle Shardingデプロイメントのシャードを管理できます。

次のトピックでは、GDSCTLを使用したシャードの管理について説明します。

シャードの検証

新しく作成したシャードをシャーディング構成に追加する前に、シャードがシャーディング環境用に正しく構成されていることを検証する必要があります。

ADD SHARDを実行する前に、シャードとして追加されるデータベースに対してvalidateShardプロシージャを実行します。validateShardプロシージャは、シャードとして正常に動作するために必要な初期化パラメータおよび特性がターゲット・データベースにあることを検証します。

validateShardプロシージャはターゲット・データベースを分析し、そのデータベースに対してGDSCTL ADD SHARDを実行する前に対処する必要がある問題を報告します。validateShardプロシージャはデータベースまたはパラメータを変更しません。情報および起こり得る問題を報告するだけです。

validateShardプロシージャは、シャードをシャード・カタログに追加するときに使用するレプリケーション・テクノロジとしてData GuardとOracle GoldenGateのどちらかを指定するオプションのパラメータを1つ取ります。Data Guardを使用する場合は、validateShard('DG')を呼び出します。Oracle GoldenGateを使用する場合は、validateShard('OGG')を使用します。validateShardにパラメータを渡さない場合のデフォルト値は、Data Guardです。

validateShardプロシージャは、シャードをデプロイメントした後に実行して、シャードのその後のライフサイクルでも設定が引き続き有効であることを確認することもできます。たとえば、ソフトウェアのアップグレード後またはシャードのデプロイメント後に、既存のシャードに対してvalidateShardを実行し、パラメータおよび構成が正しいことを確認できます。

次のようにvalidateShardを実行します。

```
sqlplus / as sysdba
SQL> set serveroutput on
SQL> execute dbms_gsm_fix.validateShard
```

出力例を次に示します。

```
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is DEN27B.
INFO: Database unique name is den27b.
INFO: Database ID is 718463507.
INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
INFO: Flashback is on.
INFO: Force logging is on.
INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is WE8DEC. This value must match the character set of
 the catalog database.
INFO: 'compatible' initialization parameter validated successfully.
INFO: Database is not a multitenant container database.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '<ORACLE_BASE>/oracle/dbs2'
INFO: db_recovery_file_dest set to: '<ORACLE_BASE>/oracle/dbs2'
INFO: db_files=1000. Must be greater than the number of chunks and/or tablespaces
 to be created in the shard.
INFO: dg_broker_start set to TRUE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '/dbs/dt, /dbs/bt, dbs2/DEN27D/, dbs2/DEN27B/'
```

```
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is '<ORACLE_BASE>//oracle/dbs2'.
```

INFOがタグ付けされている行は、基本的に情報であり正しい設定であることを示しています。WARNINGがタグ付けされている行は、構成によって問題である場合と問題ではない場合があります。たとえば、Data Guardパラメータに関する問題が報告されても、構成にプライマリ・データベースのみが含まれている場合は、Data Guardの問題を無視できます。最後に、そのシャードをシャーディング構成にデプロイして正常に稼働させるには、ERRORタグが付けられている出力を修正する必要があります。

システム管理のSDBへのシャードの追加

システム管理のSDBにシャードを追加すると、SDBを柔軟に拡張できます。システム管理のSDBのチャンクは、新しいシャードが追加された後に自動的にリバランスされます。

新しいシャードのホストを準備するには、最初にシャード・データベース環境を準備したときと同じすべてのセットアップ手順(次の手順を含む)を行います。

- Oracle Databaseソフトウェアのインストール

1. シャード・ディレクタのホストに接続して、環境変数を確認します。

```
$ ssh os_user@shard_director_home
$ env |grep ORA
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/18.0.0/gsmhome_1
```

2. 現在のセッションのグローバル・サービス・マネージャを設定し、それを管理するための資格証明を指定します。

```
$ gdsctl
GDSCTL> set gsm -gsm sharddirector1
GDSCTL> connect mysdbadmin/mysdbadmin_password
```

3. 現在のシャードの構成を確認します。

```
GDSCTL> config shard
Name            Shard Group          Status    State       Region    Availability
----            -----------          ------    -----       ------    ------------
sh1             primary_shardgroup   Ok        Deployed    region1   ONLINE
sh2             standby_shardgroup   Ok        Deployed    region2   READ_ONLY
sh3             primary_shardgroup   Ok        Deployed    region1   ONLINE
sh4             standby_shardgroup   Ok        Deployed    region2   READ_ONLY
```

4. 新しい各シャードのシャード・グループ、宛先および資格証明を指定します。

   この例では、新しいシャードはshard5およびshard6という名前で、NETCAおよびDBCAのデフォルト・テンプレートが使用されています。

```
GDSCTL> add invitednode shard5
GDSCTL> create shard -shardgroup primary_shardgroup -destination shard5
 -credential os_credential -sys_password
GDSCTL> add invitednode shard6
GDSCTL> create shard -shardgroup standby_shardgroup -destination shard6
 -credential os_credential -sys_password
```

   シャードを作成するときに、前の例に示すようにcreate shardで-sys_passwordを使用してSYSのパスワードを設定することもできます。これにより、シャードが作成されてDEPLOYされたときに、SYSのパスワードが設定されます。

前述の例では、新しいシャードを作成するためにCREATE SHARDによる方法を使用しています。ADD SHARDコマンドを使用して事前構成済のシャードを追加するには、ADD INVITEDNODEの後で次のコマンドを実行します。

```
GDSCTL> add shard -shardgroup primary_shardgroup
 -connect shard_host:TNS_listener_port/shard_database_name
 -pwd GSMUSER_password
```

追加されるシャードがPDBである場合は、ADD SHARDで-cdbオプションを使用して、PDBシャードが含まれているCDBを指定する必要があります。また、ADD SHARDコマンドの前にADD CDBを使用してカタログにCDBを追加する必要があります。ADD CDBおよびADD SHARDの構文は、*Oracle Database Global Data Services*概要および管理ガイドを参照してください。

ノート:

登録に関する有効ノード・チェック(VNCR)機能では、シャード・ディレクタによって許可された登録リクエストの IP アドレス、ホスト名またはサブネットのセットの構成や動的な更新が可能になります。シャード・ディレクタへのデータベース・インスタンスの登録は、リクエスト元が有効なノードである場合にのみ成功します。デフォルトでは、create shard または add shard が実行されるたびに、シャード管理層(Oracle Global Data Services フレームワークに基づく)は、リモート・データベースが実行されているホストの VNCR エントリを自動的に追加します。自動化(自動 VNCR と呼ばれます)によってターゲット・ホストのパブリック IP アドレスが検索され、その IP アドレスの VNCR エントリが自動的に追加されます。ホストに複数のパブリック IP アドレスがある場合は、データベースに登録されたアドレスが自動 VNCR を使用して追加されたアドレスと同じではないことがあり、登録が拒否されることがあります。ターゲット・データベースのホストに複数のパブリック IP アドレスがある場合は、GDSCTL で add invitednode コマンドまたは add invitedsubnet コマンドを使用して、このホストの VNCR を手動で構成することをお薦めします。

複数のネットカードがターゲット・ホストにある(/sbin/ifconfig で複数のパブリック・インタフェースが返されます)場合は、安全のために add invitednode を使用します(パケットをルーティングするために使用されるインタフェースを確認してから)。

登録になんらかの疑念がある場合は、必要に応じて config vncr および add invitednode を使用します。これを行っても問題はありません。ノードがすでに追加されている場合、自動 VNCR はそれを無視し、自動 VNCR によってすでに追加された後にノードを追加しようとすると、すでに存在しているという警告を受け取るからです。

5. DEPLOYコマンドを実行して、シャードとレプリカを作成します。

```
GDSCTL> deploy
```

6. 新しいシャードがデプロイされたことを確認します。

```
GDSCTL> config shard
Name         Shard Group         Status    State       Region    Availability
----         -----------         ------    -----       ------    ------------
sh1          primary_shardgroup  Ok        Deployed    region1   ONLINE
sh2          standby_shardgroup  Ok        Deployed    region2   READ_ONLY
sh3          primary_shardgroup  Ok        Deployed    region1   ONLINE
sh4          standby_shardgroup  Ok        Deployed    region2   READ_ONLY
sh5          primary_shardgroup  Ok        Deployed    region1   ONLINE
```

```
sh6          standby_shardgroup  Ok        Deployed    region2   READ_ONLY
```

7. チャンク構成を1-2分ごとにチェックして、チャンクの自動リバランスの進行状況を表示します。

```
$ gdsctl config chunks -show_Reshard
Chunks
------------------------
Database                       From      To
--------                       ----      --
sh1                            1         4
sh2                            1         4
sh3                            7         10
sh4                            7         10
sh5                            5         6
sh5                            11        12
sh6                            5         6
sh6                            11        12
Ongoing chunk movement
------------------------
Chunk     Source          Target             status
-----     ------          ------             ------
```

8. シャード(データベース)が自動的に登録されたことを確認します。

```
$ gdsctl databases
Database: "sh1" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1
 Region: region1
   Service: "oltp_ro_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%1
Database: "sh2" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1
 Region: region2
   Service: "oltp_ro_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%11
Database: "sh3" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1
 Region: region1
   Service: "oltp_ro_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%21
Database: "sh4" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1
 Region: region2
   Service: "oltp_ro_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%31
Database: "sh5" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1
```

```
 Region: region1
   Service: "oltp_ro_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%41
Database: "sh6" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1
 Region: region2
   Service: "oltp_ro_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "oltp_rw_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     cust_sdb%51
```

9. サービスが新しいシャードで自動的に起動されたことを確認します。

```
$ gdsctl services
Service "oltp_ro_srvc.cust_sdb.oradbcloud" has 3 instance(s). Affinity: ANYWHERE
   Instance "cust_sdb%11", name: "sh2", db: "sh2", region: "region2", status: ready.
   Instance "cust_sdb%31", name: "sh4", db: "sh4", region: "region2", status: ready.
   Instance "cust_sdb%51", name: "sh6", db: "sh6", region: "region2", status: ready.
Service "oltp_rw_srvc.cust_sdb.oradbcloud" has 3 instance(s). Affinity: ANYWHERE
   Instance "cust_sdb%1", name: "sh1", db: "sh1", region: "region1", status: ready.
   Instance "cust_sdb%21", name: "sh3", db: "sh3", region: "region1", status: ready.
   Instance "cust_sdb%41", name: "sh5", db: "sh5", region: "region1", status: ready.
```

関連項目:

GDSCTLコマンドの使用方法については、Oracle Database Global Data Services概要および管理ガイドを参照してください

シャードの置換

シャードで障害が発生してリカバリ不能になった場合、またはその他の理由で新しいホストにシャードを移動する必要がある場合は、GDSCTLでADD SHARD -REPLACEコマンドを使用して、シャードを置き換えることができます。

シャード・データベースに障害が発生し、同じホスト上で(RMANバックアップ/リストアなどの方法を使用して)データベースをリカバリできる場合は、-replaceパラメータを使用してシャードを置き換える必要はありません。シャードをローカルでリカバリできない場合や、なんらかの理由で別のホストまたはCDBにシャードを再配置する必要がある場合は、新しいホスト上にシャードのレプリカを作成できます。GDSCTLコマンドADD SHARDに-replaceオプションを指定すると、シャーディング構成を新しい情報で更新できます。

ADD SHARD -REPLACEを使用したシャードの置換が有用なケースを次に示します。

- シャード・データベースが実行されていたサーバー(マシン)が修復不可能な損害を受け、これを置き換える必要がある場合
- 正常に機能しているサーバーを別の(たとえば、より強力な)サーバーに置き換える必要がある場合
- PDB内のシャードが、あるCDBから別のCDBに再配置された場合

これらのいずれのケースでも、ADD SHARDを実行した後、シャードの数とシャード間のデータ分布は変更されず、シャードは同じデータを保持する別のシャードに置き換えられます。-replaceオプションを指定せずにADD SHARDを使用した場合は、これとは異なり、シャードの数が増加し、データが再分配されます。

ADD SHARD -REPLACEを実行すると、古いシャードのパラメータ(connect_stringやdb_unique_nameなど)は新しい値に置き換えられます。新しいデータベースには、障害が発生したデータベースとは異なるdb_unique_nameを付けることができます。Data Guardでは構成のすべてのメンバーに同じDBIDを付ける必要があるため、Data Guard構成内のスタンバイを置き換える場合は、新しいデータベースのDBIDが古いDBIDと一致している必要があります。

REPLACEを使用する前に

ADD SHARD -REPLACEを使用する前に、次の点を確認してください。

- データベースが(RMANリストアなどのメソッドを使用して)正しくリストアされていること。新しいデータベース・シャードは、障害が発生したシャードと同じシャーディング・メタデータを持つ必要があります。間違ったシャードへの接続文字列を指定しないように、基本的な検証を実行してください。
- 障害の発生したシャードが、障害が発生する前にデプロイ済の状態だったこと。
- 障害の発生したシャードが、ADD SHARD -REPLACEコマンドの実行時に停止していること。
- ファスト・スタート・フェイルオーバー(デフォルトで有効)が有効になっている場合は、ファスト・スタート・フェイルオーバー・オブザーバが実行されていること。

Data Guard環境でのシャードの置換

プライマリがまだ稼働している場合、ADD SHARD –REPLACEコマンドはスタンバイ・シャードを置き換える目的でのみ使用できます。障害が発生したプライマリ・シャードを置き換えるには、残りのいずれかのスタンバイがプライマリ・ロールに切り替わるまで待ってから、障害が発生したシャードを置き換えます。

切り替えができない(プライマリとすべてのスタンバイが停止している)場合は、プライマリから始めて、各メンバーに対してADD SHARD -REPLACEを実行する必要があります。これにより、白紙の状態から新しいブローカ構成が作成されます。

スタンバイが稼働していないMAXPROTECTIONモードでは、保護モードを維持するためにプライマリ・データベースが停止します。この場合、スタンバイが稼働していなければ、プライマリ・データベースをオープンできません。このシナリオで置換操作を処理するには、最初にデータベースをマウント・モードで起動し、DGMGRLを使用してData Guardの保護モードを(MAXAVAILABILITYまたはMAXPERFORMANCEに)ダウングレードする必要があります。保護モードを設定した後、プライマリ・データベースをオープンし、GDSCTLを使用して置換操作を実行します。置換操作の終了後は、DGMGRLを使用して保護モードを前のレベルに戻すことができます。

Data Guardでは構成のすべてのメンバーに同じDBIDを付ける必要があるため、Data Guard構成内のスタンバイを置き換える場合は、新しいデータベースのDBIDが古いDBIDと一致している必要があります。

例9-1 例1: 構成内にスタンバイがない場合のプライマリ・シャードの置換

次の例に示すように、初期構成には2つのプライマリ・シャードがデプロイされており、スタンバイはデプロイされていません。shdcは障害のシナリオで停止しているため、その可用性はダッシュ(-)で表示されています。

```
$ gdsctl config shard
Name    Shard Group    Status    State       Region    Availability
```

```
----    ----------    ------    -----       ------    ------------
shdb    dbs1          Ok        Deployed    east      ONLINE
shdc    dbs1          Ok        Deployed    east      -
```

リカバリするには、RMANなどを使用してバックアップからプライマリのレプリカを作成します。この例では、db_unique_name shddと接続文字列inst4を使用して新しいシャードが作成されています。これで、次のように古いシャードshdcを新しいシャードshardに置き換えることができます。

```
$ gdsctl add shard -replace shdc -connect inst4 -pwd password
DB Unique Name: SHDD
```

次のようにして構成を確認できます。

```
$ gdsctl config shard
Name     Shard Group      Status    State       Region    Availability
----     -----------      ------    -----       ------    ------------
shdb     dbs1             Ok        Deployed    east      ONLINE
shdd     dbs1             Ok        Deployed    east      ONLINE
```

例9-2 例2: スタンバイ・シャードの置換

構成にスタンバイ・シャードが含まれている場合は、プライマリ・シャードを置き換えることができないことに注意してください。このようなケースでプライマリに障害が発生した場合は、スタンバイのいずれかが自動切り替えによって新しいプライマリになった後で、置換操作を実行する必要があります。

初期構成には2つのシャードグループ(1つのプライマリと1つのスタンバイ)があり、それぞれに2つのシャードが含まれています。スタンバイshddが停止したときに、

```
$ gdsctl config shard
Name    Shard Group      Status    State       Region    Availability
----    -----------      ------    -----       ------    ------------
shdb    dbs1             Ok        Deployed    east      ONLINE
shdc    dbs1             Ok        Deployed    east      ONLINE
shdd    dbs2             Ok        Deployed    east      -
shde    dbs2             Ok        Deployed    east      READ ONLY
```

新しいスタンバイを作成します。プライマリが実行されているため、この操作はFOR STANDBYオプションを指定したRMAN DUPLICATEコマンドを使用して実行する必要があります。新しいスタンバイshdfの準備が完了したら、次のように古いシャードshddを置き換えます。

```
$ gdsctl add shard -replace shdd -connect inst6 -pwd password
DB Unique Name: shdf
```

次のようにして構成を確認できます。

```
$ gdsctl config shard
Name    Shard Group      Status    State       Region    Availability
----    -----------      ------    -----       ------    ------------
shdb    dbs1             Ok        Deployed    east      ONLINE
shdc    dbs1             Ok        Deployed    east      ONLINE
shde    dbs2             Ok        Deployed    east      READ ONLY
shdf    dbs2             Ok        Deployed    east      READ ONLY
```

Oracle GoldenGate環境でのシャードの置換

GDSCTLコマンドのオプションADD SHARD -REPLACEは、Oracle GoldenGateではサポートされていません。

一般的なエラー

ORA-03770: 不正なシャードが置換に指定されています

このエラーは、置換操作で指定されたシャードが元のシャードのレプリカでない場合にスローされます。具体的には、シャーディング・メタデータがこのシャードのシャード・カタログに格納されているメタデータと一致しません。データベースが(できればRMANを使用して)正しくコピーされたことを確認してください。これは包括的なチェックではありません。レプリカが正しく作成されたことを前提としています。

ORA-03768: 置換対象のデータベースがまだ稼働しています: shardc

add shard -replaceコマンドを実行するときは、置換対象のデータベースを停止する必要があります。GDSCTLコマンドconfig shardの出力で、これを確認してください。シャードに障害が発生したにもかかわらず、出力にONLINEと表示される場合は、しばらく(2分程度)待ってから再試行してください。

関連項目:

Oracle Database Global Data Services概要および管理ガイド(ADD SHARDコマンドの詳細)

チャンク管理

Oracle Enterprise Manager Cloud ControlおよびGDSCTLを使用したOracle Shardingデプロイメントのチャンクを管理できます。

次のトピックでは、チャンク管理の概念およびタスクについて説明します。

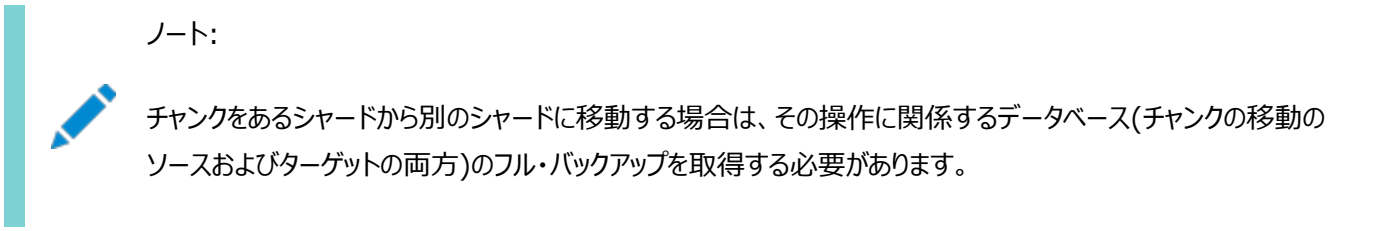
チャンクの移動について

チャンクをあるシャードから別のシャードに移動することが必要となる場合があります。シャード環境のスケーラビリティを維持するには、すべてのシャード間で負荷とアクティビティが均等に配分されるように維持することが重要です。

コンポジットSDBの環境が長期間使用されると、一部のシャードがよりアクティブになり、他のシャードよりデータが多くなります。環境内のバランスを保つために、よりアクティブなサーバーからアクティブではないサーバーにチャンクを移動する必要があります。チャンクを移動する理由は他にもあります。

- あるシャードが他のシャードよりアクティブになった場合は、よりアクティブではないシャードにチャンクを移動して、環境内で負荷が均等に配分されるようにできます。
- 範囲、リストまたはコンポジット・シャーディングを使用しているときに、シャードをシャードグループに追加する場合。
- 範囲、リストまたはコンポジット・シャーディングを使用しているときに、シャードグループからシャードを削除する場合。
- チャンクを分割したら、分割されたチャンクのいずれかを新しいシャードに移動することを多くの場合お薦めします。

スケーラビリティを維持するためにシャードを移動する場合、チャンクの最適なターゲットはよりアクティブではないシャード、またはより少ないデータがあるシャードです。Oracle Enterprise ManagerおよびAWRのレポートは、シャード間のアクティビティの配分を識別するため、およびチャンクの移動のよい候補となるシャードを識別するために役立ちます。



ノート:

チャンクをあるシャードから別のシャードに移動する場合は、その操作に関係するデータベース(チャンクの移動のソースおよびターゲットの両方)のフル・バックアップを取得する必要があります。

関連項目:

GDSCTL MOVE CHUNKコマンドの使用方法の詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください

進行中のチャンク移動操作の更新

MOVE CHUNK操作の進行中は、GDSCTL ALTER MOVEコマンドを使用して、操作で移動がスケジュールされている(移動がまだ開始されていない)すべてのチャンクを一時停止、再開または取消しできます。

このコマンドには3つのバリエーションがあります。-SUSPENDを使用してチャンク移行操作を延期し、-RESUMEを使用して移動プロセスを再起動し、-CANCELによってチャンク移行を取り消します。

また、-CHUNKおよび-SHARDオプションを使用して、スケジュールされたチャンク移動のリストをフィルタします。CONFIG CHUNKS -SHOW_RESHARDコマンドを使用して、スケジュール済チャンク移動のリストを取得できます。

チャンク移動の一時停止

ALTER MOVE -SUSPENDは、操作を再開または取消しを実行するまで、指定されたスコープのチャンク移行を延期します。操作を一時停止するシャードを指定し、ソース・シャードとターゲット・シャードをリストできます。一時停止する特定のチャンクのリストを指定することもできます。

定義したスコープ内のチャンクがすでに移動されている場合(スケジュール済以外の状態)、そのチャンクは一時停止されません。

たとえば、次のコマンドは、shard1との間でスケジュールされたすべてのチャンク移動を一時停止します。

```
GDSCTL> alter move -suspend -shard shard1
```

チャンク移動の再起動

ALTER MOVE -RESUMEは、指定されたシャード上の「move failed」フラグをリセットし、停止中または一時停止中のチャンク移動をすべて再起動します。

オプションで、移動の再起動前に「move failed」フラグがリセットされるソースおよびターゲット・シャードのリストを指定できます。シャードが指定されていない場合、進行中の移動が完了すると、一時停止された移動が再開されます。

たとえば、次のコマンドは、shard1との間でスケジュールされた一時停止または失敗したチャンク移動で、チャンク移動を再起動

します。

```
GDSCTL> alter move -resume -shard shard1
```

チャンク移動の取消し

ALTER MOVE -CANCELは、移動チャンク・スケジュールから指定されたチャンクを削除します。

-CHUNKオプションは、リストされたすべてのチャンクをスケジュールから削除することを指定し、-SHARDは、このデータベースとの間のすべてのチャンク移動をスケジュールから削除することを指定します。チャンクまたはシャードが指定されていない場合、まだ処理されていないチャンク移動はすべて取り消されます。

定義したスコープ内のチャンクが現在移動されている場合(スケジュール済以外の状態)、そのチャンク移動は取り消されません。

取り消されたチャンクは再開/再起動できません。これらのチャンクを移動するには、新しいMOVE CHUNKコマンドを発行する必要があります。

たとえば、チャンク1、2および3がまだ移動されていない場合は、次のコマンドによってチャンク移動スケジュールから削除されます。

```
GDSCTL> alter move -cancel -chunk 1,2,3
```

チャンクの移動

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントの1つのシャードから他のシャードにチャンクを移動できます。

1. シャード領域管理ページから、シャード・データベース・ターゲット・ページの左上隅にある「シャード領域」メニューを開き、「シャードグループの管理」を選択します。
2. リストのシャードグループを選択して、「チャンクの移動」をクリックします。
3. 「チャンクの移動」ダイアログで、チャンクを移動するソース・シャードと宛先シャードを選択します。
4. オプションのいずれかを選択して、移動するチャンクを選択します。
    - IDリストの入力: チャンクID番号のカンマ区切りリストを入力します
    - 表からIDを選択: 表のチャンクIDをクリックします
5. チャンクの移動が発生する時間を示します。
    - 即時: 確認後にチャンクの移動がプロビジョニングされます
    - 後で: 隣接しているフィールドのカレンダ・ツールを使用して、チャンクを移動するタイミングをスケジュールします
6. 「OK」をクリックします
7. ページ上部にある「情報」ボックスのリンクをクリックして、チャンク移動のプロビジョニング・ステータスを表示します。

チャンクの分割について

チャンクが大きくなりすぎた場合や、チャンクの一部のみを別のシャードに移行する必要がある場合は、シャード・データベースのチャンクを分割する必要があります。

Oracle Shardingはチャンクのオンライン分割をサポートします。理論上は、各シャードに1つのチャンクを作成し、データ移行が必要になるたびに分割できます。ただし、チャンクの分割はデータの可用性に影響しないものの、分割するパーティションのすべて

の行をスキャンしてから、1行ずつ新しいパーティションに挿入するため、時間がかかり、CPUにも負荷がかかる操作です。コンポジット・シャーディングの場合、チャンクの分割には時間がかかり、シャード・キーまたはスーパー・シャード・キーの新しい値を再定義するために停止時間が必要となることがあります。

したがって、各シャードに事前に複数のチャンクを作成し、次のいずれかの場合に分割することをお薦めします。再シャーディング中にデータをバランスよく分散するにはチャンク数が足りない場合、または特定のチャンクがホット・スポットになっている場合です。

システム管理のシャーディングの場合でも、単一のチャンクが他のチャンクより大きくなったり、よりアクティブになったりすることがあります。この場合は、そのチャンクを分割し、分割されたチャンクのいずれかが自動再シャーディングによって別のシャードに移動されることを許可すると、環境内のデータおよびアクティビティがより均等に配分されるように維持されます。

Oracle Enterprise Managerのヒート・マップには、どのチャンクが他のチャンクよりもアクティブになっているかが表示されます。この機能を使用してどのチャンクを分割できるかを識別し、分割されたチャンクのいずれかを別のシャードに移動して環境をリバランスできます。

関連項目:

GDSCTL SPLIT CHUNKコマンドの使用方法の詳細は、Oracle Database Global Data Services概要および管理ガイドを参照してください

チャンクの分割

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのチャンクを分割できます。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャード領域」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストのシャード領域を選択して、「チャンクの分割」をクリックします。
4. オプションのいずれかを選択して、分割するチャンクを選択します。
    - IDリストの入力: チャンクID番号のカンマ区切りリストを入力します
    - 表からIDを選択: 表のチャンクIDをクリックします
5. チャンクの分割が発生する時間を示します。
    - 即時: 確認後にチャンクの分割がプロビジョニングされます
    - 後で: 隣接しているフィールドのカレンダ・ツールを使用して、チャンクを分割するタイミングをスケジュールします
6. 「OK」をクリックします
7. ページ上部にある「情報」ボックスのリンクをクリックして、チャンク分割のプロビジョニング・ステータスを表示します。

チャンクを正常に分割すると、チャンクの数が「シャード領域」リストで更新されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

シャード・ディレクタ管理

Oracle Enterprise Manager Cloud Controlを使用したOracle Shardingデプロイメントのシャード・ディレクタを追加、編集および削除できます。

次の各項では、シャード・ディレクタの管理タスクについて説明します。

シャード・ディレクタの作成

Oracle Enterprise Manager Cloud Controlを使用して、シャード・ディレクタをOracle Shardingデプロイメントに作成して追加します。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャード・ディレクタ」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「作成」をクリックするか、リストからシャード・ディレクタを選択して「類似作成」をクリックします。

   「作成」を選択すると、フィールドのデフォルトの構成値とともに「シャード・ディレクタの追加」ダイアログが開きます。

   「類似作成」を選択すると、フィールドの選択されたシャード・ディレクタの構成値とともに「シャード・ディレクタの追加」ダイアログが開きます。リストからシャード・ディレクタを選択して「類似作成」オプションを有効化する必要があります。
4. 「シャード・ディレクタの追加」ダイアログの必要な情報を入力して、「OK」をクリックします。

   ノート:

   シャード・ディレクタの作成後すぐに実行を開始しない場合、「作成後にシャード・ディレクタを開始」チェックボックスの選択を解除する必要があります。
5. 確認ダイアログで「OK」をクリックします。
6. ページ上部にある「情報」ボックスのリンクをクリックして、シャード・ディレクタのプロビジョニング・ステータスを表示します。

シャード・ディレクタが正常に作成されると、「シャード・ディレクタ」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

シャード・ディレクタ構成の編集

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのシャード・ディレクタ構成を編集します。

Cloud Controlのシャード・ディレクタのリージョン、ポート、ローカル・エンドポイントおよびホスト資格証明を変更できます。シャード・ディレクタ名、ホストまたはOracleホームは編集できません。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャード・ディレクタ」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストからシャード・ディレクタを選択して、「編集」をクリックします。

シャード・ディレクタ名、ホストまたはOracleホームは編集できないことに注意してください。

4. フィールドを編集してGSMCATUSERパスワードを入力し、「OK」をクリックします。
5. ページ上部にある「情報」ボックスのリンクをクリックして、シャード・ディレクタ構成変更のプロビジョニング・ステータスを表示します。

シャード・ディレクタの削除

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントからシャード・ディレクタを削除します。

「シャード・ディレクタ」リストの列のチェック・マークに示されているように、削除するシャード・ディレクタが管理シャード・ディレクタの場合、削除する前に別のシャード・ディレクタを管理シャード・ディレクタに選択する必要があります。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャード・ディレクタ」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストからシャード・ディレクタを選択して、「削除」をクリックします。
4. ページ上部にある「情報」ボックスのリンクをクリックして、シャード・ディレクタ削除のプロビジョニング・ステータスを表示します。

シャード・ディレクタが正常に削除されると、「シャード・ディレクタ」リストに表示されなくなります。変更内容を参照するには、ページのリフレッシュが必要な場合があります。

リージョン管理

Oracle Enterprise Manager Cloud Controlを使用したOracle Shardingデプロイメントのリージョンを追加、編集および削除できます。

次の各項では、リージョンの管理タスクについて説明します。

リージョンの作成

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのシャード・データベース・リージョンを作成します。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「リージョン」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「作成」をクリックします。
4. 「リージョンの作成」ダイアログのリージョンに一意の名前を入力します。
5. オプションで、既存のリージョンからバディ・リージョンを選択します。
6. 「OK」をクリックします
7. ページ上部にある「情報」ボックスのリンクをクリックして、リージョンのプロビジョニング・ステータスを表示します。

リージョンが正常に作成されると、「リージョン」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場

合があります。

リージョン構成の編集

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのシャード・データベース・リージョン構成を編集します。

Cloud Controlのシャード・データベース・リージョンのバディ・リージョンを変更できます。リージョン名は編集できません。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「リージョン」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」でシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストからリージョンを選択して、「編集」をクリックします。
4. バディ・リージョンを選択または削除して、「OK」をクリックします。
5. ページ上部にある「情報」ボックスのリンクをクリックして、リージョン構成変更のプロビジョニング・ステータスを表示します。

リージョン構成が正常に更新されると、変更が「リージョン」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

リージョンの削除

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのシャード・データベース・リージョンを削除します。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「リージョン」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」でシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. リストからリージョンを選択して、「削除」をクリックします。
4. ページ上部にある「情報」ボックスのリンクをクリックして、リージョン削除のプロビジョニング・ステータスを表示します。

リージョン構成が正常に削除されると、変更が「リージョン」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

シャード領域管理

Oracle Enterprise Manager Cloud Controlを使用したOracle Shardingデプロイメントのシャード領域を追加、編集および削除できます。

次の各項では、シャード領域の管理タスクについて説明します。

シャード領域の作成

Oracle Enterprise Manager Cloud Controlを使用して、コンポジットOracle Shardingデプロイメントのシャード領域を作成します。

コンポジット方法を使用してシャードされるデータベースのみ、複数のシャード領域を格納できます。システム管理のシャード・デー

タベースは、1つのシャード領域のみ格納できます。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「シャード領域」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「作成」をクリックします。

ノート:

このオプションは、システム管理のシャード・データベースのシャード領域ページで無効化されます。

4. 「シャード領域の追加」ダイアログのフィールドの値を入力して、「OK」をクリックします。
   - 名前: シャード領域の一意の名前を入力します(必須)
   - チャンク: シャード領域に作成されるチャンクの数を入力します(デフォルトは120です)
   - 保護モード: Data Guard保護モードを選択します(デフォルトは「最大パフォーマンス」です)
5. ページ上部にある「情報」ボックスのリンクをクリックして、シャード領域のプロビジョニング・ステータスを表示します。

シャード領域が正常に作成されると、「シャード領域」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

コンポジット・シャード・データベースへのシャード領域の追加

新しいシャード領域の作成、シャード領域へのシャードの追加、新しいシャード領域での表領域セットの作成、追加したシャード領域のシャード表へのパーティションセットの追加を学習します。次に、表内のパーティションが、対応する表領域に新しく追加されたシャードに作成されていることを確認します。

既存のシャード・データベースに新しいシャード領域を追加するには、コンポジット・シャード・データベースがデプロイされ、すべてのDDLがシャードに伝播されていることを確認してください。

1. 新しいシャード領域を作成し、シャード領域にシャードを追加し、環境をデプロイします。
   a. シャード・カタログ・データベースに接続します。

```
GDSCTL> connect mysdbadmin/mysdbadmin_password
```

   b. シャード領域を追加し、シャードグループをシャード領域に追加します。

```
GDSCTL> add shardspace -chunks 8 -shardspace cust_asia
GDSCTL> add shardgroup -shardspace cust_asia -shardgroup asia_shgrp1 -deploy_as primary
-region region3
```

   c. シャードを追加します

```
GDSCTL> add shard -shardgroup asia_shgrp1 -connect
shard_host:TNS_listener_port/shard_database_name -pwd GSMUSER_password
GDSCTL> add shard asia_shgrp1 -connect shard_host:TNS_listener_port/shard_database_name
-pwd GSMUSER_password
```

   d. 環境をデプロイします。

```
GDSCTL> deploy
```

DEPLOYを実行すると、以前のすべてのDDLが新しいシャードでリプレイされ、すべての表が作成されます。パーティションは、デフォルトのSYS_SHARD_TS表領域に作成されます。

2. シャード・カタログでシャード領域用の表領域セットを作成し、シャード・セットをシャード・ルート表に追加します。
   a. 表領域セットを作成します。

```
SQL> CREATE TABLESPACE SET
  TSP_SET_3 in shardspace cust_asia using template
  (datafile size 100m autoextend on next 10M maxsize
   unlimited extent management
   local segment space management auto );
```

   b. パーティションセットを追加します。

```
SQL> ALTER table customers add PARTITIONSET asia VALUES ('ASIA"') TABLESPACE SET
TSP_SET_3 ;
```

   c. LOBが存在する場合は、lobの表領域セットを作成し、add partitionsetコマンドでLOB記憶域情報を記述します。

```
SQL> alter table customers add partitionset asia VALUES ('ASIA') tablespace set
TSP_SET_3 lob(docn) store as (tablespace set LOBTSP_SET_4)) ;
```

   d. ルート表にサブパーティションが含まれる場合、ストアを句として使用して、サブパーティションの表領域セットを指定します。

```
SQL> alter table customers add partitionset asia VALUES ('ASIA') tablespace set
TSP_SET_3 subpartitions store in(SUB_TSP_SET_1, SUB_TSP_SET_2);
```

   ADD PARTITIONSETコマンドは、子表を適切な表領域に移動します。

3. 新しいシャード領域内のパーティションが新しい表領域に移動されていることを確認します。

   新しいシャードに接続し、パーティションが新しい表領域セットに作成されていることを確認します。

```
SQL> select table_name, partition_name, tablespace_name, read_only from dba_tab_partitions;
```

シャードグループ管理

Oracle Enterprise Manager Cloud Controlを使用したOracle Shardingデプロイメントのシャードグループを追加、編集および削除できます。

次の各項では、シャードグループの管理タスクについて説明します。

シャードグループの作成

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのシャードグループを作成します。

1. シャードグループを追加するシャード領域を選択します。
2. シャード領域ターゲット・ページの左上隅にある「シャード領域」メニューを開き、「シャードグループの管理」を選択します。
3. 「作成」をクリックします。
4. 「シャードグループの作成」ダイアログで値を入力して、「OK」をクリックします。
5. ページ上部にある「情報」ボックスのリンクをクリックして、シャードグループのプロビジョニング・ステータスを表示します。

たとえば、前述のスクリーンショットの入力された値を使用して、次のコマンドを実行します。

```
GDSCTL Command: ADD SHARDGROUP -SHARDGROUP 'north' -SHARDSPACE 'shardspaceora'
 -REGION 'north' -DEPLOY_AS 'STANDBY'
```

シャードグループが正常に作成されると、「シャードグループの管理」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

サービス管理

Oracle Enterprise Manager Cloud Controlを使用したOracle Shardingデプロイメントのサービスを管理できます。

Oracle Shardingサービスを管理するには、シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「サービス」を選択します。サービスページで、サービスのリストの上部にあるコントロールを使用して、サービスを起動、停止、有効化、無効化、作成、編集および削除できます。

サービスを選択すると、サービスを実行しているホストおよびシャード、ステータス、状態およびそれらの各インスタンスのData Guardロールを表示するサービス詳細リストが開きます。このリストのシャードを選択すると、個々のシャードのサービスを有効化、無効化、起動および停止できます。

次の各項では、サービスの管理タスクについて説明します。

サービスの作成

Oracle Enterprise Manager Cloud Controlを使用して、Oracle Shardingデプロイメントのサービスを作成します。

1. シャード・データベース・ターゲット・ページの左上隅にある「シャードされたデータベース」メニューを開き、「サービス」を選択します。
2. 要求された場合、シャード・カタログ資格証明を入力し、「シャード・ディレクタ資格証明」で管理するシャード・ディレクタを選択し、シャード・ディレクタ・ホスト資格証明を選択してログインします。
3. 「作成」をクリックするか、リストからサービスを選択して「類似作成」をクリックします。

   「作成」を選択すると、フィールドのデフォルトの構成値とともに「サービスの作成」ダイアログが開きます。

   「類似作成」を選択すると、フィールドの選択されたサービスの構成値とともに「サービス{0}の類似作成」ダイアログが開きます。リストからサービスを選択して「類似作成」オプションを有効化する必要があります。
4. ダイアログに必要な情報を入力し、「OK」をクリックします。

   ノート:

   サービスの作成後すぐに実行を開始しない場合、「作成後にすべてのシャードでサービスを開始します。」チェックボックスの選択を解除する必要があります。
5. ページ上部にある「情報」ボックスのリンクをクリックして、サービスのプロビジョニング・ステータスを表示します。

サービスが正常に作成されると、「サービス」リストに表示されます。更新内容を参照するには、ページのリフレッシュが必要な場合があります。

10 Oracle Shardingによるデータ主権の実現

クラウド・コンピューティングの普及に伴い、特にデータとそのプライバシの保護に関する業界標準に関する関心が高まっています。今日、ほとんどの組織は、データがどこに保存され、誰がデータにアクセスできるかを把握することを必要としています。これにより、データ・レジデンシの管理が重大な関心事項になり、データを特定の地理的な場所に格納することが要求されるようになりました。

オンプレミスであれクラウドであれ、市民のデータをより厳格に保護および管理するために、すでに120か国以上がデータ保護に関するなんらかの国際プライバシ法に関与しています。

データ主権の概要

データ主権とは、一般的に、データが作成された地域に固有の規制によってデータが管理されることを指します。これらのタイプの規制では、データの保存場所、データへのアクセス方法、データの処理方法およびデータのライフサイクルが指定されることがあります。

国境やパブリック・クラウド・リージョンをまたぐデータの急増に伴い、100か国以上でデータの保存場所と転送方法に関する規制が可決されました。特に、個人を特定できる情報(PII)は、それが収集される国の法律およびガバナンス構造の対象となることが増えています。多くの場合、他の国へのデータ転送は、その国が同様のレベルのデータ保護を提供しているかどうか、およびその国が科学捜査で協力するかどうかに基づいて制限または許可されます。

データ主権の要件は地域の規制に基づくため、アプリケーション・アーキテクチャは様々なものになる可能性のあるます。そのいくつかは次のとおりです。

- データを特定の地理的な場所で物理的に保存する必要があります。たとえば、特定の国または複数の国で構成される地域の境界内です。データが遠隔地に保存されていないかぎり、データにリモートでアクセスして処理することは問題ありません。技術的な観点からは、これは、永続データを物理的に格納するデータベース、オブジェクト・ストア、メッセージング・ストアなどのデータ・ストアが特定の地理的な場所にある必要があることを意味します。ただし、データを処理するためのビジネス・ロジックを持つアプリケーション・ランタイムは、地理的な場所の外に配置できます。このようなアプリケーション・パーツの例には、アプリケーション・サーバー、モバイル・アプリケーション、APIゲートウェイ、ワークフローなどがあります。
- データを特定の地理的な場所で物理的に保存および処理する必要があります。この場合、データの保存と処理は、定義された地理的な場所内で行う必要があります。

Oracle Shardingを使用してデータ主権を実装するメリット

Oracle Shardingはデータ主権の要件を満たし、低レイテンシと高可用性を必要とするアプリケーションをサポートします。

- シャーディングにより、データの様々な部分を様々な国や地域に配置できるため、データを特定の管轄区域に配置する必要がある規制要件を満たします。
- また、特定のデータを利用者の近くに保存することもサポートしています。Oracle Shardingは、優れたランタイム・パフォーマンス、柔軟なスケーリング、ライフサイクル管理により、シャード・データベースのライフサイクル全体(デプロイメント、スキーマ作成、データ依存ルーティング)を自動化します。

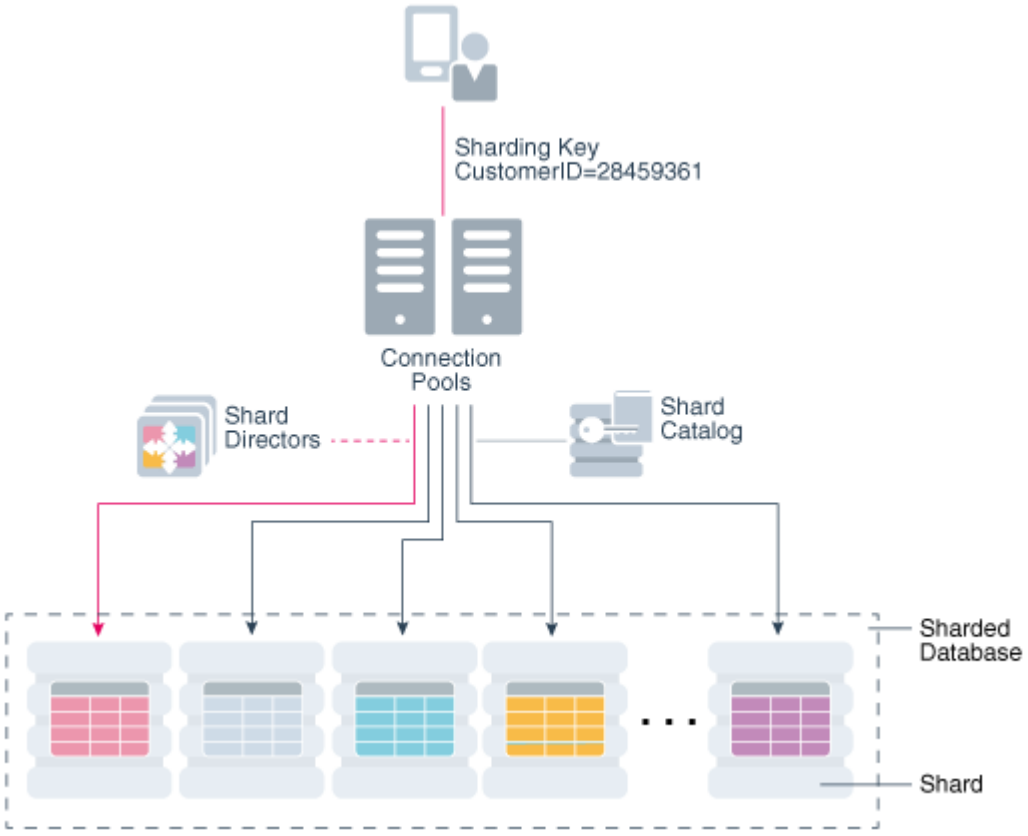
- リレーショナル・スキーマ、SQL、およびその他のプログラム・インタフェース、複雑なデータ型のサポート、オンライン・スキーマの変更、マルチコア・スケーラビリティ、高度なセキュリティ、圧縮、高可用性、ACIDプロパティ、読取り一貫性、JSONを使用した開発者の俊敏性などを含むエンタープライズRDBMSの利点も提供します。

Oracle Shardingによるデータ主権の実装

Oracle Shardingは、様々なコンピュータ、オンプレミスまたはクラウドの多数のデータベース(シャード)にデータ・セットのセグメントを分散します。これらのシャードは、世界中の複数のリージョンにデプロイできます。これにより、Oracle Shardingは、データ・レジデンシを考慮してグローバルに分散したデータベースを作成できます。

特定のデータベース内のすべてのシャードは、単一の論理データベースとしてアプリケーションに提供されます。アプリケーションは、実行する問合せに基づいて適切なシャードにシームレスに接続されます。たとえば、米国にデプロイされたアプリケーション・インスタンスがヨーロッパに存在するデータを必要とする場合は、特別な処理の必要なくアプリケーション・リクエストがEUデータ・センターにシームレスにルーティングされます。

図10-1 Oracle Shardingのアーキテクチャ



また、Real Application Security (RAS)やOracle Database VaultなどのOracle Databaseのセキュリティ機能を使用して、リージョン内でもデータ・アクセスをさらに制限できます。たとえば、すべてのEU諸国ではなく、一部の国からのデータのみがEUリージョンの管理者に表示されるようにさらに制限できます。データ主権リージョン内では、レプリケーションにOracle Data GuardおよびOracle GoldenGateを使用することで、データを複数のデータ・センターにレプリケートできます。

Oracle Sharding管理インタフェースにより、グローバル・メタデータを制御し、物理データベース(レプリカ)、それに含まれるデータ、レプリケーション・トポロジなどを表示できます。Oracle Shardingは、ノードが追加または削除されたときにデータの再分散

を処理します。

様々なリージョンからデータを実際にコピーすることなく、世界中のレポートにアクセスできます。シャーディングでは、どのリージョンからもデータをコピーせずにマルチシャード・レポートを実行できます。Oracle Shardingは、データが存在するノードに問合せをプッシュします。

Oracle Shardingは、次の側面に焦点を当てた包括的なデータ主権ソリューションを提供します。

- データ・レジデンシ: データを複数のシャードに分散でき、地理的に異なる場所にデプロイできます。
- データ処理: アプリケーション・リクエストは、アプリケーションの実行場所に関係なく、正しいシャードに自動的にルーティングされます。
- データ・アクセス: Oracle Databaseの仮想プライベート・データベース機能を使用して、リージョン内のデータ・アクセスをさらに制限できます。
- 派生データ: データをOracle Databaseに格納し、Oracle Databaseの機能を使用して派生データの増加を抑制します。
- データ・レプリケーション: Oracle ShardingをOracle Data GuardまたはOracle GoldenGateとともに使用して、同じデータ主権リージョン内でデータをレプリケートできます。

Oracle Shardingを使用してデータ主権を実現するユースケース

架空の大手金融機関であるShard Bankは、複数の国でユーザーにクレジット・サービスを提供しようとしています。クレジット・サービスが提供される各国には独自のデータ・プライバシ規制があり、個人を特定できる情報(PII)データを当該国に保存する必要があります。

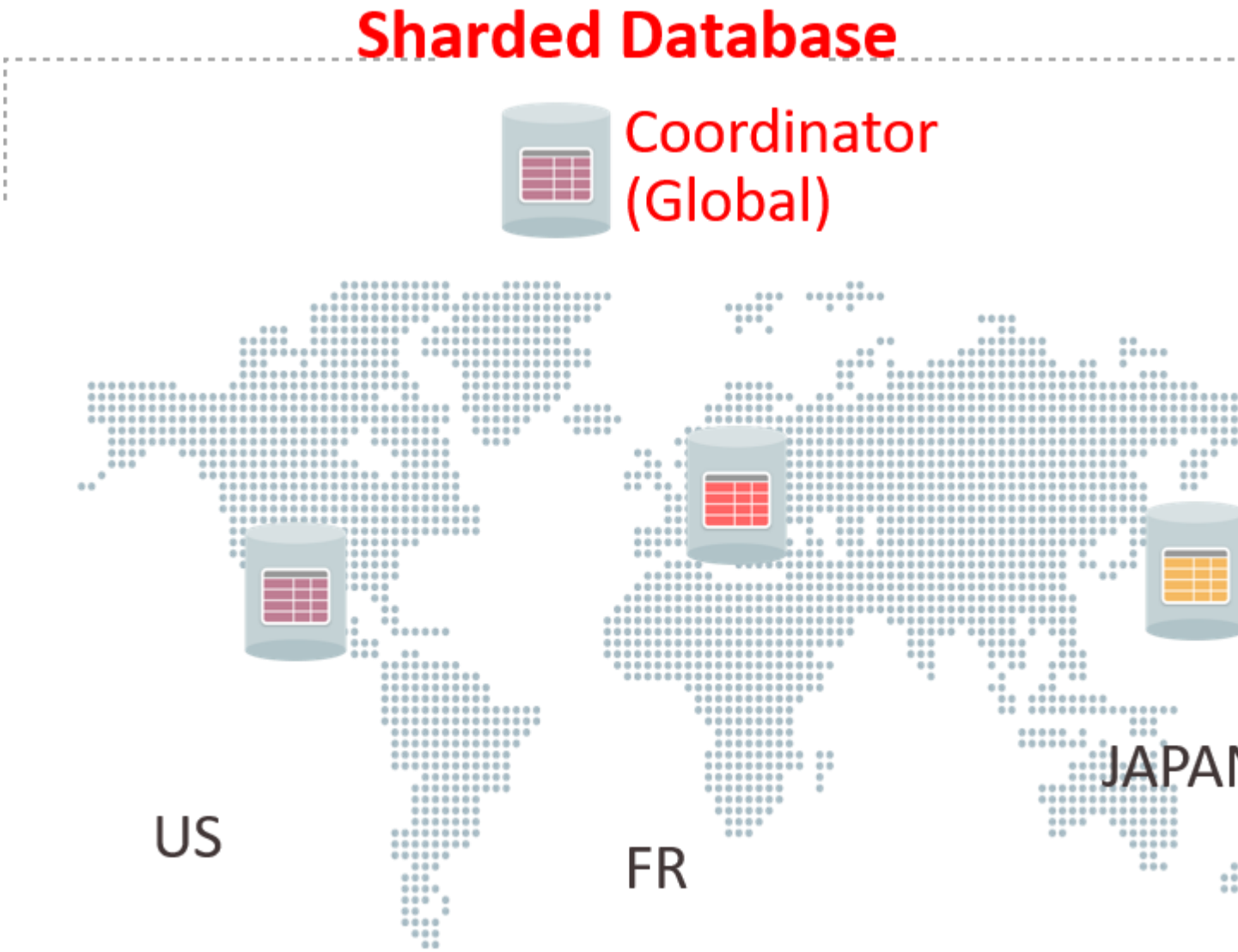
データへのアクセスを制限する必要があり、ある国のデータ管理者は他の国のデータを表示できません。このユースケースのソリューションは、様々な国でシャードが構成されたユーザー定義のシャーディングと、データ・アクセス制御のためのReal Application Security (RAS)です。

Oracle Shardingソリューションの概要

Oracle Shardingソリューションは、国内のデータ・ストレージを提供し、すべてのデータのグローバル・ビューもサポートしています。

次の例は、複数のリージョンを横断するOCIデータ・センターとオンプレミスの間のハイブリッドOracle Shardingユーザー定義デプロイメントを示しています。このOracle Sharding構成では、すべてのデータをローカルに保存および処理できます。各データベース(各主権リージョン内)はシャードになり、シャードは単一のシャード・データベースに属します。Oracle Shardingを使用すると、1つのシャード(1つの国内)のデータを問い合せることができます。Oracle Shardingは、(すべての国からデータを問い合せることができる)マルチシャード問合せをサポートしています。

図10-2 シャード・データベース



グローバル・シャード・データベースは、存在する必要がある国を示すキーでシャードされます。国内のアプリケーションは通常どおりローカル・データベースに接続し、すべてのデータはローカルに保存および処理されます。

マルチシャード問合せは、シャード・コーディネータに転送されます。コーディネータは問合せをリライトし、必要なデータがある各シャード(国)に送信します。コーディネータは、すべての国からの結果を処理して集計し、結果を返します。

Oracle Shardingでは、次の機能でこのユースケースが可能になります。

- 国内の問合せのシャードへの直接ルーティング。
- ユーザー定義のシャーディング方法により、国の範囲またはリストを使用して、シャード間でデータをパーティション化できます。
- Oracle Active Data Guardを使用してレプリケーションを自動構成し、レプリカを国内に制限します。

この方法のメリットは次のとおりです。

- 各シャードは、国内のクラウドまたはオンプレミスに配置できます。
- シャードは様々なクラウド・プロバイダを使用でき(マルチクラウド戦略)、シャードのレプリカを異なるクラウドまたはオンプレ

ミスに配置できます。

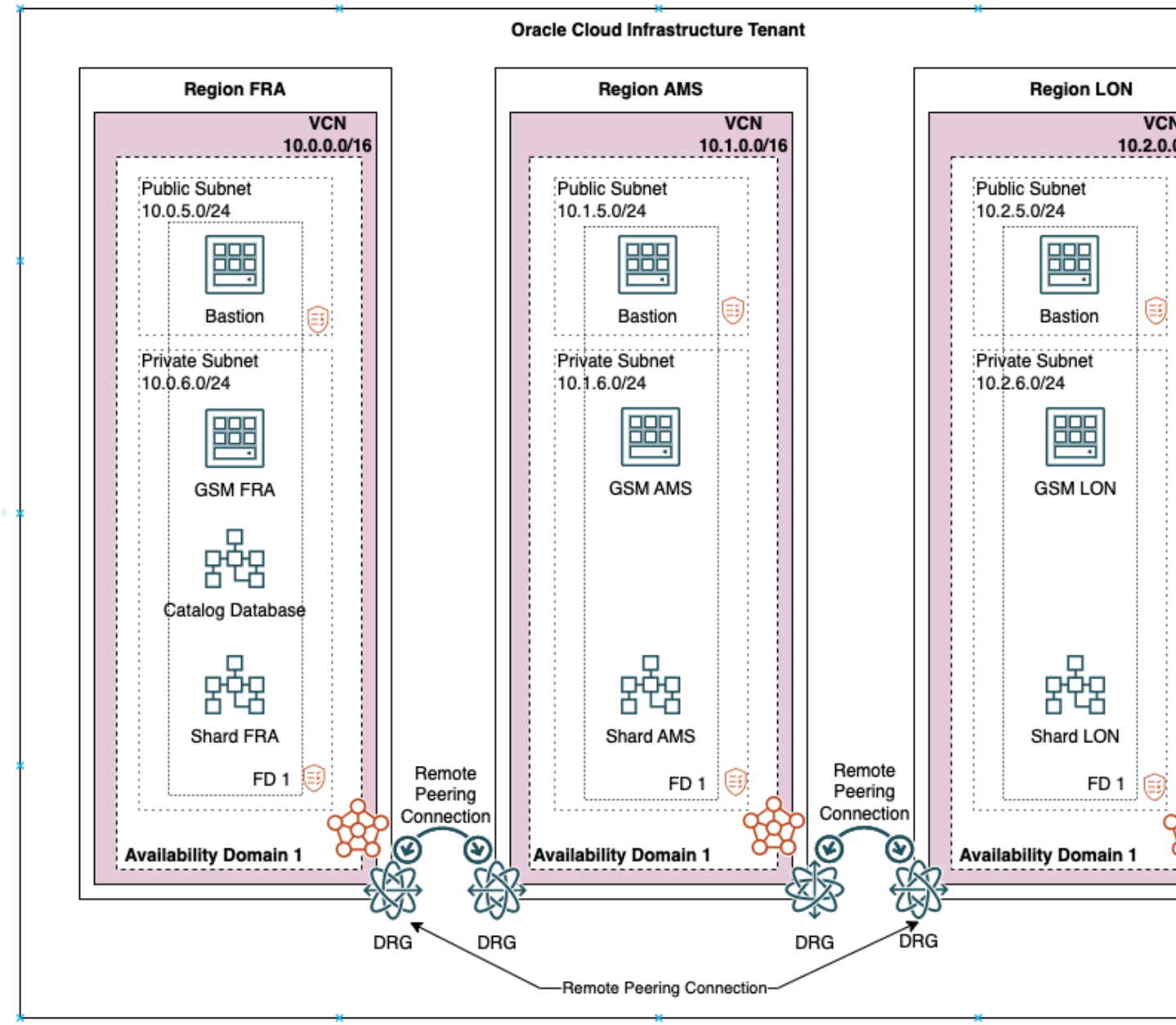
- オンライン再シャーディングにより、クラウド間、またはクラウドとオンプレミスとの間でデータを移動できます。
- データ主権の厳格な施行により、不注意によるリージョンをまたいだデータ漏えいを防止します。
- データの重複を低減した単一のマルチモデル・ビッグ・データ・ストア。
- 1つのリージョン/LOB内の計画/計画外停止時間が他のリージョン/LOBに影響しないため、より優れた障害分離が実現します。
- ビジー状態のパーティションとシャードを必要に応じて分割できます。
- 完全なACIDプロパティのサポートは、トランザクション・アプリケーションにとって重要です。

Oracle Shardingを使用したデータ主権のデプロイメント・トポロジ

このユースケースの例では、フランクフルト(リージョン1 FRA)、アムステルダム(リージョン2 AMS)、およびロンドン(リージョン3 LON)の3つのリージョンにまたがるOracle Cloud Infrastructureにシャード・データベースを作成します。

各リージョンは、シャード・ディレクタ(仮想マシン・グローバル・サービス・マネージャ(GSM))と1つのシャード(それぞれシステム・データベース・シャード1、2および3)をホストし、リージョン1 (FRA)はシャード・カタログ(システム・データベースGSMカタログ・データベース)をホストします。

図10-3 Oracle Shardingを使用したデータ主権のデプロイメント・トポロジ



Oracle Shardingを使用したデータ主権の構成

次のトピックで説明するステップを実行して、Oracle Shardingを使用してデータ主権を構成します。

3つのOCIリージョンすべてにおけるVCNネットワークの構成

Oracle Cloud Infrastructure (OCI)では、仮想クラウド・ネットワークは、インスタンスが実行される従来のネットワークの仮想バージョンです。各リージョン(FRA、AMSおよびLON)に仮想クラウド・ネットワーク(VCN)をデプロイして構成します。

各リージョンで、パブリックとプライベートの2つのサブネットを持つVCNを作成します。

1. プライベート・サブネットの新しいルート表を作成し、プライベート・サブネットに関連付けます。デフォルトのルート表はパブリック・サブネットにのみ使用し、プライベート・サブネットには専用のプライベート・ルート表を設定することが適切です。
2. インターネット・ゲートウェイを作成し、デフォルトのルート表に関連付けます。
3. ネットワーク・アドレス変換(NAT)ゲートウェイであるサービス・ゲートウェイを作成し、プライベート・サブネットのルート表に関連付けます。

FRAの例:

- VCN名/CIDER: Oracle Sharding VCN FRA 10.0.0.0/16
- パブリック・サブネット名/CIDER: public_fra 10.0.5.0/24
- プライベート・サブネット名/CIDER: private_fra 10.0.6.0/24

ノート:

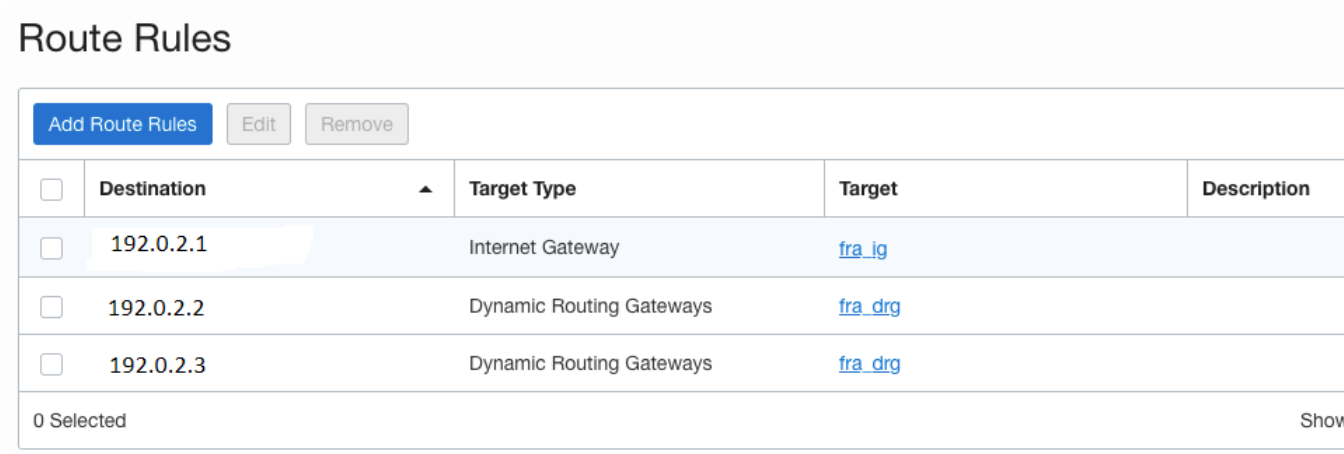
シャーディング・デプロイメントで使用されるすべてのリージョンでステップを繰り返します。サブネット CIDER は各リージョンで異なる必要があり、VCN/サブネット名にリージョン接頭辞を指定する必要があります。

3つのリージョン間のリモートVCNピアリングの構成

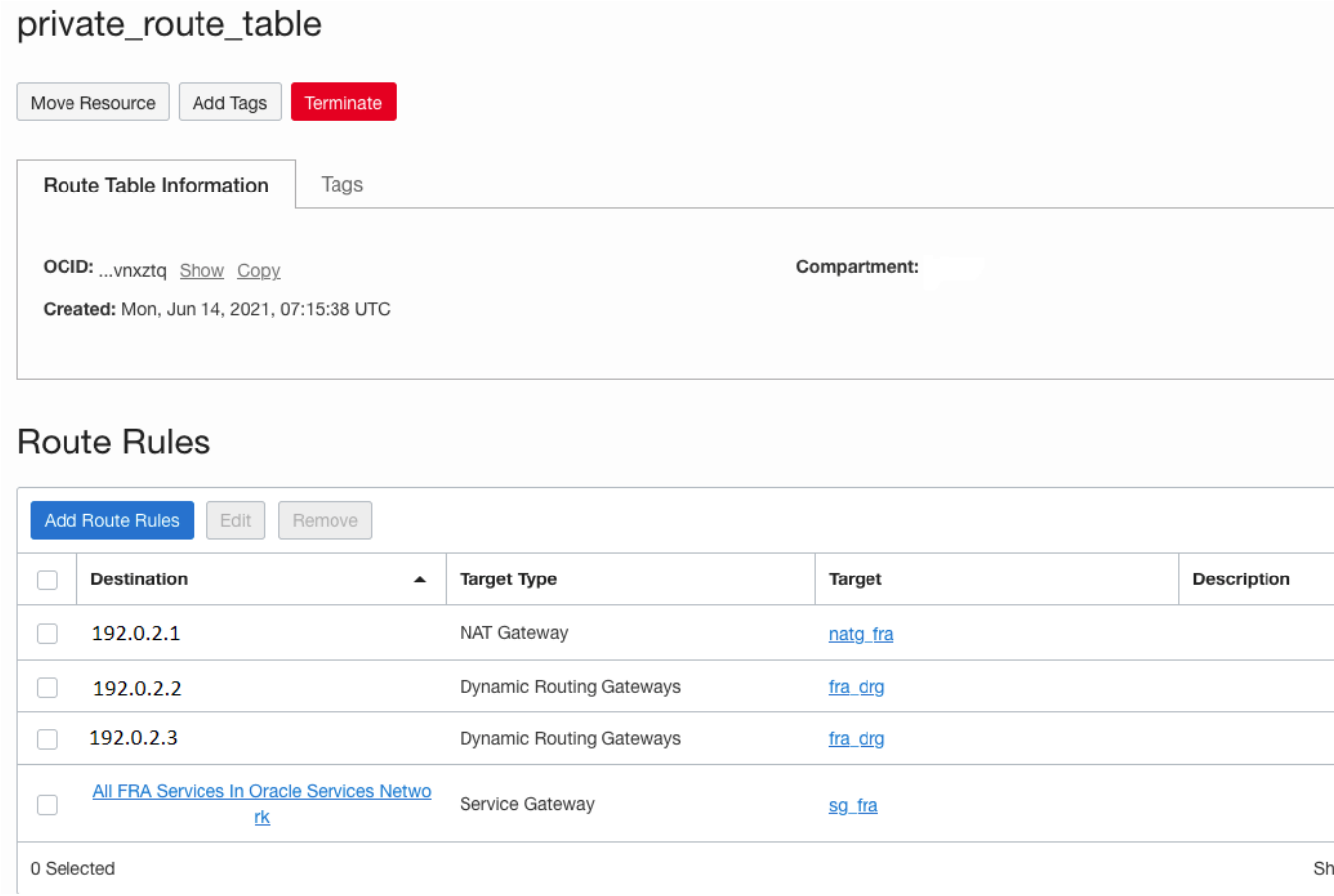
リモートVCNピアリングは、異なるリージョンにある2つのVCNを接続することで、トラフィックをインターネット経由でルーティングすることなく、プライベートIPアドレスを使用してVCNのリソースが通信できるようにするプロセスです。

トポロジ内の他の2つのリージョンに接続するために、各リージョンに2つのリモート・ピアリング接続(RPC)を構成します。

1. RPCを構成するステップについては、RPCを使用したリモートVCNピアリングに関する項を参照してください。
2. パブリック・サブネット/VCNのルーティング・ルールを構成します。

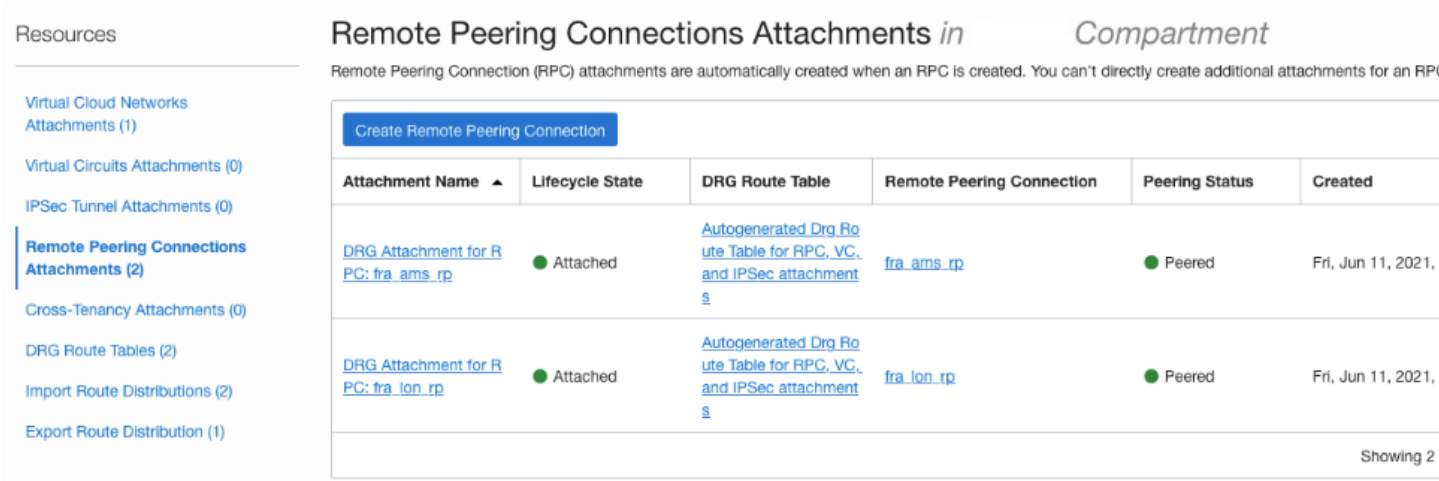


3. プライベート・サブネット/VCNのルーティング・ルールを構成します。



4. セキュリティ・ルールを構成します。

たとえば、リージョン1 (FRA)では次のようになります。



リージョン間のネーミング解決のためのプライベートDNSの構成

各リージョンのドメインごとにパブリック・サブネットとプライベート・サブネットのプライベート・ビューを作成し、1つのゾーン内に合計6つのプライベート・ゾーンを作成します。その後、すべてのエントリが各プライベート・ゾーン構成に追加されます。

1. プライベートDNSゾーンを作成および管理するには、プライベートDNSに関する項を参照してください。
2. 次のタスクに進む前に、すべての名前が正しく解決されていることを確認します。

ノート:

これらの手順は、名前を正しく解決できるように、すべての VCN/VM の各リージョンで実行する必要があります。

各リージョンへのグローバル・サービス・マネージャのインストール

Oracle Global Data Servicesグローバル・サービス・マネージャ(GSM)は、Oracle Shardingで、アプリケーションからシャード・データベース内の正しいシャードに問合せをルーティングするために使用されます。

ソフトウェアをダウンロードし、次のタスクを実行します。

- グローバル・サービス・マネージャ(Oracle Database 19c)ソフトウェアを要塞VMにダウンロードします。
- 最新バージョンのOPatchを適用します。
- 新しくインストールされたグローバル・サービス・マネージャ(Oracle Database 19c)に、利用可能な最新のOracle Databaseバンドル・パッチを適用します。

各リージョンにGSMをインストールするには:

1. iSCSIを使用して200 GBのブロック・ストレージを作成します。GSM用OCIコンピュートでiSCSIを構成します。ブロック・ストレージを/u01にマウントします。
   ブロック・ストレージのマウント・プロセスについては、一貫性のあるデバイス・パスを使用したボリュームへの接続に関する項を参照してください。
2. rootユーザーとして、必要なすべてのパッケージをインストールします。
   `# yum install -y oracle-database-preinstall-19c`
3. rootユーザーとして、/u01がoracle:oinstallによって所有されていることを確認します。
   `# chown oracle:oinstall /u01`
4. GSMソフトウェアを指定されたシャード・ディレクタVMにダウンロードし、サイレント・モードでインストールします。
   グローバル・サービス・マネージャのサイレント・インストールの実行に関する項を参照してください。
5. gsm homeを/etc/oratabに追加します。
   `gsm:/u01/app/oracle/product/19.0.0.0/dbhome_1:N`
6. 最新のOPatchバージョンを適用します。
7. Oracle Database 19cに利用可能な最新のバンドル・パッチ・バージョンを適用します。
8. ファイアウォールでGSMポートを開きます。

```
$ systemctl start firewalld.service
$ systemctl enable firewalld.service
$ firewall-cmd --permanent --zone=public --add-port=1522/tcp # firewall-cmd --reload
$ firewall-cmd --permanent --zone=public --list-ports
1522/tcp 22/tcp
```

9. アプリケーションがGSMに接続できるように、GSM VMに割り当てられたセキュリティ・リストで必要なポートが開いていることを確認します。

シャード・カタログおよびシャード・データベースのTNSエントリの収集

シャード・カタログおよびシャード・データベースの構成用にGSMサーバーを準備するには、TNSエントリのコレクションが必要です。シャード・カタログは、シャード・カタログ・オブジェクトを格納するPDBにのみアクセスする必要があります。ただし、シャード・データベースには、アプリケーション・スキーマを格納する各共有CDBおよびPDBのエントリを準備します。

1. シャード・カタログ・データベースおよびすべてのシャード(シャード・カタログとシャード)にアクセスするためのtnsnamesエントリを準備します。

2. これらのエントリをGSM VMの$ORACLE_HOME/network/admin/tnsnames.oraに追加します。

ノート:

接続文字列のホスト名には FQDN を使用します。

```
db_unique_name =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = host_name_fqdn)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = cdb_service_name)
    )
  )

pdb_name =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = host_name_fqdn)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdb_service_name)
    )
  )
```

シャード・カタログの構成

シャード・カタログは、Oracle Shardingのメタデータを管理します。シャード・カタログ・データベースとなるリージョン1 (FRA)でデータベースを構成します。

1. すべてのDBCSインスタンスに接続し、sqlnet.oraファイルに構成されているsqlnet暗号化アルゴリズムを更新し、クライアントおよびサーバーでサポートされるアルゴリズムとしてRC4_256暗号化方法を追加します。

ノート:

AES 暗号化は GSM: Enh 29496977 - GDS ONLY USES RC4_256 TYPE ENCRYPTION ではデフォルトではサポートされていないため、AES 暗号化を有効にするためにパッチが必要です。AES 暗号化を有効にするには、Oracle Database 19c でパッチを適用します。ただし、このパッチは Oracle Database 21c では必要ありません。

ノート:

RC4_256 アルゴリズムは、Oracle Database 19c でのみ必要です。

2. Oracle Shardingの要件を使用してシャード・カタログ・データベースを構成します。

```
SQL> alter system set open_links=16 scope=spfile;
SQL> alter system set open_links_per_instance=16 scope=spfile;
SQL> shu immediate
SQL> startup
```

3. シャード・カタログ・データベースでユーザーを構成します。

```
SQL> alter user gsmcatuser account unlock.
```

```
SQL> alter user gsmcatuser identified by password;
# Switch to PDB dedicated for catalog database
SQL> alter session set container=catalog_db_pdb;
SQL> create user mysdbadmin identified by password;
SQL> grant connect, create session, gsmadmin_role to mysdbadmin;
SQL> grant inherit privileges on user SYS to GSMADMIN_INTERNAL;
```

シャード・データベースの構成

Oracle Sharding構成でシャードになるデータベースを各リージョンに構成します。

1. すべてのDBCSインスタンスに接続し、sqlnet.oraファイルに構成されているsqlnet暗号化アルゴリズムを更新し、クライアントおよびサーバーでサポートされるアルゴリズムとしてRC4_256暗号化方法を追加します。

   ノート:

   AES 暗号化は GSM: Enh 29496977 - GDS ONLY USES RC4_256 TYPE ENCRYPTION ではデフォルトではサポートされていないため、AES 暗号化を有効にするためにパッチが必要です。AES 暗号化を有効にするには、Oracle Database 19c でパッチを適用します。ただし、このパッチは Oracle Database 21c では必要ありません。

   ノート:

   RC4_256 アルゴリズムは、Oracle Database 19c でのみ必要です。

2. 次のコマンドを実行します。

```
SQL> alter database flashback on;
SQL> alter system set dg_broker_start=true;
SQL> alter user GSMROOTUSER account unlock;
SQL> alter user GSMUSER account unlock;
SQL> alter user GSMADMIN_INTERNAL account unlock;
SQL> alter user GSMROOTUSER identified by password;
SQL> alter user GSMUSER identified by password;
SQL> alter user GSMADMIN_INTERNAL identified by password;
SQL> grant sysdg to gsmuser;
SQL> grant SYSBACKUP to gsmuser;
SQL> grant sysdg to GSMROOTUSER;
SQL> grant SYSBACKUP to GSMROOTUSER;
SQL> alter system set global_names=false;
SQL> shu immediate
SQL> startup
# Switch to PDB used as shared database
SQL> alter session set container= pdb_name;
SQL> grant read,write on directory DATA_PUMP_DIR to GSMADMIN_INTERNAL;
SQL> grant sysdg to gsmuser;
SQL> grant SYSBACKUP to gsmuser;
```

Oracle Shardingグローバル・データベースの作成

グローバル・サービス・マネージャ・リスナーを構成し、シャード・カタログ・データベースを作成して、すべてのシャードを構成に追加します。デプロイメント・ステップでは、すべてのシャードを単一のグローバル・データベースとして構成します。

1. Oracle Shardingでシャード・カタログを構成します。

   ノート:

   デフォルトでは、システム管理のシャーディングが構成されています。他のシャーディング方法が必要な場合は、シャードカタログの作成時に指定します。

   ```
   GDSCTL> create shardcatalog -database catalog_pdb_tns_entry -sharding user -user
   ```

   mysdbadmin/password -region region1

2. GSMリスナーを追加して起動します。GDSCTLからリスナーを実行します。

   ```
   GDSCTL> add gsm -gsm sharddirector1 -listener 1522 -pwd password -catalog pdb_tns_entry
         -region region1
   ```

3. 次のテンプレートを使用して、構成にシャードを追加します。シャード・データベースごとに繰り返します。

   FRAでシャードを追加します。

   ```
   GDSCTL> add invitednode shard_hostname
   GDSCTL> add cdb -connect cdb_conn_tns_entry -pwd gsmrootuser_pwd
   GDSCTL> add shardspace -shardspace primary_shardspace_fra
   GDSCTL> add shard -cdb cdb_conn_string -connect pdb_conn_string
    -shardspace primary_shardspace_fra -pwd gsmuser_pwd -deploy_as PRIMARY
   ```

   AMSでシャードを追加します。

   ```
   GDSCTL> add invitednode shard_hostname
   GDSCTL> add cdb -connect cdb_conn_tns_entry -pwd gsmrootuser_pwd
   GDSCTL> add shardspace -shardspace primary_shardspace_ams
   GDSCTL> add shard -cdb cdb_conn_string -connect pdb_conn_string
    -shardspace primary_shardspace_ams -pwd gsmuser_pwd -deploy_as PRIMARY
   ```

   LONでシャードを追加します。

   ```
   GDSCTL> add invitednode shard_hostname
   GDSCTL> add cdb -connect cdb_conn_tns_entry -pwd gsmrootuser_pwd
   GDSCTL> add shardspace -shardspace primary_shardspace_lon
   GDSCTL> add shard -cdb cdb_conn_string -connect pdb_conn_string
    -shardspace primary_shardspace_lon -pwd gsmuser_pwd -deploy_as PRIMARY
   ```

4. シャード・データベース構成をデプロイします。

   GDSCTL DEPLOYコマンドを実行して、次の出力を取得します。

   ```
   GDSCTL> deploy
   deploy: examining configuration...
   deploy: requesting Data Guard configuration on shards via GSM
   deploy: shards configured successfully
   The operation completed successfully
   ```

5. アプリケーションからの着信接続リクエストを処理するために、シャードでグローバル・データベース・サービスを作成します。

   グローバル・サービスは、従来のデータベース・サービスの拡張です。グローバル・サービスでは、従来のサービスのすべてのプロパティがサポートされます。シャード・データベースの場合、グローバル・サービスに追加のプロパティが設定されます。「グローバル・データベース・サービスの作成と開始」を参照してください。

   たとえば、データベース・ロール、レプリケーション・ラグの許容範囲、クライアントとシャードの間のリージョン・アフィニティな

どです。読取り/書込みトランザクション・ワークロードの場合は、シャード・データベースのプライマリ・シャードのデータにアクセスするために、単一のグローバル・サービスを作成します。Active Data Guardを使用した高可用性シャードの場合は、個別の読取り専用グローバル・サービスを作成します。

```
GDSCTL> add service -service oltp_rw_srvc -role primary
```

「シャード・データベースへの移行」で説明されている方法を使用して、データをシャードにロードします

関連項目

- C.35 create shardcatalog
- シャード・カタログ・データベースの作成

11 Oracle Shardingのトラブルシューティング

トレースの有効化、ログとトレース・ファイルの検索、および一般的な問題のトラブルシューティングを行うことができます。

次の各項では、Oracle Shardingのトラブルシューティングについて詳細に説明します。

トラブルシューティングのヒント

これらのヒントを使用して、問題のトラブルシューティングに必要なシャード・データベースに関する情報を検出します。

内容は次のとおりです。

- シャーディング方法の確認
- レプリケーション・タイプの確認
- Oracle Data Guard保護モードの確認
- キーにマップされているシャードの確認
- シャード操作モード(読取り専用または読取り/書込み)の確認
- DDLテキストの確認
- チャンク移行ステータスの確認
- 表タイプ(シャードまたは重複)の確認
- ユーザー・タイプ(ローカルまたはALL_SHARD)の確認
- シャード表領域として作成された表の識別
- シャードDDLが有効か無効かの確認
- シャーディング・キーによるデータのフィルタ処理

シャーディング方法の確認

gdsctl config sdbを実行して、シャード・データベースで使用されているシャーディング方法(シャード・タイプとも呼ばれます)を確認します。

シャーディング方法は、システム管理、コンポジットまたはユーザー定義が可能です。

次に示すように、シャーディング方法は、gdsctl config sdbの出力の「Shard type」の下に表示されます。

```
gdsctl> config sdb


GDS Pool administrators
------------------------

Replication Type
------------------------
Data Guard

Shard type
------------------------
System-managed

Shard spaces
------------------------
```

```
shd1

Services
------------------------
srv1
```

レプリケーション・タイプの確認

gdsctl config sdbを実行して、シャード・データベースのシャード・レプリケーションに使用されている方法を確認します。

次に示すように、レプリケーション・タイプは、gdsctl config sdbの出力の「Replication Type」の下に表示されます。

```
gdsctl> config sdb


GDS Pool administrators
------------------------

Replication Type
------------------------
Data Guard

Shard type
------------------------
System-managed

Shard spaces
------------------------
shd1

Services
------------------------
srv1
```

表11-1 config sdb出力のレプリケーション・タイプ

レプリケーション・タイプ	出力に表示される値
Oracle Data Guard	Data Guard
Oracle GoldenGate	Golden Gate

Oracle Data Guard保護モードの確認

DGMGRLに切り替えるのではなく、特定のシャード領域でgdsctl config shardspaceを実行して、GDSCTLセッションでOracle Data Guard保護モードを確認できます。

Data Guardは、MaxProtection、MaxAvailabilityおよびMaxPerformanceの3つの異なる保護モードで構成できます。

次に示すように、Data Guard保護モードは、gdsctl config shardspaceコマンド出力のPROTECTION MODEの下に表示されます。

```
GDSCTL> config shardspace -shardspace shd1
Shard Group                   Region                        Role
-----------                   ------                        ----
```

```
dbs1                          east                          Primary

PROTECTION_MODE               Chunks
--------------                ------
MaxProtection                 6
```

キーにマップされているシャードの確認

gdsctl config chunks -keyを実行して、シャーディング・キーにマップされているシャードを確認できます。

例1: 単一の表ファミリ

次の例では、シャード・データベース構成に1つの表ファミリのみが存在し、表はデータ型番号でパーティション化(シャード)されています。

この例では、ユーザーは、チャンク・シャーディング・キー値「2」のマップ先を確認します。出力では、シャーディング・キー2がチャンク「3」にマップされ、データベース「aime1b」に存在することが示されています。

```
GDSCTL> config chunks -key 2
Range Definition
------------------------
Chunks    Range Definition
------    ----------------
3         1431655764-2147483646

Databases
------------------------
aime1b
```

同様に、これは、シャーディングが行われるすべてのデータ型に対して実行できます。また、カンマ区切り値を使用して複数列のシャーディング・キーを確認することもできます。

範囲定義はハッシュ値の範囲であり、無視できます。

例2: 複数の表ファミリ

複数の表ファミリ構成で、オプション-table_familyを追加して、指定されたシャーディング・キーが属する表ファミリを指定します。

config chunksコマンドにより、トポロジ内のすべてのシャードグループのシャードがリストされます。この例では、データベース(シャード)リストに「aime1e」が追加されていることからわかるように、Data Guardスタンバイ・シャードグループもリストされています。

```
GDSCTL> config chunks -key 1 -table_family testuserfam3.customersfam1

Range Definition
------------------------
Chunks    Range Definition
------    ----------------
1         0-357913941

Databases
------------------------
aime1b
aime1e
```

例3: 複数列のシャーディング・キーの指定

表が複数列によってシャードされている場合は、次に示すように、シャーディング・キー値をカンマ区切りリストとして指定します。

```
GDSCTL> config chunks -key 10,mary,2010-04-04

Range Definition
------------------------
Chunks    Range Definition
------    ----------------
4         1288490187-1717986916

Databases
------------------------
aime1b
aime1e
```

シャード操作モード(読取り専用または読取り/書込み)の確認

gdsctl config chunks -cross_shardを実行して、シャードが読取り専用モードで実行されているか、または読取り/書込みモードで実行されているかを確認できます。

gdsctl config chunks -cross_shardコマンド出力には、次に示すように、読取り専用モードで実行されているシャード、および読取り/書込みモードで実行されているシャードが「Database」の下にリストされます。このコマンドにより、これらのシャードのチャンク範囲もリストされます。

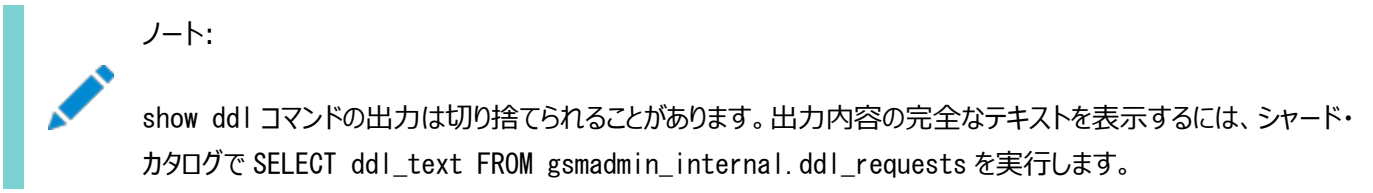
```
gdsctl config chunks -cross_shard
Read-Only cross shard targets
------------------------
Database                      From To
--------                      ---- --
tst3b_cdb2_pdb1               1    3
tst3c_cdb3_pdb1               9    10
tst3d_cdb2_pdb1               4    5
tst3e_cdb3_pdb1               6    8

Chunks not offered for cross-shard
------------------------
Shard space    From To
-----------    ---- --

Read-Write cross-shard targets
------------------------
Database                      From To
--------                      ---- --
tst3b_cdb2_pdb1               1    5
tst3c_cdb3_pdb1               6    10

Chunks not offered for Read-Write cross-shard activity
------------------------
Data N/A
```

DDLテキストの確認

gdsctl show ddl -ddl ddl_idを実行して、指定されたDDLのテキストを取得します。

次に示すように、DDL数値識別子は-ddl ddl_idを使用して指定され、特定のDDLのテキストおよびその他の詳細を取得します。

```
gdsctl show ddl -ddl 5
DDL Text: CREATE SHARDED TABLE Customers ( CustNo NUMBER NOT NULL, Name VARCHAR2(50), Address
VARCHAR2(250), Location VARCHAR2(20), Class VARCHAR2(3), CONSTRAINT RootPK PRIMARY KEY(CustNo))
PARTITION BY CONSISTENT HASH (CustNo) PARTITIONS AUTO TABLESPACE SET ts1
Owner: TESTUSER1
Object name: CUSTOMERS
DDL type: C
Obsolete: 0
Failed shards:
```



ノート:

show ddl コマンドの出力は切り捨てられることがあります。出力内容の完全なテキストを表示するには、シャード・カタログで SELECT ddl_text FROM gsmadmin_internal.ddl_requests を実行します。

チャンク移行ステータスの確認

gdsctl config chunks -show_reshardを実行して、チャンク移行のステータスを確認します。

チャンク移動は、ユーザーが開始したか内部的に開始されたか(増分デプロイ時)に関係なく、長時間実行される操作であるため、ステータスを確認する必要がある場合は、gdsctl config chunks -show_reshardを使用すると、移動が進行するにつれて、次のステータス・インジケータが表示されます。

- empty - 進行中のチャンク移動がないことを示します
- scheduled - チャンクは移動待ちです。これは、別のチャンク移動の完了を待機しているか、なんらかのエラーのために移動が開始されなかったためです
- running - 現在進行中です
- failed - チャンク移動に失敗しました。詳細は、GSMトレース、およびソース・データベースとターゲット・データベースのトレースを確認してください。

次の例では、コマンド出力の「Ongoing chunk movement」表にチャンク移動ステータスが表示されています。

```
gdsctl config chunks -show_reshard
Chunks
------------------------
Database                       From      To
--------                       ----      --
tst3b_cdb2_pdb1                1         6
tst3c_cdb3_pdb1                7         10
tst3d_cdb2_pdb1                1         6
tst3e_cdb3_pdb1                7         10

Ongoing chunk movement
------------------------
```

```
Chunk     Source                    Target                    status
-----     ------                    ------                    ------
7         tst3c_cdb3_pdb1           tst3b_cdb2_pdb1           Running
8         tst3c_cdb3_pdb1           tst3b_cdb2_pdb1           scheduled
9         tst3c_cdb3_pdb1           tst3b_cdb2_pdb1           scheduled
10        tst3c_cdb3_pdb1           tst3b_cdb2_pdb1           scheduled
```

表タイプ(シャードまたは重複)の確認

SELECT TABLE_NAME,SHARDED,DUPLICATED FROM user_tables;を使用して、dba/all/user_tablesの表がシャードであるか、重複であるかを確認できます。

次の例では、列「S」は表がシャードかどうかを示し、列「D」は表が重複かどうかを示します。

```
SQL> select TABLE_NAME,SHARDED,DUPLICATED from user_tables;

TABLE_NAME      S D
--------------- - -
CUSTOMERS       Y N
DUP1            N Y
LINEITEMS       Y N
MLOG$_DUP1      N N
ORDERS          Y N
```

ユーザー・タイプ(ローカルまたはALL_SHARD)の確認

dba/all/user_usersでユーザー名およびALL_SHARD列を選択することで、ローカル・ユーザーとして作成されたユーザー、およびシャード・データベース・ユーザーを確認できます。

```
SQL>  select USERNAME,ALL_SHARD from users_users where username='TESTUSER1';

USERNAME        ALL_SHARD
--------------- ---------
TESTUSER1       YES
```

シャード表領域として作成された表の識別

dba/all/user_tablespacesでTABLESPACE_NAME列およびCHUNK_TABLESPACE列を選択することで、シャード表に表領域が使用されているかどうかを確認できます。

シャード表の表領域である場合、dba/all/user_tablespacesのCHUNK_TABLESPACE列の値はYになります。

```
SQL> select TABLESPACE_NAME,CHUNK_TABLESPACE from user_tablespaces;

TABLESPACE_NAME               C
------------------------------ -
SYSTEM                         N
SYSAUX                         N
TEMP                           N
SYSEXT                         N
TS1                            Y
```

シャードDDLが有効か無効かの確認

現在のSQLセッションでシャードDDLが有効か無効かを確認できます。

これらの例は、シャードDDLを有効および無効にした後に、シャードDDLステータスを確認した結果を示しています。

```
SQL> alter session enable shard ddl;

Session altered.

SQL> select shard_ddl_status from v$session where AUDSID = userenv('SESSIONID');

SHARD_DD
--------
ENABLED

SQL> alter session disable shard ddl;

Session altered.

SQL> select shard_ddl_status from v$session where AUDSID = userenv('SESSIONID');

SHARD_DD
--------
DISABLED
```

シャーディング・キーによるデータのフィルタ処理

SHARD_QUERIES_RESTRICTED_BY_KEYパラメータを設定して、指定されたシャーディング・キーによるデータのフィルタ処理を有効または無効にできます。

パラメータSHARD_QUERIES_RESTRICTED_BY_KEYは、システム・レベルまたはセッション・レベルでALTERを使用して設定できます。有効にすると、DMLには、クライアント接続で設定された指定のSHARDING_KEYの選択データのみが表示されます。

次の例では、SHARDING_KEYが「1」と指定されたシャードを使用してクライアント接続が確立されています。ただし、クライアントがcustomers表でSELECTを実行すると、シャード内のその表内のすべての行が表示されます。

```
connection established for client with sharding_key=1
SQL> select * from customers order by custno;

    CUSTNO NAME       ADDRESS    LOCATION   CLA
---------- ---------- ---------- ---------- ---
         1 John       Oracle KM  Bangalore  A
        50 Larry      Oracle HQ  SFO        B

2 rows selected.

SQL>
```

次に示すように、セッション・レベルのフィルタ処理を有効にすると、同じSELECT文の結果は、クライアント接続で指定されたSHARD_KEYに一致する単一行のみに制限されます。

```
SQL> alter session set shard_queries_restricted_by_key = true;
```

```
Session altered.

SQL> select current_shard_key from dual;

CURRENT_SHARD_KEY
-----------------
                1

1 row selected.

SQL> select * from customers;

    CUSTNO NAME       ADDRESS    LOCATION   CLA
---------- ---------- ---------- ---------- ---
         1 John       Oracle KM  Bangalore  A
```

重複表のリフレッシュ率の設定

SHRD_DUPL_TABLE_REFRESH_RATEデータベース・パラメータを設定して、重複表のリフレッシュ率を変更できます。

デフォルトでは、重複表は60秒ごとにリフレッシュされます。次の例では、リフレッシュ間隔を100秒に増やしています。

```
SQL> show parameter refresh

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
shrd_dupl_table_refresh_rate         integer     60

SQL> alter system set shrd_dupl_table_refresh_rate=100 scope=both;

System altered.

SQL> show parameter refresh

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
shrd_dupl_table_refresh_rate         integer     100
```

Oracle Shardingのトレースおよびデバッグ情報

次の各項では、トレースを有効化する方法とログを見つける方法について説明します。

Oracle Shardingに対するトレースの有効化

シャード・データベース内の問題を追跡するため、PL/SQLトレースを有効にします。

完全なトレースを取得するには、次に示すようにGWM_TRACEレベルを設定します。次の文を実行すると、ただちにトレースが有効になりますが、データベースを再起動するとトレースは無効になります。

```
ALTER SYSTEM SET EVENTS 'immediate trace name GWM_TRACE level 7';
```

次の文を実行すると、永続的に動作するトレースが有効になりますが、データベースを再起動しないと起動しません。

```
ALTER SYSTEM SET EVENT='10798 trace name context forever, level 7' SCOPE=spfile;
```

前述の両方のトレースを完全に設定することをお薦めします。

Oracle Sharding環境のすべてをトレースするには、シャード・カタログとすべてのシャードでトレースを有効にする必要があります。トレースは、シャード・カタログのGDSCTLセッションまたは個々のシャードでシャード・ディレクタ(GSMとも呼びます)によって作成されたセッションのRDBMSセッション・トレース・ファイルに書き込まれます。

Oracle Shardingのアラート・ログおよびトレース・ファイルの検索場所

Oracle Sharding環境には、トレース・ファイルとアラート・ログを検索する場所がいくつかあります。

diag/rdbms/..にある標準RDBMSトレース・ファイルには、トレース出力が格納されます。

「deploy」の出力は、ジョブ・キューのトレース・ファイルdb_unique_name_jXXX_PID.trcに格納されます。

他のGDSCTLコマンドの出力は、使用される接続文字列に応じて、共有サーバー・トレース・ファイルdb_unique_name_sXXX_PID.trcまたは専用トレース・ファイルdb_unique_name_ora_PID.trcのいずれかに格納されます。

通常、カタログおよびシャードへの接続の多くに共有サーバーが使用されるため、トレースはSID_s00*.trcという名前の共有サーバー・トレース・ファイルに含まれています。

GDSCTLには、ステータスおよびエラー情報を表示できるコマンドがいくつか用意されています。

シャード・ディレクタ(GSM)のトレース・ファイルとログ・ファイルの場所を表示するには、GDSCTL STATUS GSMを使用します。

```
GDSCTL> status
Alias                     SHARDDIRECTOR1
Version                   18.0.0.0.0
Start Date                25-FEB-2016 07:27:39
Trace Level               support
Listener Log File         /u01/app/oracle/diag/gsm/slc05abw/sharddirector1/alert/log.xml
Listener Trace File       /u01/app/oracle/diag/gsm/slc05abw/sharddirector1/trace/
ora_10516_139939557888352.trc
Endpoint summary          (ADDRESS=(HOST=shard0)(PORT=1571)(PROTOCOL=tcp))
GSMOCI Version            2.2.1
Mastership                N
Connected to GDS catalog  Y
Process Id                10535
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       TRUE
Total messages published  71702
Time Zone                 +00:00
Orphaned Buddy Regions:   None
GDS region                region1
Network metrics:
   Region: region2 Network factor:0
```

非XMLバージョンのalert.logファイルは、次に示す/traceディレクトリにあります。

```
/u01/app/oracle/diag/gsm/shard-director-node/sharddirector1/trace/alert*.log
```

GSMのログ出力を復号化するには、次のコマンドを使用します。

```
GDSCTL> set _event 17 -config_only
```

マスター・シャード・ディレクタ(GSM)のトレース/アラート・ファイルには、すべての非同期コマンドまたはバックグラウンド・タスク(move chunk、split chunk、deploy、シャード登録、Data Guard構成、シャードDDLの実行など)のステータスとエラーが含まれています。

シャード・ディレクタのエラー・ステータスを含む保留中のAQリクエストを見つけるには、GDSCTL CONFIGを使用します。

進行中およびスケジュール済のチャンク移行を表示するには、GDSCTL CONFIG CHUNKS -show_reshardを使用します。

DDLが失敗したシャードを表示するには、GDSCTL SHOW DDL -failed_onlyを使用します。

特定のシャードのDDLエラー情報を表示するには、GDSCTL CONFIG SHARD -shard shard_nameを使用します。

シャード・データベースの一般的なエラー・パターンおよび解決

Oracle Shardingの一般的なエラーをトラブルシューティングする方法の詳細は、次の各項を参照してください。

リモート・スケジューラ・エージェントの起動時の問題

すべてのシャード・ホストでリモート・スケジューラ・エージェントの起動時に問題が発生した場合は、次の操作を試行します。

スケジューラを起動するには、各シャード・サーバーのORACLE_HOMEに移動する必要があります。

```
[oracle@shard2 ~]$ echo welcome | schagent -registerdatabase 192.0.2.24 8080
Agent Registration Password?
Failed to get agent Registration Info from db: No route to host
Solution: Disable firewall
service ipchains stop
service iptables stop
chkconfig ipchains off
chkconfig iptables off
```

シャード・ディレクタの起動時の障害

シャード・ディレクタの起動時に問題が発生した場合は、次の操作を試行します。

スケジューラを起動するには、各シャード・サーバーのORACLE_HOMEに移動する必要があります。

```
GDSCTL>start gsm -gsm shardDGdirector
GSM-45054: GSM error
GSM-40070: GSM is not able to establish connection to GDS catalog
GSM alert log, /u01/app/oracle/diag/gsm/shard1/sharddgdirector/trace/alert_gds.log
GSM-40112: OCI error. Code (-1). See GSMOCI trace for details.
GSM-40122: OCI Catalog Error. Code: 12514. Message: ORA-12514: TNS:listener does not
currently know of service requested in connect descriptor
GSM-40112: OCI error. Code (-1). See GSMOCI trace for details.
2017-04-20T22:50:22.496362+05:30
Process 1 in GSM instance is down
GSM shutdown is successful
```

```
GSM shutdown is in progress
NOTE : if not message displayed in the GSM log then enable GSM trace level to 16
while adding GSM itself.
```

1. 起動に失敗した新規作成のシャード・ディレクタ(GSM)を削除します。

```
GDSCTL> remove gsm -gsm shardDGdirector
```

2. トレース・レベル16を使用してシャード・ディレクタを追加します。

```
GDSCTL> add gsm -gsm shardDGdirector -listener port_num -pwd gsmcatuser_password
 -catalog hostname:port_num:shard_catalog_name
 -region region1 -trace_level 16
```

3. シャード・カタログ・データベースがデフォルト以外のポート(1521以外)で実行されている場合は、リモート・リスナーを設定します。

```
SQL> alter system set local_listener='(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=hostname)(PORT=port_num)))';
```

CREATE SHARDで作成されたシャードのエラー

GDSCTL CREATE SHARDコマンドで作成されたシャードのDEPLOY実行中に発生したエラーについては、次の項目を確認します。

- シャード・ホスト上のリモート・スケジューラ・エージェントのログ
- シャード・カタログのDBA_SCHEDULER_JOB_RUN_DETAILSビュー
- シャード・ホスト上の$ORACLE_BASE/cfgtoollogsにあるNETCA/DBCAの出力ファイル

CREATE SHARDの使用時の問題

GDSCTL CREATE SHARDコマンドの使用時に発生した問題には、次のような解決策があります。

次のエラーを回避するには$ORACLE_BASE/oradataおよび$ORACLE_BASE/fast_recovery_areaディレクトリを作成する必要がある

```
GDSCTL> create shard -shardgroup primary_shardgroup -destination che -osaccount
 oracle -ospassword oracle
GSM-45029: SQL error
ORA-03710: directory does not exist or is not writeable at destination:
 $ORACLE_BASE/oradata
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 6920
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 4730
ORA-06512: at line 1
GDSCTL>create shard -shardgroup primary_shardgroup -destination che -osaccount oracle
 -ospassword oracle
GSM-45029: SQL error
ORA-03710: directory does not exist or is not writeable at destination:
 $ORACLE_BASE/fast_recovery_area
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 6920
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 4755
ORA-06512: at line 1
```

解決策: すべてのシャード・ホスト上の$ORACLE_BASEの下に/oradataとfast_recovery_areaを作成します。

権限の問題

```
GDSCTL>create shard -shardgroup primary_shardgroup -destination blr -credential cred
GSM-45029: SQL error
ORA-02610: Remote job failed with error:
EXTERNAL_LOG_ID="job_79126_3",
USERNAME="oracle",
STANDARD_ERROR="Launching external job failed: Login executable not setuid-root"
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 6920
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 4596
ORA-06512: at line 1
```

解決策: 次のディレクトリに対するroot権限があることを確認します。

```
chown root $ORACLE_HOME/bin/extjob
chmod 4750 $ORACLE_HOME/bin/extjob
chown root $ORACLE_HOME/rdbms/admin/externaljob.ora
chmod 640 $ORACLE_HOME/rdbms/admin/externaljob.ora
chown root $ORACLE_HOME/bin/jssu
chmod 4750 $ORACLE_HOME/bin/jssu
```

CREATE SHARDでのエラー

```
GDSCTL>create shard -shardgroup primary_shardgroup -destination mysql02 -osaccount
 oracle -ospassword oracle
GSM-45029: SQL error
ORA-03719: Shard character set does not match catalog character set.
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 7469
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 5770
ORA-06512: at line 1
```

解決策: Javaのバージョンを確認します(シャード・カタログおよびすべてのシャード・サーバーで同じである必要があります)。

```
rpm -qa|grep java
```

deployコマンドの使用時の問題

```
GDSCTL> deploy
GSM-45029: SQL error
ORA-29273: HTTP request failed
ORA-06512: at "SYS.DBMS_ISCHED", line 3715
ORA-06512: at "SYS.UTL_HTTP", line 1267
ORA-29276: transfer timeout
ORA-06512: at "SYS.UTL_HTTP", line 651
ORA-06512: at "SYS.UTL_HTTP", line 1257
ORA-06512: at "SYS.DBMS_ISCHED", line 3708
ORA-06512: at "SYS.DBMS_SCHEDULER", line 2609
ORA-06512: at "GSMADMIN_INTERNAL.DBMS_GSM_POOLADMIN", line 14284
ORA-06512: at line 1
```

解決策: $ORACLE_HOME/data/pendingjobsで正確なエラーを確認します。ウォレットに問題がある場合は、ORA-1017がスローされます。

1. 問題のあるシャード・ホスト上で、リモート・スケジューラ・エージェントを停止します。

```
schagent -stop
```

2. データベース・ホーム上のウォレット・ディレクトリの名前を変更します。

```
mv $ORACLE_HOME/data/wallet $ORACLE_HOME/data/wallet.old
```

3. リモート・スケジューラ・エージェントを起動すると、新しいウォレット・ディレクトリが作成されます。

```
schagent -start
schagent -status
echo welcome | schagent -registerdatabase 10.10.10.10 8080
```

12 Oracle Shardingリファレンス

次のトピックでは、Oracle Shardingシャード・データベース構成の計画、構成、デプロイ、および管理に役立つ参照情報を示します。

Oracle ShardingでのGDSCTLの使用

Oracle Sharding構成の設定およびデプロイでは、Global Data ServicesのGDSCTLコマンドをいくつか使用します。次のトピックでは、GDSCTLコマンドライン・ツールおよびOracle Sharding関連のGDSCTLコマンドの使用方法を学習します。

GDSCTLの起動

GDSCTLを起動するには、オペレーティング・システムのプロンプトで、gdsctlを入力します。

```
$ gdsctl
```

GDSCTLが起動し、GDSCTLコマンド・プロンプトが表示されます。

```
GDSCTL>
```

GDSCTLコマンドの対話的な実行

GDSCTLコマンドは、オペレーティング・システム・プロンプトでもGDSCTLコマンド・プロンプトでも対話的に実行できます。

システム・プロンプトでGDSCTLコマンドを実行します。

```
$ gdsctl add gsm -gsm gsm1 -catalog 127.0.0.1:1521:db1
```

GDSCTLコマンド・プロンプトでGDSCTLコマンドを実行します。

```
GDSCTL> add gsm -gsm gsm1 -catalog 127.0.0.1:1521:db1
```

これらの方法のどちらでも同じ結果が得られます。このドキュメントのコマンド構文の例では、GDSCTLコマンド・プロンプトを使用します。

GDSCTLバッチ操作の実行

すべてのGDSCTLコマンドを1つのファイルにまとめ、バッチとして実行できます。

次のコマンドはGDSCTLを起動し、指定されたスクリプト・ファイルに含まれるコマンドを実行します。

```
$ gdsctl @script_file_name
```

GDSCTLのヘルプ・テキスト

GDSCTLおよびGDSCTLコマンドのヘルプを表示できます。

GDSCTL HELPコマンドでは、すべてのGDSCTLコマンドのサマリーが表示されます。

```
GDSCTL> help
```

HELPの後にコマンド名を指定すると、そのコマンドのヘルプ・テキストが表示されます。

```
GDSCTL> help start gsm
```

また、GDSCTLコマンドに-hオプションを使用すると、指定したコマンドのヘルプ・テキストを表示できます。

```
GDSCTL> start gsm -h
```

GDSCTL接続

一部のGDSCTLコマンドではシャード・カタログに接続する必要があり、特定の操作の場合、GDSCTLはシャード・ディレクタに接続する必要があります。

GDSCTLとシャード・カタログの接続

シャード・カタログへの接続が必要なGDSCTLコマンドを実行する場合は、接続が必要な最初のコマンドの前に、GDSCTLのCONNECTコマンドを実行する必要があります。

CONNECTコマンドは、GDSCTLセッションで1回実行するのみで済みます。

GDSCTLは、Oracle Net Servicesを使用して、Oracle Sharding構成のシャード・カタログ・データベースまたは他のデータベースに接続します。これらの接続の場合、必要なネットワーク構成を持つクライアントまたはホストからGDSCTLを実行できます。

指定がない場合、GDSCTLは現在の名前の解決方法(TNSNAMESなど)を使用して接続文字列を解決します。

シャード・カタログへの接続が必要なGDSCTL操作は、各コマンドの使用上のノートに示されています。

GDSCTLとシャード・ディレクタの接続

特定の操作では、GDSCTLはシャード・ディレクタ(グローバル・サービス・マネージャとも呼ばれます)に接続する必要があります。

指定がない場合、GDSCTLは現在の名前の解決方法(TNSNAMESなど)を使用して接続文字列を解決します。ただし、シャード・ディレクタ名を解決するために、GDSCTLはgsm.oraファイルを参照します。

シャード・ディレクタに接続するには、シャード・ディレクタと同じホストでGDSCTLが実行されている必要があります。シャード・ディレクタに接続するときに、GDSCTLはローカル・シャード・ディレクタに関連付けられているgsm.oraファイルを探します。

次に、シャード・ディレクタへの接続を必要とするGDSCTL操作を示します。

- ADD GSMはシャード・ディレクタを追加します。
- START GSMはシャード・ディレクタを起動します。
- STOP GSMはシャード・ディレクタを停止します。
- MODIFY GSMは、シャード・ディレクタの構成パラメータを変更します。
- STATUS GSMは、シャード・ディレクタのステータスを返します。
- SET INBOUND_CONNECT_LEVELでは、INBOUND_CONNECT_LEVELリスナー・パラメータを設定します。

- SET TRACE_LEVELでは、指定されたシャード・ディレクタに関連付けられているリスナーのトレース・レベルを設定します。
- SET OUTBOUND_CONNECT_LEVELでは、指定されたシャード・ディレクタに関連付けられているリスナーのアウトバウンド接続のタイムアウト値を設定します。
- SET LOG_LEVELでは、特定のシャード・ディレクタに関連付けられているリスナーのログ・レベルを設定します。

Oracle Shardingで使用されるGDSCTLコマンド

Oracle Shardingでは、GDSCTLコマンドのサブセットが使用されます。

- add cdb
- add credential
- add file
- add gsm
- add invitednode (add invitedsubnet)
- add region
- add service
- add shard
- add shardgroup
- add shardspace
- config
- config cdb
- config chunks
- config credential
- config file
- config gsm
- config region
- config sdb
- config service
- config shard
- config shardgroup
- config shardspace
- config table family
- config vncr
- configure
- connect

- create shard
- create shardcatalog
- delete catalog
- deploy
- disable service
- enable service
- modify catalog
- modify cdb
- modify credential
- modify file
- modify gsm
- modify region
- modify service
- modify shard
- modify shardgroup
- modify shardspace
- move chunk
- relocate service
- remove cdb
- remove credential
- remove file
- remove gsm
- remove invitednode (remove invitedsubnet)
- remove region
- remove service
- remove shard
- remove shardgroup
- remove shardspace
- services
- set gsm
- set inbound_connect_level
- set log_level

- set outbound_connect_level
- set trace_level
- split chunk
- sql
- start gsm
- start service
- status gsm
- status service
- stop gsm
- stop service
- validate catalog