

Oracle Globally Distributed Database 23ai によるJSON全文検索

Oracle Cloud Infrastructure Base Database Serviceを使用

2024年8月、バージョン[1.0]

Copyright © 2024, Oracle and/or its affiliates

公開

本書の目的

本書では、Oracle Database Release 23aiに含まれる特定のユースケースと一連の機能の詳細について説明します。本書は、23aiへのアップグレードに関するビジネス上の利点の評価と、説明した製品機能の実装およびアップグレードの計画を支援することのみを目的としています。

免責事項

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

本書は情報提供のみを目的としたものであり、ここで説明する製品の機能を実装およびアップグレードする際の資料として使用されることを意図しています。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないでください。本文書に記載されている機能の開発、リリース、時期および価格については、弊社の裁量により決定されます。製品アーキテクチャの性質上、本書に記述されているすべての機能を安全に組み込むことができず、コードの不安定化という深刻なリスクを伴う場合があります。

目次

はじめに	5
テクノロジー	5
Oracle DatabaseによるJSON（JavaScript Object Notation）のサポート	5
Oracle Text	5
Oracle Globally Distributed Database	6
ソリューションの概要	7
ソリューションの計画	7
サンプル・アプリケーションの要件	8
初期JSONデータセット	8
スキーマと問合せを設計するためのJSONデータの分析	9
スキーマ設計	10
JSON列	11
仮想列	11
シャード・データの分散方法	11
シャーディング・キー	11
スキーマ構造	12
データベース・スキーマの詳細	12
シャード・データベース・トポロジの計画	13
Globally Distributed Databaseのコンポーネント	13
環境の構築	14
シャードおよびシャード・カタログ用のデータベース・インフラストラクチャのデプロイ	14
シャード・ディレクタ用のコンピュート・インフラストラクチャのデプロイ	15
シャード・ディレクタ・ホストへのGlobal Data Servicesのインストール	15
接続性の検証	16
データベース・インスタンスの構成とチューニング	16
パラメータのチューニングとイベントのトレース	16
データベース・ユーザー・アカウントとパスワード・ポリシー	17
各シャード・データベースの検証	17
OCIでのシャーディング用のTLSを使用したTCPSの構成	18
シャード・ディレクタでのOracle Globally Distributed Databaseトポロジの構成	18
シャード・カタログの作成	19
シャード・ディレクタの追加と起動	19
ホスト・メタデータの追加	20
シャードグループの追加	21
シャーディング・トポロジの検証	21
シャードCDBの追加	23
シャードPDBの追加	24
シャード構成のデプロイ	25
Global Database Servicesの作成と開始	26

シャード・データベース・スキーマの作成	28
JSONデータのバルク・ロード	34
JSONデータファイルのデプロイ	34
JSONドキュメントのロード	35
統計の収集	36
シャードJSONデータの索引付け	37
主キーの索引の作成	38
JSON配列値の複数値索引の作成	38
Oracle Text索引の作成	38
Oracle Text索引のステータスの検証	39
検索問合せの設計と例	43
検索フォームの例	43
ユースケース	45
問合せの設計	45
SQL問合せの概要	46
バインド変数	46
検索結果の問合せ	47
カテゴリ・ファセット問合せ	52
言語ファセット問合せ	52
まとめ	53
付録A：高可用性とディザスタ・リカバリ	55
付録B：シャード・データの分散方法	56

図表一覧

図1 - 検索フォームのユーザー・インタフェースの概要	8
図2 - 物理インフラストラクチャの概要 - システム管理のシャード・データベースのデプロイメント	13
図3 - ファセット検索フォームのUIモックアップの詳細ビュー	44

はじめに

本書は、Oracle 23ai Globally Distributed Database環境での動作が保証された、JSONドキュメント用のスケーラブルな分散型全文検索ソリューションの設計とデプロイの手引きです。

Oracle Globally Distributed Databaseを全文検索のリポジトリおよびエンジンとして使用する利点は、次のとおりです。

- Oracle Databaseは、Oracle Textを使用することで、テキスト検索機能を必要とするデータの記録ソースの役割を果たし続けることができます。そのためにサードパーティの検索プラットフォームでデータを抽出して継続的に調整する必要はありません。
- サードパーティのJSONベースの検索索引を使用しているオラクルのお客様は、それらの索引をOracle Databaseに移行することで、全体的なコストを削減できる可能性があります。
- Oracle Globally Distributed Databaseは、ドキュメント・コレクションやアプリケーション/サービスのパフォーマンス・ニーズに合わせて拡張できる、分散型の水平スケーラブルなデータベース・プラットフォームを提供します。

これらすべては、Oracle DatabaseのネイティブJSONサポートを、Oracle Textと組み合わせて索引付けや検索の機能を実現し、OCI上のOracle Globally Distributed Databaseと組み合わせて水平スケーラビリティを実現し、データ主権の要件を満たすことで実現できます。

テクノロジー

このソリューションは、3つのオラクル・テクノロジーを組み合わせています。それらはOracle Globally Distributed Database、Oracle Text、およびオラクルのネイティブなJSONサポートです。

Oracle DatabaseによるJSON (JavaScript Object Notation) のサポート

Oracle Databaseは、SQLやリレーショナル・データベースの利点をJSONデータに提供し、他のタイプのデータベース・データと同じ方法および信頼性でJSONドキュメントを保存および操作できるようにします。

Oracle Databaseは、トランザクション、索引付け、宣言的問合せ、ビューなどのリレーショナル・データベース機能によってJSONをネイティブにサポートします。Oracle DatabaseによるJSONのサポートは、リレーショナル・ストレージの領域とJSONデータの問合せの領域が最適な組合せとなるようにし、リレーショナル問合せとJSON問合せが適切に連携できるようにします。オラクルのSQL/JSONサポートは、SQL標準のJSONサポートとも密接に連携しています。

ビジネス・ドキュメントは、リレーショナル表にJSONドキュメントとして保存できます。各行はドキュメントを表し、ドキュメント内のコンテンツはJSONデータタイプの列に保存されます。

参考資料：

- [23aiにおけるJSONドキュメントのシャーディングの概要](#)
- [23aiドキュメント：JSON開発者ガイド](#)
- [新しいJSONデータタイプに関するオラクルのブログ記事](#)
- [JSONドキュメントをシャード環境に保存する方法に関するオラクルのブログ記事](#)

Oracle Text

外部テキスト・エンジンは広く利用可能ですが、そのほとんどはファイルベースのテキスト・ストアでのみ機能します。Oracle Databaseは、Oracle Database内外のテキスト・ストアをサポートする、統合されたテキスト索引付けおよび検索エンジンを提供します。まず、検索するテキストがデータベース内にある場合、Oracle Database内の統合テキスト・エンジンを使用することはいくつかの利点があります。この場合、開発者またはアプリケーション所有者は以下を利用できます。

- すべてのデータ（テキスト・データおよび構造化データ）用の、2つではなく1つのリポジトリ。これによりメンテナンスやバックアップなどが簡素化されます。
- 同じリポジトリ内の索引。これにより、テキスト問合せおよび複合問合せが効率的に処理されるようになります。
- アプリケーション開発のための単一のAPI。オラクルのコンバインド・データベースを活用し、テキスト検索をデータベース内のその他の処理（空間、JSON処理、グラフなど）と組み合わせます。
- Oracle SQL実行エンジンおよび問合せプラン・オプティマイザとの統合。文全体を対象としたもっとも効率的な実行計画をオラクルに決定させます。
- Oracle Textは、Oracle Databaseのすべてのエディションに組み込まれています。テキストはデータベースに保存されるため、統合された包括的なデータ・ビューが得られます。さらに、データベースによって、テキストと構造コンテンツの両方を含む問合せを実行する最速の計画が選択されます。

今回のユースケースの観点では、Oracle TextはJSONドキュメントをサポートします。これは、Oracle Textがドキュメントの構造を理解し、解析できることを意味します。Oracle Textでは、未処理テキストだけでなく、各行のJSON列データに含まれるJSON要素の名前と値を使用できます。Oracle Textは、検索結果をスコアリングしてランク付けし、ドキュメント・コンテンツの“スニペット”をHTMLでレンダリングする機能を提供します。また、問合せの結果がアプリケーションによって表示用に処理されるときに、語句が強調表示されてコンテキストと関連性が示されます。検索問合せのこの側面やその他のリレーショナルな側面により、アプリケーションに返されるデータを最適化し、必要なアプリケーションベースの処理を最小限に抑えることができます。

参考資料：

- [技術概要：Oracle DatabaseのOracle Textの新しいユーザー・ガイド](#)
- [23ai Oracle Textアプリケーション開発者ガイド](#)

Oracle Globally Distributed Database

Oracle Globally Distributed Databaseを使用すると、ハイパースケールのグローバル分散コンバインド・データベースのデプロイが可能になります。これは、直線的なスケーラビリティ、柔軟性、障害分離と、データ主権のためのデータの地理的分散に関するアプリケーション要件をサポートする、Oracle Databaseの機能です。Oracle Databaseは、データセットのチャンクを独立したOracleデータベース（シャード）全体に分散させることによって、これらの機能を実装します。シャードはクラウドまたはオンプレミスにデプロイでき、特殊なハードウェアやソフトウェアは必要ありません。

アプリケーションは、シャードのプールを単一の論理データベースとして認識します。単一の論理データベースは、シャード・データベースと呼ばれます。ビジネス・ソリューションは、シャードをシャード・データベースに追加するだけで、どのプラットフォーム上でも、どのようなレベルにも、そのデータ層（データ、トランザクション、およびユーザー）を柔軟に拡張できます。データとワークロードは、アプリケーション層に対して透過的に、シャード全体で自動的にバランシングされます。

Oracle Globally Distributed Databaseを使用したシャーディングの利点：

- **直線的なスケーラビリティ：** Globally Distributed Databaseは、パフォーマンスのボトルネックを解消するとともに、シャードを追加することでパフォーマンスと容量の直線的なスケーリングを可能にします。
- **高可用性と障害の封じ込め：** Globally Distributed Databaseはシェアードナッシング・ハードウェア・インフラストラクチャであり、強力な障害分離を実現し、共有ディスク、SAN、クラスタウェアなどの単一障害点となる可能性のある要件を排除します。これにより、1つのシャードでの障害やパフォーマンス低下が、他のシャードのパフォーマンスや可用性に影響を与えることがなくなります。
- **地理的分散によるデータ主権：** 特定のデータをコンシューマの近くに保存できるようにすると同時に、保管データを特定の管轄区域内に保存する必要がある場合の規制要件も満たします。

Oracle Globally Distributed Databaseの機能と利点の詳細については、[Oracle.com の「Globally Distributed Database」のページ](#) および [23ai ドキュメント](#) を参照してください。

本書で示すソリューションでは、“システム管理のシャーディング”方式を使用して、シャード全体でデータを均等に分散します。さまざまな要件を満たすために、シャード表データをシャード・データベース全体に分散させる場合に使用できる他のデータ分散方式として、ユーザー定義のシャーディング、ディレクトリベースのシャーディング、コンポジット・シャーディングがあります。これらの方法は、データ常駐や他のビジネス要件および規制要件をサポートするなど、より明確な目的でデータセットをシャーディングする必要がある場合に機能を追加します。さらなる詳細については、「付録B：シャード・データの分散方法」を参照してください。

ソリューションの概要

設計と実装戦略のインサイト、例、プロセスの各手順の重要な側面など、JSON全文検索ソリューションのデータベースを中心とした側面について、順を追って説明します。

ソリューション全体は、基本的に以下のことが含まれ、デプロイは複雑ではありません。

- 1) JSONとリレーショナルの両方の力を活用する適切な組合せを決定する
- 2) Oracle Globally Distributed Databaseトポロジをデプロイする
- 3) 適切なスキーマと索引を作成する
- 4) ユーザー検索に使用できるテンプレート問合せを構築する

本書では、各手順について順を追って詳細に説明し、選択可能であり、参考として利用できるさまざまな決定を強調して示します。検索アプリケーション全体の設計について、順を追って説明します。また、後で診断に役立てたり、単に情報を示したりするためにも使用できる、多くの検証手順も含まれています。

この説明には次の内容が含まれます。

1. 計画
 - a. アプリケーション要件を確認する。
 - b. JSONデータセットを準備して分析する。
2. 構築
 - a. OCI 23ai Base Database Serviceをデプロイする。
 - b. Oracle Globally Distributed Databaseトポロジを構成する。
 - c. スキーマ・ユーザー、表領域、表ファミリーを作成する。
3. ロード
 - a. JSONデータを各シャードの表ファミリーに並列でロードする。
 - b. 統計を収集する。
4. 索引
 - a. 主キーの索引を作成する。
 - b. JSON配列データの複数値索引を作成する。
 - c. Oracle Textプリファレンスを構成する。
 - d. JSON列にOracle Text索引を作成する。
5. 検索
 - a. アプリケーションのユースケース・シナリオと、検索フォームのユーザー・エクスペリエンスを確認する。
 - b. 問合せの設計要素と設計要件について検討する。
 - c. SQLの例を確認する。

ソリューションの計画

この技術概要では、JSONデータセットを分析して、スキーマ設計に情報を提供し、Oracle Databaseにロードし、索引付けし、全文検索問合せの基礎として使用する方法を示します。

スキーマには単一の表を使用することを選択しています。公開されているWikipedia記事データをJSON形式に変換してから、スキーマにバルク・ロードします。WikipediaのWikiCommonsスキーマには、Wikipediaアプリケーションの完全な機能に関連する、いくつかの表と利用可能なデータファイルがあります。全文検索の索引付けと問合せのユースケースを示すために必要なのは、記事データのコーパス（JSON形式）だけです。今回の例の表には、リレーショナル問合せ演算子や検索条件で直接使用できるように、JSON列データの値を自動的に解析して公開するためのいくつかの仮想列も含まれています。

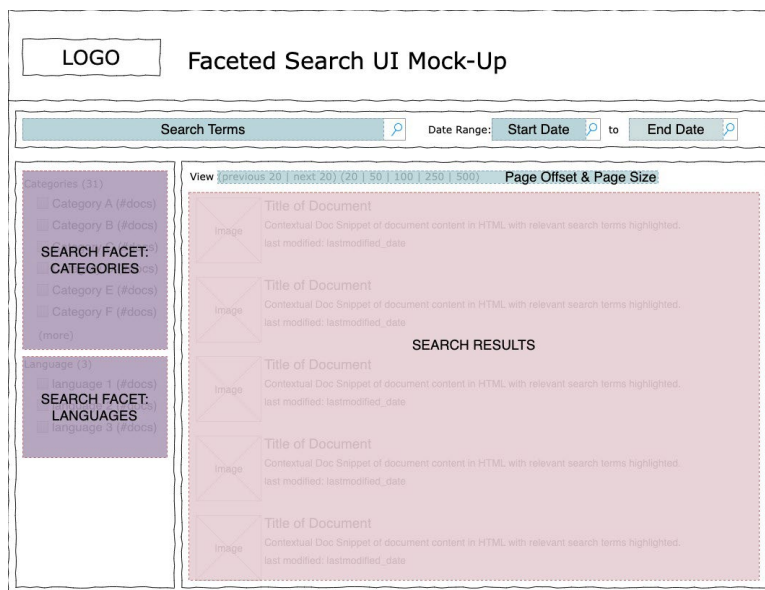
サンプル・アプリケーションの要件

今回の例の検索フォームのユーザー・インタフェースは、Wikipediaの高度な検索オプションのユーザー・インタフェースとは異なります。今回の例では、検索する語句のフィールドに加え、開始日と終了日を含めて、ユーザーが検索の期間を限定できるようにします。ユーザーがこれらのフィールドのいずれかを使用して検索を実行すると、検索結果に結果の最初のページが移入されます。ユーザーは、結果の数と、表示するページをカスタマイズできます。これはすべて、検索のユーザー・エクスペリエンスの標準的な内容です。

検索結果に加えて、特定のデータ・ディメンションごとに一致するドキュメントの数を集計するために実行される、2つの検索問合せがあります。これらは口語表現的に“ファセット”と呼ばれ、ユーザーがパラメトリックな方法で検索結果をさらに絞り込むために選択できる、これらのディメンションに沿って値を提示できます。

今回の例では、データの2つのディメンション、すなわちファセットである、ドキュメントを書くために使用された言語と、作成者が各ドキュメントに指定した“カテゴリ”を使用しています。今回のユーザー・インタフェースにおいて、これら2つのファセットは、左側のナビゲーション・パネルに検索結果と並べて表示されます。各ファセットのデータには、ディメンションで見つかったすべての値と、その値を持つドキュメントの数が含まれます。

図1：検索フォームのユーザー・インタフェースの概要



このソリューションには3つの問合せがあります。最初の問合せにより、検索結果パネルにデータが移入されます。2番目と3番目の問合せは、2つのファセットを各ファセットの値のチェック・ボックス・グループとして表示するために必要なデータと、それぞれのドキュメントの数を生成します。リレーショナル・データを使用する場合、これらの問合せの設計は学術的なものになります。JSONデータベース機能とOracle Textを使用してこれを効率的に行うことが、本書の要点です。これらの問合せを実行するために必要な機能と設計を紹介します。

JSONデータセット

今回の例では、公開されているWikipedia記事の英語データセット（“enwiki”）のJSON表現を使用します。Wikipediaを選択したのは、公開されているデータセットと、私たちがまさに関心を持っている、全文検索のユースケースの両方を提供しているからです。元々はJSON形式ではありませんでしたが、今回の例で使用するためにenwiki記事のデータファイルをJSONに変換しました。最新の“enwiki”データ・ダンプ・ファイル（XML形式）は、<https://dumps.wikimedia.org/enwiki/latest/>からダウンロードできます。

WikiCommonsスキーマの多くと各記事のJSONドキュメントの一部は、今回の全文検索ソリューションのデモには関係ありません。変換されたJSONデータファイルは、各レコードから不要なJSON要素を削除するように変更されました。

以下に、データセット内のJSON要素を表す、短い初歩的なドキュメント例を示します。今回の例でロードするデータファイルは、これと同様のドキュメントで構成されていますが、Wikipediaの実際の内容とメタデータ値が含まれています。

その後、次のセクションでは、JSONドキュメントのさまざまな要素について説明し、今回の全文検索のユースケースで効率的なSQL問合せを実現するためにオラクルのJSONデータベース機能を最大限に活用する方法の例をいくつか説明します。

Wikipediaのenwikiの記事：人為的な例（表示用に書式を変更）

```
{
  "title": "Oracle Demo Intro Article ",
  "wikiTitle": "ORACLE_DEMO_INTRO_ARTICLE",
  "wid": 999019999,
  "lang": "EN",
  "timestamp": "2022-06-21T05:22:00Z",
  "paragraphs": [
    "This is the first paragraph of the MAA Demo Intro Article test document for database sharding. This is the second sentence of the first paragraph",
    "This is the second paragraph. There are multiple lines each with multiple sentences.\nA second line in the second paragraph.",
    "With Regards to the Data structure of each document:\n",
    "The .wid element is unique per article and extracted for use as the shard key and primary key in the Articles table.",
    "There may be hundreds of strings in the .paragraphs array with up to 4kb per string.",
    "The longest article is over 440kb.",
    "The .categories array .id element contains the actual category term with spaces replaced with underscore and prefixed with the phrase: Category:. \n The .templates array contains various metadata about the article with each .name element being the key and corresponding .description array containing the value."
  ],
  "categories": [
    {
      "id": "Category:Database_Sharding",
      "anchor": "database sharding", "start": 71, "end": 88, "type": "CATEGORY"
    },
    {
      "id": "Category:Oracle_Demo_Article",
      "anchor": "", "start": 0, "end": 0, "type": "CATEGORY"
    },
    {
      "id": "Category:Facet B",
      "anchor": "", "start": 0, "end": 0, "type": "CATEGORY"
    }
  ]
}
```

スキーマと問合せを設計するためのJSONデータの分析

データセット内の各JSONフィールドは、関連付けられたデータベース表の構築にどのように使用できるかを評価する必要があります。パフォーマンス上の理由から、またはアプリケーションのSQLコーディング要件を簡素化するために、索引または仮想列の追加が適切である可能性がある問合せで頻繁に使用されるであろうデータを含む他のJSON要素を特定します。

サンプル・データセットの各JSON要素の分析は次のとおりです。

wid

このJSON要素には、1〜10桁の数字で構成される、各記事の一意のIDが含まれます。この値をArticle_ID列の基礎として使用します。この列は、Articles表の主キーおよび表ファミリーのシャーディング・キーになります。

Title

これは、判読可能な、記事のタイトルです。タイトルは、検索し、かつ検索結果に効率的に表示したい値の1つです。仮想列を実装すると、すべての問合せでJSON要素値を解析するためにJSON_VALUE()関数呼出しを追加するという複雑さなしに、この値をSELECT句に含めることができます。検索問合せフィルタの条件の観点から、JSON列を直接使用するJSON_TEXTCONTAINS()関数を用いて、Oracle Text索引を使用した全文検索を実行します。

Paragraph[]配列データ

JSONドキュメントのparagraph配列には、カンマで区切られた引用符付き文字列が含まれています。これは記事コンテンツの大部分を構成し、リクエストされたキーワード文字列検索条件のプライマリ・ソース・データ・フィールドとして、メインの検索結果問合せのタイトル (Title) とともに使用されます。paragraph配列は、JSON_TEXTCONTAINS()関数を使用して検索され、この関数の実行は、使用可能なOracle Text検索索引を使用するように最適化されます。paragraph JSON要素値は、メインの検索結果問合せのSELECT句のCTX_DOC.POLICY_SNIPPET()というOracle Text関数でも使用され、検索語句が強調表示された、文脈的に関連するドキュメント・テキストの短いセクションがHTML形式で返されます。

Categories[]配列データ

Wikipediaでは、記事へのカテゴリの割当ては多対多の関係です。

カテゴリ・データは、JSON配列ベースの値を使用したファセット検索集計問合せの実行方法を示す便利な例として機能します。全文検索のユーザー・エクスペリエンスにおいて、ファセット検索機能の提示には、意味のあるグループ化を提供する、関連付けられた値に基づく結果の集計または数の問合せが含まれます。これらのグループは“バケット”と呼ばれ、関連付けられたデータまたはメタデータの任意の部分のバケットのコレクションは“ファセット”と呼ばれます。通常、ユーザーには1つ以上のファセットのバケットのリストが表示され、検索結果のフィルタとして使用する、バケットのサブセットを選択することで検索パラメータを絞り込むことができます。

Lang

このJSON要素には、2文字で表す、地域言語名のISO 639-1ショート・コードが含まれています。

単純なリレーショナル検索条件の使用と、これらの値の効率的な問合せを可能にするために、リレーショナル・スキーマにこの要素の仮想列を実装することを選択しました。これにより、アプリケーション・コードでJSON関数呼出しを繰り返して使用しなくてもよくなり、問合せ構文が簡素化され、データベースがSQL実行を効率的に最適化できるようになります。

Timestamp

この要素には、各記事ドキュメントの最終変更日の文字列が含まれます。記事データの“timestamp”の形式は、オラクルのタイムスタンプ形式とそのまま一致するわけではありませんが、timestamp_tzデータタイプに簡単にキャストできます。

これは、仮想列がもたらす効率の絶好の例です。データタイプ変換は、スキーマ定義で1回構成するだけで済み、最終変更タイムスタンプ値を必要とするクライアント側の問合せごとに構成する必要はありません。

スキーマ設計

オラクルのコンバード・データベースを使用すると、Oracle TextおよびJSONデータベース関数を介して、データへのリレーショナル・アクセス、索引付け、検索が可能になります。スキーマ設計を検討する際には、これらすべてのアクセス方法を考慮に入れることができます。これによりお客様には、リレーショナル・データベースの機能を利用できるだけでなく、すでにJSON形式であるデータを利用および維持できるという利点があります。

JSON列

Oracleでは、JSONをデータベース表のJSONデータタイプ列内に格納します。この列内のJSONドキュメントは完全に索引付け可能かつ検索可能です。標準SQL問合せには、JSON関数を使用してJSONドキュメント内で検索を直接実行するフィルタ条件を含めることができます。

もっとも単純なケースでは、すべてのJSONレコードが大きな表内の個別の行としてデータベースに保存されます。各表はシャード全体でパーティション化されていますが、1つの論理表のままです。今回の例のスキーマは、後述の「スキーマ構造」セクションで説明されている単一の表で構成されています。

仮想列

これらは、他の列または固定式から導出された値を持つ列です。仮想列の利点は、表内のJSONフィールドの式の評価から導出された列として、リレーショナル表に列を作成できることです。今回の例の表における列の多くは、仮想列です。

たとえば、ドキュメントのタイトル（Title）は、JSON列に保存されるJSONドキュメント内の要素として保存できます。タイトル・データは、JSON要素値から自動的に導出される値を使用して、（たいていは“title”という名前の）標準リレーショナル列に表示することもできます。利点は、すべての問合せではなく、スキーマ・レベルで抽象化された、JSON固有の関数呼出しの詳細を用いて、純粋なリレーショナル式でこの仮想列を使用できることです。

シャード・データの分散方法

トポロジ内のシャード・データベース全体でデータをパーティション化するには、複数の方法があります。これは、データの分散方法を決定するため、データ設計に関連します。

この技術概要の例では、システム管理のデータ分散方法が使用されます。

システム管理の方法では、列を“シャーディング・キー”として宣言することのみが必要です。さらに、ユーザーはシャードへのデータのマッピングを指定する必要はありません。データは、シャード・キーの値のコンシステント・ハッシュを使用したパーティション化により、自動的にシャード全体に分散されます。パーティション化・アルゴリズムにより、各シャード間に均等かつランダムにデータが分散されます。システム管理の方法で 사용되는分散は、ホット・スポットを排除し、シャード全体で均一なパフォーマンスを提供し、シャード・データベースでシャードの追加または削除が行われたときにバランスのとれたデータ分散を自動的に維持することを目的としているため、適切な名前が付けられます。

今回のユースケースは、次のようなものです。

- データとユースケースには、特定のレコードを同じシャードに関連付けてグループ化するための特定のビジネス要件や技術要件はない。
- 各ドキュメント・レコードは、他のドキュメント・レコードから独立している。
- パフォーマンスを向上させるためにシャード・データベースをスケール・アウトする必要がある場合は、すべてのドキュメント・レコードを他のシャードに再シャードできる。

何らかの相関関係がある場合（共通のアカウントに複数の販売請求書がある場合や、データ主権の要件のため、国コードやその他の地域フィールドの値に基づいてレコードを一緒にシャーディングする必要がある場合など）は、他のシャーディング方法の方が適切な場合があります。

シャーディング・キー

シャード・データベースの中心にあるのは、シャーディング・キーの概念です。多くの場合、これはもちろんレコード識別子にマッピングできますが、シャーディング・キーとレコードの間に正式な関係が存在する必要はありません。

シャーディングの利点を得るには、単一のシャードで処理されるデータベース・リクエストの数を最大化するように、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。シャーディング・キーとして使用するフィールドの選択は、シャード・データベース全体でドキュメントを整理するうえで重要です。

システム管理のデータ分散を使用する場合、シャーディング・キーは厳密に一意である必要はなく、必ずしも主キーである必要もありません。ただし、関連性のないレコードを一意のドキュメントIDで整理する場合は、主キーをシャーディング・キーとして使用することをお勧めします。

詳細については、Oracle Globally Distributed Databaseドキュメントの「[シャーディング・キー](#)」セクションを参照してください。

オンライン・トランザクション処理のシナリオにおいて、問合せと挿入/更新/削除の操作は既知の主キーを使用して、その結果、既知のシャーディング・キーも使用して、実行されます。これらの操作は、Oracleのドライバが使用されている場合、クライアント側のデータベース・ドライバにより、シャーディング・キーに基づいて、適切な単一のシャードに限定して送信することができます。

今回の例で使用されているドキュメント検索のユースケースでは、検索の送信時にシャーディング・キーは不明であり、検索対象のキーワード/全文のみがわかっています。検索結果で返されたドキュメントをユーザーがクリックすると、アプリケーションはその一意のドキュメント・リクエストのシャーディング・キー・データとしてドキュメントIDを取得し、正しいシャードからドキュメントを直接取得できます。

以下は、シャーディング・キーを選択する際のいくつかの推奨事項です。

- シャーディング・キーにはテキスト文字列または数値を使用できますが、ドキュメントを一意に識別できるようにシャーディング・キーを一意にすることをお勧めします。
- シャーディング・キーはめったに、または一切変更すべきではありません。ドキュメントに常に関連付けられる固定IDを使用するのが最善です。

スキーマ構造

同じ方法でシャード/分散される、関連する複数の表は、シャード表ファミリーと呼ばれます。シャード・データベースには複数の表ファミリーを含めることができます。すべてのシャードにわたって各シャードが同じ表コンテンツを持つ“複製表”も存在でき、この表は通常、“ルックアップ”表に使用されます。シャード表はシャード全体で水平にパーティション化され、複製表はすべてのシャードに完全にレプリケートされます。

詳細な説明については、[シャード表ファミリー](#)および[複製表](#)のドキュメントを参照してください。

データベース・スキーマの詳細

今回の例では、シャード表ファミリーの親表として、“Articles”という名前の単一の表を実装します。Articles表には、主キーおよびシャーディング・キーとしてのID列、Article_JSON列、およびArticles_JSON列の内容に基づく4つの仮想列が含まれています。各列の設計意図を以下にまとめています。

Article_ID

これは主キーであり、シャーディング・キーとして使用されます。データタイプはNUMBER(10)です。Article_IDには、\$.wid JSON要素から抽出されたスカラー値が含まれます。この値は、データがJSONファイルの外部表からロードされるときや、単一レコードの挿入/更新操作時に、逐次的に抽出および挿入される必要があります。主キー列を仮想列にすることはできません。

Article_JSON

Article_JSON列のデータタイプはJSONで、データ・ロード時にJSONファイルの外部表から抽出されたものとして、またはアプリケーションによって挿入/更新されたものとして逐次的に挿入されたJSONドキュメントが含まれます。

Title

TitleはデータタイプがVARCHAR2(512)の仮想列で、Articles_JSONの\$.titleのJSON_VALUE()を評価します。この列には値が動的に移入されるため、値を明示的に挿入することはできません。

Lang

これはVARCHAR2(2)データタイプの仮想列で、Articles_JSONの\$.langのJSON_VALUE()を評価します。この値には、ISO規格による、2文字の国コードが含まれます。この列には値が動的に移入されるため、値を明示的に挿入することはできません。

Last_Modified

Articles_JSONの\$.timestamp要素のJSON_VALUE()を文字列として問い合わせ、それをTIMESTAMP(0) WITH TIMEZONEデータタイプに変換する仮想列。この列には値が動的に移入されるため、値を明示的に挿入することはできません。

後のセクションで作成する、今回の例のArticle表は、次のように記述されます。

SQL> desc Articles

Name	Null?	Type
ARTICLE_JSON	NOT NULL	JSON
ARTICLE_ID	NOT NULL	VARCHAR2(10)
DATE_LOADED	NOT NULL	TIMESTAMP(6)
TITLE		VARCHAR2(512)
WIKI_TITLE		VARCHAR2(512)
LANG		VARCHAR2(2)
LAST_MODIFIED		TIMESTAMP(0) WITH TIME ZONE

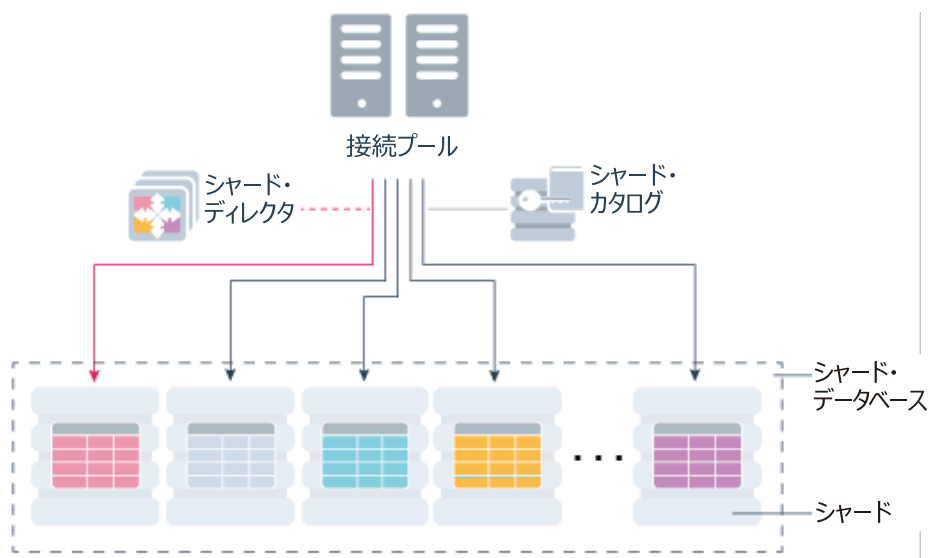
シャード・データベース・トポロジの計画

各シャード・データベースのトポロジとアーキテクチャの計画は、他のデータベース・ソリューションと同様に、課せられる特定のビジネス要件、技術要件、および規制要件に基づいて、ソリューションごとに、また環境間で異なります。Oracle Globally Distributed Databaseでは、方程式で考慮に入れるべき要素がさらにいくつかあります。

JSON全文検索の索引付けと問合せの機能に焦点を当てた概念実証（POC）というこの技術概要の目的を考慮すれば、厳密なエンタープライズ実装に必要な高可用性とディザスタ・リカバリの側面は、いずれも本書の範囲外であり、取り上げていません。オラクルのドキュメントへのリンクは、本書の最後にある付録の「参考資料」にあります。

シャード・データベース・ソリューションのPOC環境をデプロイするためのもっとも単純なトポロジは、1つのシャード・カタログ・データベース（シングル・インスタンス）、1つのシャード・ディレクタ、データ・パイロードおよび索引の一部を含めるのに十分な容量を持ついくつかのシャード・データベース（シングル・インスタンス）で構成されます。

図2：物理インフラストラクチャの概要 - システム管理のシャード・データベースのデプロイメント



Globally Distributed Databaseのコンポーネント

シャード

個々のシャードには、データセット内のレコードの一部が保持されます。構成されたシャードの数は、各シャードにかかる全体的な負荷に直接影響します。今回のユースケース例で推奨される、システム管理のデータ分散方法では、レコード識別子のハッシュ（シャード・キー）を使用して、各行を特定のシャードに割り当てます。これにより、データのスケラブルで均等な分散が保証されます。

シャード・ディレクタ

シャード・ディレクタはステートレスで軽量なコンポーネントであり、シャーディング・キーまたはレコード・キーに基づいてクライアント・アプリケーションに問合せリダイレクション機能を提供します。このシャーディング・キー値を使用して、シャード・ディレクタは、適切なシャード・データベースへの接続リダイレクト・メッセージでクライアント・アプリケーションに応答します。これは、アプリケーションが検索の結果として、または読取りや更新のためにドキュメントを取得することが必要となった時点で、ドキュメントを直接取得する必要がある場合に使用されます。

シャード・カタログ

シャード・カタログ・データベースの複数シャード問合せコーディネータ機能は、複数のシャードにわたる検索を要する問合せに使用されます。コーディネータは、これらの複数シャード問合せをすべてのシャードに分散し、結果を取得して結合する役割を担います。シャード・カタログ・データベースは、シャード・データベース構成メタデータのリポジトリとしても機能します。

環境の構築

Oracle Globally Distributed Databaseは、オンプレミスまたはクラウドにデプロイできます。現在、この技術概要のソリューションに必要な機能を備えたデータベースの最新バージョンは、OCI Base Database Serviceとしてデプロイできます。シャード・ディレクタは、OCIコンピュート・インスタンスとしてデプロイできます。

オンプレミスにデプロイする場合、Oracle Databaseにおける従来のハードウェア、ネットワーク、オペレーティング・システム、およびソフトウェア要件とチューニングが通常どおり適用されます。Oracle Globally Distributed Databaseのインストールに特に適用される、注意すべきチューニング仕様がいくつかあります。詳細な仕様については、『Oracle Globally Distributed Databaseガイド』の「[Provision and Configure Hosts and Operating Systems](#)」セクションを確認してください。

ここでは、オンプレミスで、またはOCI Base Database Serviceを使用してのいずれかで、Globally Distributed Databaseを手動で作成するための正確な手順を示していることに注意してください。OCIでMarketPlaceを使用して環境を作成するなど、自動化ソリューションを使用する場合は、本書のデータベース・スキーマの作成に関するセクション（28ページ）までスキップしてください。

シャードおよびシャード・カタログ用のデータベース・インフラストラクチャのデプロイ

現在、Oracle Database 23aiインスタンスは、OCI Base Database Serviceを使用してデプロイできます。23aiデータベース・バージョンを選択して、実装に必要な仕様を持つ、シャード・カタログおよび各シャード用の新しいベース・データベース・サービスを作成します。これは、OCIコンソールまたはOCIコマンドライン・インタフェース（OCI CLI）を使用して実行できます。

シャード・カタログ用のベース・データベース・サービスを作成するOCI CLIコマンドの例：

注：この例には、自動バックアップを構成するためのパラメータは含まれていません。すべてのパラメータ値は環境に合わせて調整する必要があり、そのままでは適切ではない場合があります。

```
[user@laptop ~/dev/oci]$ oci db system launch --display-name "w23catyyz10" \
--hostname w23catyyz10 \
--db-name catcdb \
--pdb-name catpdb \
--region ca-toronto-1 \
--compartment-id ocid1.compartment.oc1..aaaaaaaarc... \
--subnet-id ocid1.subnet.oc1.ca-toronto-1..aaaaaaa3q... \
--availability-domain efXT:CA-TORONTO-1-AD-1 \
--fault-domains ["FAULT-DOMAIN-1"] \
--shape VM.Standard.E4.Flex \
--node-count 1 \
--cpu-core-count 4 \
--initial-data-storage-size-in-gb 2048 \
--disk-redundancy NORMAL \
--database-edition ENTERPRISE_EDITION_EXTREME_PERFORMANCE \
--storage-management LVM \
--license-model BRING_YOUR_OWN_LICENSE \
--db-version 23.4.0.24.05 \
--character-set AL32UTF8 \
--ncharacter-set AL16UTF16 \
--db-workload OLTP \
--admin-password sys_credential \
--ssh-authorized-keys-file /full/path/to/.ssh/desired_key.pub \
--is-incident-logs-enabled true \
--is-diagnostics-events-enabled true
```

シャード・ディレクタ用のコンピュート・インフラストラクチャのデプロイ

ほとんどすべてのLinuxベースのコンピュート・インスタンスは、パッケージ、カーネルのバージョン、物理ストレージ、およびメモリの要件が満たされている限り、シャード・ディレクタをホストするために使用できます。『Oracle Database Global Data Services概要および管理ガイド』の「[インストールの計画](#)」を参照してください。

シャード・ディレクタ・ホストへのGlobal Data Servicesのインストール

Oracle Global Data Services（Oracle GDS）グローバル・サービス・マネージャRelease 21.3.0.0を[Oracle eDelivery](#)からダウンロードし、プロンプトに従ってソフトウェア・メディアのルート・ディレクトリからOracle Universal Installerを使用してインストールするか、サイレント・インストールを実行します。グローバル・サービス・マネージャ（GSM）ソフトウェアのインストールについては、『[Oracle Database Global Data Services概要および管理ガイド](#)』を参照してください。

ローカルOSファイアウォールでGSMに必要なポートを開きます。これには、GSMインスタンスの作成時に指定されるポートが含まれます。次のデフォルト値は、必要に応じてカスタマイズできます。

- ローカル・リスナー：TCP/1522
- ローカルONS：TCP/6123
- リモートONS：TCP/6234

例：GSMポート用のOracle LinuxファイアウォールIDの構成

```
[opc@w23diryyz10 ~]$ sudo firewall-offline-cmd --port=1522:tcp
[opc@w23diryyz10 ~]$ sudo firewall-offline-cmd --port=6123:tcp
[opc@w23diryyz10 ~]$ sudo firewall-offline-cmd --port=6234:tcp
[opc@w23diryyz10 ~]$ sudo /bin/systemctl restart firewalld
```

接続性の検証

ネットワーク・トポロジとコンピュート・インスタンスのオペレーティング・システムで、シャード・ディレクタ・ホスト上のGSMインスタンスとシャード・データベース・サーバーおよびシャード・カタログ・データベース・サーバーとの間で必要とされる接続に、適切なネットワーク送受信が許可されていることを確認します。

OCIセキュリティ・リストまたはネットワーク・セキュリティ・グループは、正しく定義して適用する必要があります。GSMインスタンスのコンピュートOSファイアウォールは、必要なポートを許可するように構成する必要があります。すべてのシャード・ディレクタとデータベースには、完全修飾ドメイン名だけでなく短いホスト名を解決する機能も必要です。

データベースのTNS接続のテスト

シャード・ディレクタから、短いホスト名を使用し、シャード・データベースおよびシャード・カタログ・データベースに対してTNS pingのテストを実行します。また、同じ方法でシャード・カタログからシャードに対してTNS pingのテストを実行します。

例：tnspingコマンドによる接続のテスト

```
[oracle@w23diryya10 ~]$ tnsping w23shdyyz11:1521 | egrep 'msec|TNS-|Attempting'
Attempting to contact
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=))(ADDRESS=(PROTOCOL=tcp)(HOST=10.8.208.94)(PORT=1521)))
OK (0 msec)
```

データベース・インスタンスの構成とチューニング

汎用データベース・サービスを使用できる場合は、いくつかの構成を実行する必要があります。

パラメータのチューニングとイベントのトレース

OCI Base Database Systemに設定されているデフォルトのデータベース・パラメータのほとんどは、少なくとも研究開発の概念実証または開発環境での使用には十分です。ただし、一部のパラメータのデフォルト値は、シャーディング要件に応じるために特に調整する必要があります。

- OPEN_LINKS* – シャード・カタログとシャード上の一部のダイレクト・パス・ロード機能では、デフォルトを上回るようにこのパラメータを調整する必要があります。依存関係の複雑さを考慮して、開始値255または環境内のシャード数のいずれか大きい方を指定することをお勧めします。
- GLOBAL_NAMES – シャード・ディレクタ（GSM）インスタンス、シャード・カタログ、およびシャード間で意図した相互運用性を可能にするには、グローバル名をFALSEに設定する必要があります。
- DB_CREATE_FILE_DEST – 環境に合わせて有効な値に設定する必要があります。
- STANDBY_FILE_MANAGEMENT – GSM構成での警告を回避するには、AUTOに設定します。
- NLS_NCHAR_CHARACTERSET – すべてのシャードおよびシャード・カタログ・データベースにおいて同一である必要があります。
- NLS_CHARACTERSET – すべてのシャードおよびシャード・カタログ・データベースにおいて同一である必要があります。

いくつかのトレース・イベントを設定することも推奨されます。すべてのシャード・データベースおよびシャード・カタログ・データベースでPL/SQLトレースのロギングを許可するようにトレース・イベントを設定し、シャードに対して、パラレル問合せワーカーのエラーを問合せコーディネータに伝播するイベントを組み込み、Oracle Text索引付けでのパラレル・ダイレクト・パス・ロードを無効にします。

- PL/SQLトレースを有効にする
'10798 trace name context forever, level 7'
- ORA-12801 '10397 trace name context forever, level 1'
をスローする代わりに、パラレル問合せ（PQ）ワーカーのエラーを問合せコーディネータに伝播する
- Oracle Textで、索引付け時のパラレル・ダイレクト・パス・ロードを無効にする
'30580 trace name context forever, level 256'

DDLの例：23ai DBS CDBパラメータを調整する

```
[oracle@w23catyyz10 ~]$ sqlplus / as sysdba
SQL> ALTER SYSTEM SET MAX_DUMP_FILE_SIZE=UNLIMITED SCOPE=BOTH;
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT=AUTO SCOPE=BOTH;
SQL> ALTER SYSTEM SET GLOBAL_NAMES=FALSE SCOPE=BOTH;
SQL> ALTER SYSTEM SET OPEN_LINKS=255 SCOPE=SPFILE;
SQL> ALTER SYSTEM SET OPEN_LINKS_PER_INSTANCE=255 SCOPE=SPFILE;
```

DDLの例：シャード・カタログ23ai DBS CDBトレース・イベントをシャーディング用に調整する

```
SQL> ALTER SYSTEM SET EVENT='10798 TRACE NAME CONTEXT FOREVER, LEVEL 7',
COMMENT='PL/SQL Tracing for Sharding for Shard Catalog'
SCOPE=SPFILE;
```

DDLの例：シャード23ai DBS CDBトレース・イベントをシャーディングおよびOracle Text用に調整する

```
SQL> ALTER SYSTEM SET EVENT='10397 TRACE NAME CONTEXT FOREVER, LEVEL 1',
'10798 TRACE NAME CONTEXT FOREVER, LEVEL 7',
'30580 TRACE NAME CONTEXT FOREVER, LEVEL 256',
COMMENT='PL/SQL Tracing for Sharding plus: PQ Worker Error
Propagation, and disable Parallel Direct Path Load for Oracle Text Indexing on Shards only'
SCOPE=SPFILE;
```

CDBを変更してから再起動し、トレース・イベントが設定されていることを確認します。

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
SQL> CONNECT / AS SYSDBA
SQL> ORADEBUG SETMYPID
SQL> ORADEBUG EVENTDUMP system
10798 trace name context forever, level 7
30582 trace name context forever, level 1048576
```

データベース・ユーザー・アカウントとパスワード・ポリシー

シャード・ディレクタがデータベースに接続してシャード・データベースを管理できるように、シャード・カタログ・データベースとシャード・データベースで特定のアカウントを有効にする必要があります。ドキュメントの指示に従って、データベースと、指定したユーザー・アカウントをセットアップします。
「[シャード・カタログ・データベースの作成](#)」および「[シャード・データベースの作成](#)」を参照してください

各シャード・データベースの検証

シャード・データベースを完全に構成してチューニングしたら、それぞれのシャードPDBでdbms_gsm_fix.validateShardプロシージャを実行します。

注：Oracle Data Guardが必要でなく、dg_broker_startパラメータがfalseに設定されている場合、validateShardプロシージャでエラーがスローされます。Data Guardが使用されていない場合、このエラーは無視できます。

例：シャードPDB構成の検証

```
$ sqlplus / as sysdba
SQL> set serveroutput on
SQL> execute dbms_gsm_fix.validateShard;
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is SHDCDB0.
INFO: Database unique name is shdcdb0_3s2_yyz.
INFO: Database ID is 493557786.
INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
WARNING: Flashback is off.
INFO: Force logging is on.
INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is AL32UTF8. 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 SHDPDB0.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '/u02/app/oracle/oradata/shdcdb0_3s2_yyz/'
INFO: db_recovery_file_dest set to: '/u03/app/oracle/fast_recovery_area'
INFO: db_files=200. Must be greater than the number of chunks and/or tablespaces to be
created in the shard.
ERROR: dg_broker_start set to FALSE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '*', /u01/app/oracle/oradata/'
INFO: standby_file_management is set to AUTO.
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is '/u01/app/oracle/admin/ORCL/dpdump/16E9F79186922679E0631701F40AB0C5'.

PL/SQL procedure successfully completed.
```

OCIでのシャーディング用のTLSを使用したTCPSの構成

シャード・データベース・コンポーネント間の通信を保護するために、シャード・カタログとシャードへの（またはそれらの間の）すべての接続に、Oracle Databaseのネイティブ・ネットワーク暗号化を使用するか、またはTCPSプロトコルとTransport Layer Security (TLS) を使用することをお勧めします。OCI Base Database Serviceを使用してデプロイする場合は、「[Configure TCP/IP with SSL/TLS for Sharding – GSM OCI Mode \(Doc ID 2881390.1\)](#)」の情報をを使用してTLSを有効にしてください。

ウォレットの使用、透過的データ暗号化（TDE）の構成、およびその他のセキュリティ関連の概念と推奨事項に関する追加のガイダンスについては、「[Oracle Globally Distributed Database環境のセキュリティ](#)」ドキュメントを参照してください。

シャード・ディレクトラでのOracle Globally Distributed Databaseトポロジの構成

シャード・カタログおよびすべてのシャードのデータベースが、対応するTNSリスナーとともに構成されたら、シャード・ディレクトラからGDSCCTLを使用して、シャーディング・メタデータをシャード・カタログ・データベースに追加できます。シャーディング・メタデータは、シャード・データベースに使用されるトポロジを記述します。

高可用性のない、システム管理のシャード・データベース・トポロジの作成例について説明します。

これらの手順も、ドキュメントの「[Oracle Globally Distributed Database トポロジの構成](#)」セクションで説明されています。また、2つ目の地域データセンターでのOracle Active Data Guardスタンバイ・レプリカ・シャードによる高可用性を含む「[Oracle Globally Distributed Databaseのデプロイの例](#)」のセクションもあります。

構成プロセスの概要は次のとおりです。

1. シャード・カタログの作成
2. シャード・ディレクトの追加と起動
3. ホスト・メタデータの追加
4. プライマリ・シャードグループの追加
5. シャーディング・トポロジの検証
6. シャードCDBの追加
7. シャードPDBの追加
8. 空きDB_FILESの確認
9. シャード構成のデプロイ
10. グローバル・サービスの作成と開始
11. シャードのステータスの確認

シャード・カタログの作成

GDSCCTL CREATE SHARDCATALOGコマンドを使用して、シャード・データベース・トポロジを記述するメタデータをシャード・カタログ・データベースに作成します。

ベスト・プラクティスとして、AUTOVNCRパラメータを常にOFFに設定し（デフォルトはON）、後の手順で必要なホストとIPアドレスを“招待されたノード”として宣言的に指定することをお勧めします。

シャード・カタログ PDB の接続文字列の場合、文字列全体をパラメータ値としてインラインで指定することも、ORACLE_HOME/network/admin/gsm.oraファイルに追加された接続文字列のTNSエイリアスを指定することもできます。

Oracle Active Data Guardを使用して関連するスタンバイを提供する場合、この接続文字列でプライマリ・データベースとスタンバイ・データベースを指定する必要があります。『Oracle Database高可用性概要およびベスト・プラクティス』のドキュメントにある「[高可用性のための接続文字列の構成](#)」を参照してください。

今回の例では、レプリケーションのない、シングル・インスタンスのシャード・カタログ・データベースおよびシャード・データベースを使用して、単一リージョン内におけるシステム管理のデータ分散を使用します。実際的な例については、Oracle Shardingのドキュメントの「[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。

例 : gdsctl create shardcatalog

```
$ gdsctl
GDSCCTL> connect
gsmcatuser/gsmcatuser_credential@w23catyyz10.db.maasdbpoc.oraclevcn.com:1521/catpdb.db.maasdb
poc.oraclevcn.com
Catalog connection is established
```

```
GDSCCTL> create shardcatalog -user gsmcatuser/gsmcatuser_credential -database
w23catyyz10.db.maasdbpoc.oraclevcn.com:1521/catpdb.db.maasdbpoc.oraclevcn.com -region ca-
toronto-1 -configname oradbcloud -sdb sharddb -autovncr OFF -chunks 120 -sharding system
Catalog is created
```

GSMインスタンスの追加と起動

シャード・ディレクトはシャーディング・システムを監視し、GDSCCTLコマンドやその他のイベントに対応してバックグラウンド・タスクを実行します。

各シャード・ディレクト・ホスト上でGSMインスタンスを1つずつ追加して起動し、シャード・ディレクト・ホスト上でローカルにコマンドを実行します。シャード・ディレクトを既存の構成に追加する場合は、新しいGSMを追加する前に、シャード・カタログ・データベースに接続します。

今回の例では、初期構成のために以前に開始されたGDSCTLセッションを継続しています。

例 : gdsctl create/start GSM

```
GDSCTL> add gsm -gsm w23diryyz10 -catalog
w23catyyz10.db.maasdbpoc.oraclevcn.com:1521/catpdb.db.maasdbpoc.oraclevcn.com -region ca-
toronto-1 -listener 1522 -localons 6123 -remoteons 6234 -pwd gsmcatuser_credential
GSM successfully added
```

```
GDSCTL> start gsm -gsm w23diryyz10
GSM is started successfully
```

```
GDSCTL> config gsm
Name      Region      ENDPOINT
----      -
w23diryyz ca-toront  (ADDRESS=(HOST=w23diryyz10.db.maasdbpoc.oraclevcn.com)(PORT
10        o-1      =1522)(PROTOCOL=tcp))
```

ホスト・メタデータの追加

シャード・ホストの許可されたホスト名とIPアドレスを構成に追加します。上記の例に示したように、"-autovncr off"パラメータ値を使用して、登録用有効ノード・チェックの自動化（AUTOVNCR）機能を無効にした状態でシャード・カタログを作成することをお勧めします。

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にアクセスし、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要求のみを受け入れ、未知のホストからの登録要求を拒否します。

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、シャードからシャード・ディレクタへの着信登録リクエストが、GDSCTL ADD SHARD操作時に自動的に追加されなかったアドレスから送信される可能性があります。この場合、登録リクエストは拒否され、シャードは正しくデプロイされません。この問題の目に見える兆候は、DEPLOYの完了後にGDSCTL CONFIG SHARDを実行すると、シャードの可用性がPENDINGと表示されることです。

Oracle Real Application Clusters（Oracle RAC）を使用してデプロイする場合、各Oracle RACインスタンスのパブリックIPとフローティングVIPのIPを追加する必要があります。SCAN VIPとFQDNをここに含める必要はありません。

GDSCTL ADD INVITEDNODEコマンドを使用して、シャード・ホストに必要なエントリをシャード・カタログ・メタデータに手動で追加します。

詳細と例については、「[ホスト・メタデータの追加](#)」のドキュメントを参照してください。

例 : gdsctl add inspirednode - 4つのシャードのエントリ - IPアドレスとホスト名

```
add invitednode 10.8.208.94
add invitednode w23shdyyz10
add invitednode w23shdyyz10.db.maasdbpoc.oraclevcn.com

add invitednode 10.8.208.78
add invitednode w23shdyyz11
add invitednode w23shdyyz11.db.maasdbpoc.oraclevcn.com

add invitednode 10.8.208.89
add invitednode w23shdyyz12
add invitednode w23shdyyz12.db.maasdbpoc.oraclevcn.com
```

```
add invitednode 10.8.208.72
add invitednode w23shdyyz13
add invitednode w23shdyyz13.db.maasdbpoc.oraclevcn.com
```

すべてを追加した後、`gdsctl config vncr`を実行して、すべてのホストとIPのエントリが含まれていることを確認します。

シャードグループの追加

システム定義のシャーディングを使用する場合は、`ADD SHARDGROUP`コマンドを使用して目的のシャーディング・トポロジを構成します。

プライマリ、スタンバイ、アクティブ・スタンバイの各データベースに対応する複数のシャードグループ・タイプは、`-deploy_as`パラメータを使用して宣言できます。デフォルトは"standby"です。

シャードを新しいシャードグループに追加する前に、各シャード・データベースが正しいステータスで開かれていることを確認してください。その後シャードグループに追加されたシャードは、シャードグループの`-deploy_as`設定に対応するモードで開く必要があります。たとえば、プライマリ・シャードグループでは`READ WRITE`、スタンバイ・シャードグループでは`MOUNTED`、アクティブ・スタンバイ・シャードグループでは`READ ONLY WITH APPLY`です。

例 : `gdsctl create shardgroup` - プライマリ

```
GDSCtl> add shardgroup -shardgroup shardgroup_primary -shardspace shardspaceora -deploy_as
primary -region ca-toronto-1
The operation completed successfully
```

今回の例では、HAまたはディザスタ・リカバリのためのレプリケーションが実装されていなかったため、すべてのシャードがデフォルトのプライマリ・シャード領域に構成されています。詳細と例については、「[シャード領域の追加](#)」のドキュメントを参照してください。

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シャード・カタログに追加する前に、シャーディング・トポロジが正しいことを確認してから、さまざまなGDSCtl CONFIGコマンドを使用して続行してください。

さまざまなGDSCtl CONFIGコマンドを使用して、シャード領域、シャードグループ、およびその他のシャード・カタログ・オブジェクトに関する詳細情報を表示できます。GDSCtl CONFIGコマンドのバリエーション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、またはGDSCtl HELPを実行してください。

注 : この時点では、データベースやサービスは何も表示されません。

例 : `gdsctl status`コマンドと`gdsctl config`コマンドを使用して構成を検証する

GDCTL> status

Alias W23DIRYYZ10

Version 23.0.0.0.0

Start Date 09-MAY-2024 16:36:04

Trace Level off

Listener Log File /u01/app/oracle/diag/gsm/w23diryyz10/w23diryyz10/alert/log.xml

Listener Trace File

/u01/app/oracle/diag/gsm/w23diryyz10/w23diryyz10/trace/ora_949066_139912260905920.trc

Endpoint summary

(ADDRESS=(HOST=w23diryyz10.db.maasdbpoc.oraclevcn.com)(PORT=1522)(PROTOCOL=tcp))

GSMOCI Version 4.0.191114

Mastership Y

Connected to GDS catalog Y

Process Id 949068

Number of reconnections 0

Pending tasks. Total 0

Tasks in process. Total 0

Regional Mastership TRUE

Total messages published 0

Time Zone +00:00

Orphaned Buddy Regions:

None

GDS region ca-toronto-1

GDCTL> config

Catalog connection is established

Regions

ca-toronto-1

GSMs

w23diryyz10

Sharded Database

Sharddb

Databases

Shard Groups-----
shardgroup_primary

Shard spaces

shardspaceora

Services

GDCTL pending requests

Command

Object

Status

Global properties

Name: oradbcloud

Master GSM: w23diryyz10

DDL sequence #: 0

シャードCDBの追加

ADD CDBコマンドを使用して、シャードPDBを含む各CDBをシャーディング構成に追加します。接続文字列またはTNSエイリアスをgsmrootuser資格証明とともに含めます。これは、以前にシャード・カタログ接続に使用されたものとは異なる、データベース・スキーマ・ユーザー資格証明です。すべてのCDBが追加されたことを確認し終わったら、CONFIG CDBコマンドを使用します。詳細と例については、「[シャードCDBの追加](#)」のドキュメントを参照してください。

各シャードの接続文字列は、単純な直接接続の詳細で構成されている必要があります。Oracle GDSは各シャードCDBを個別に処理するため、DR対応の接続文字列は必要ありません。OCI Base Database Systemデータベースを使用する場合、CDBの接続文字列では、システムのドメイン名で完全修飾された、データベースの一意の名前が使用されます。

例：シャードCDBの追加

```
GDSTCL> add cdb -connect  
w23shdyyz10.db.maasdbpoc.oraclevcn.com:1521/shdcdb0_3s2_yyz.db.maasdbpoc.oraclevcn.com -pwd  
gsmrootuser_password
```

```
GDSTCL> add cdb -connect  
w23shdyyz11.db.maasdbpoc.oraclevcn.com:1521/shdcdb1_224_yyz.db.maasdbpoc.oraclevcn.com -pwd  
gsmrootuser_password
```

```
GDSTCL> add cdb -connect  
w23shdyyz12.db.maasdbpoc.oraclevcn.com:1521/shdcdb2_542_yyz.db.maasdbpoc.oraclevcn.com -pwd  
gsmrootuser_password
```

```
GDSTCL> add cdb -connect  
w23shdyyz13.db.maasdbpoc.oraclevcn.com:1521/shdcdb3_pqd_yyz.db.maasdbpoc.oraclevcn.com -pwd  
gsmrootuser_password
```

```
GDSTCL> config cdb  
Catalog connection is established  
Name                               Host
```

```
----                               ----  
shdcdb0_3s2_yyz  
shdcdb1_224_yyz  
shdcdb2_542_yyz  
shdcdb3_pqd_yyz
```

config cdbの出力には、すべてのケースでホスト名が表示されるというわけではないことに注意してください。これは問題ではありません。

シャードPDBの追加

ADD SHARDコマンドを使用して、各シャードPDBをシャーディング構成に追加します。

gsmuser資格証明、シャードグループ名、CDB名とともに、シャードPDBの接続文字列またはTNSエイリアスのパラメータを含めます。すべてのPDBが追加されたことを確認し終わったら、CONFIG SHARDコマンドを使用します。

詳細と例については、「[シャードPDBの追加](#)」のドキュメントを参照してください。

例：シャードPDBの追加

```
GDCTL> add shard -connect
w23shdyz10.db.maasdbpoc.oraclecn.com:1521/shdpdb0.db.maasdbpoc.oraclecn.com -shardgroup
shardgroup_primary -cdb shdcdb0_3s2_yyz -pwd gsmuser_credential
```

```
GDCTL> add shard -connect
w23shdyz11.db.maasdbpoc.oraclecn.com:1521/shdpdb1.db.maasdbpoc.oraclecn.com -shardgroup
shardgroup_primary -cdb shdcdb1_224_yyz -pwd gsmuser_credential
```

```
GDCTL> add shard -connect
w23shdyz12.db.maasdbpoc.oraclecn.com:1521/shdpdb2.db.maasdbpoc.oraclecn.com -shardgroup
shardgroup_primary -cdb shdcdb2_542_yyz -pwd gsmuser_credential
```

```
add shard -connect
w23shdyz13.db.maasdbpoc.oraclecn.com:1521/shdpdb3.db.maasdbpoc.oraclecn.com -shardgroup
shardgroup_primary -cdb shdcdb3_pqd_yyz -pwd gsmuser_credential
```

```
GDCTL> config shard
```

Name	Shard Group	Status	State	Region	Availability
-----	-----	-----	-----	-----	-----
shdcdb0_3s2_yyz_shd pdb0	shardgroup_primary	U	none	ca-toront o-1	none
shdcdb1_224_yyz_shd pdb1	shardgroup_primary	U	none	ca-toront o-1	none
shdcdb2_542_yyz_shd pdb2	shardgroup_primary	U	none	ca-toront o-1	none
shdcdb3_pqd_yyz_shd pdb3	shardgroup_primary	U	none	ca-toront o-1	none

DEPLOYコマンドが正常に実行されるまで、Statusの値はU（“アンデプロイ済み”）であり、StateとAvailabilityはnoneと-であることに注意してください。また、列の長さはハードコードされています。

シャード構成のデプロイ

シャードを追加したら、GDCTL DEPLOYコマンドを実行します。これにより、シャード・カタログに対して、アンデプロイ済みのステータスで最近追加されたシャードのデプロイがリクエストされます。

トポロジによっては、出力が大幅に異なる場合があります。何らかの問題がある場合の対処法に関するその他の例とアドバイスについては、「[構成のデプロイ](#)」のドキュメントを参照してください。

例 : gdsctl deploy

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

GDSTCL> config shard

Name	Shard Group	Status	State	Region	Availability
----	-----	-----	-----	-----	-----
shdcdb0_3s2_yyz_shd	shardgroup_primary	Ok	Deployed	ca-toront	ONLINE
pdb0				o-1	
shdcdb1_224_yyz_shd	shardgroup_primary	Ok	Deployed	ca-toront	ONLINE
pdb1				o-1	
shdcdb2_542_yyz_shd	shardgroup_primary	Ok	Deployed	ca-toront	ONLINE
pdb2				o-1	
shdcdb3_pqd_yyz_shd	shardgroup_primary	Ok	Deployed	ca-toront	ONLINE
pdb3				o-1	

Global Database Servicesの作成と開始

シャードが正常にデプロイされ、正しいステータスが確認されたら、シャード上でグローバル・データベース・サービスを作成して開始し、アプリケーションからの着信接続要求を処理します。

最初の例のコマンドでは、構成内のプライマリ・シャードに読み取り/書き込みサービスが作成されます。スタンバイ・シャードがある場合は、スタンバイ・シャード上に読み取り専用サービスを作成できます。これらのサービス名をアプリケーションからの接続文字列で使用して、リクエストを正しいシャードに適切にルーティングできます。詳細については、ドキュメントの「[Global Data Servicesの作成と開始](#)」を参照してください。

例：プライマリ・シャード（非Oracle RAC）で実行されるグローバル・サービスを追加して開始する

```
GDSTCL> add service -service oltp_rw_srvc -role primary
```

```
GDSTCL> start service -service oltp_rw_srvc
```

例：スタンバイ・シャード（非Oracle RAC）で実行される読み取り専用ワークロードのグローバル・サービスを追加して開始する

```
GDSTCL> add service -service oltp_ro_srvc -role physical_standby
```

```
GDSTCL> start service -service oltp_ro_srvc
```

注： Oracle RACデータベースを使用している場合は、サービスを開始する前に、各サービスを変更して、優先Oracle RACインスタンスを設定する必要があります。ポリシー管理ではなく、管理者管理のOracle RACを使用する場合、Oracle RACノードごとにサービスを変更し、サービスが実行される可能性のあるすべてのインスタンスを含める必要があります。これには、プライマリOracle RACとすべてのOracle Active Data Guardスタンバイが含まれます。サービスを構成、変更、開始する前に、スタンバイ・インスタンスを読み取り専用モードで開きます。マウントされた非オープン・スタンバイは、デガスタ・リカバリと高可用性のみを目的として存在し、接続を処理することはできません。

注： -preferredパラメータのデータベース・インスタンス名では大文字と小文字が区別され、‘show parameters instance_name’ SQL文のインスタンス名dbのパラメータ値と一致する必要があります。

例：サービスを変更して優先Oracle RACインスタンスを設定する

```
GDSTCL> add service -service oltp_rw_srvc -role primary
```

```
GDSTCL> modify service -gds pool poc -service oltp_rw_srvc -database poc_iad1f9_usshard -  
modify_instances -preferred POC1,POC2
```

```
GDSTCL> modify service -gds pool poc -service oltp_rw_srvc -database poc_yyz173_usshard -  
modify_instances -preferred POC1,POC2
```

```
GDSTCL> start service -service oltp_rw_srvc
```

最後に、セットアップが完了したので構成を検証します。

GDCTL> config
Catalog connection is established

Regions

ca-toronto-1

GSMs

w23diryyz10

Sharded Database

sharddb

Databases

shdcdb0_3s2_yyz_shdpdb0
shdcdb1_224_yyz_shdpdb1
shdcdb2_542_yyz_shdpdb2
shdcdb3_pqd_yyz_shdpdb3

Shard Groups

shardgroup_primary

Shard spaces

shardspaceora

Services

GDCTL pending requests

Command	Object	Status
-----	-----	-----

Global properties

Name: oradbcloud
Master GSM: w23diryyz10
DDL sequence #: 0

GDCTL> config service

Name	Network name	Pool	Started	Preferred all
-----	-----	-----	-----	-----
oltp_rw_srvc	oltp_rw_srvc.sharddb.oradbcloud	sharddb	Yes	Yes

データベースがすべて正しく登録されており、サービスが適切に開始されていることを確認します。データベースは正常に登録され、そのStateは“OK”であると示されている必要があります。各データベースのサービスはすべて「Globally started」、「Started」、「Enabled」、および「Preferred」である必要があります。これらのパラメータの値が“Y”であることを確認します。

GDSTCL> databases

Database: "shdcdb0_3s2_yyz_shdpdb0" Registered: Y State:Ok ONS:N. Role:PRIMARY Instances:

1 Region: ca-toronto-1

Service: "oltp_rw_srvc" Globally started:Y Started:Y

Scan:N Enabled:Y Preferred:Y

Registered instances:

sharddb%1

Database: "shdcdb1_224_yyz_shdpdb1" Registered: Y State:Ok ONS:N. Role:PRIMARY Instances:

1 Region: ca-toronto-1

Service: "oltp_rw_srvc" Globally started:Y Started:Y

Scan:N Enabled:Y Preferred:Y

Registered instances:

sharddb%11

Database: "shdcdb2_542_yyz_shdpdb2" Registered: Y State:Ok ONS:N. Role:PRIMARY Instances:

1 Region: ca-toronto-1

Service: "oltp_rw_srvc" Globally started:Y Started:Y

Scan:N Enabled:Y Preferred:Y

Registered instances:

sharddb%21

Database: "shdcdb3_pqd_yyz_shdpdb3" Registered: Y State:Ok ONS:N. Role:PRIMARY Instances:

1 Region: ca-toronto-1

Service: "oltp_rw_srvc" Globally started:Y Started:Y

Scan:N Enabled:Y Preferred:Y

Registered instances:

sharddb%31

完成したシャード・データベース構成に警告やエラーがないか検証します。

GDSTCL> validate

Validation results:

[Warning] VLD2: Region "ca-toronto-1" does not have buddy region [Warning] VLD63: No sharded table family exists.

Total errors: 0.Total warnings:2

この時点では、最終的な“検証”出力にある上記の警告が表示されることが予想されます。今回の例では、リージョンを1つだけ設定しており、リージョン間レプリケーションの計画はありません。次の手順は、シャード表ファミリーを作成することです。

シャード・データベース・スキーマの作成

シャード・データベースでスキーマ・オブジェクトを作成するために重要な、いくつかの手順と特定の詳細があります。詳細な仕様については、『Oracle Globally Distributed Databaseガイド』の[「スキーマ・オブジェクトの作成」](#)セクションを参照してください。

今回のシャード表ファミリーは、"Articles"という名前の単一表と、"Article_JSON"という名前のJSON列に保存された、Wikipedia記事のJSONデータで構成されています。今回のシャード表ファミリーは、主キーと一致するシャーディング・キーでシステム管理されており、両方ともArticle_ID列を使用します。Article_ID列の主キー/シャーディング・キーの値は、データ・ロード時にWikipedia記事のJSONドキュメント・コンテンツである\$.wid要素から抽出されます。

関係演算子を使用して一部のSQL問合せをより効率的にし、複雑さを軽減するために、仮想列は一部のJSON要素値の検索を抽象化するように構成されています。特に、Article_JSONからJSON_VALUE()問合せを使用して特定の要素を抽出し、場合によっては、仮想列仕様の一部として型変換を集中的に実行します。

仮想列は、Wikipedia記事ドキュメントにJSON要素の.wid、.title、.wikiTitle、.lang、.timestampを含めるために作成されます。

シャード・データベース環境で必要とされる、新しいスキーマをデプロイする手順と追加の詳細は、次のように要約できます。

1. SQLセッションを開始し、SYSDBAとしてシャード・カタログPDBに接続する。
2. セッションを変更してシャードDDLを有効にし、シャード・カタログ・データベースだけでなく、すべてのシャードでスキーマのデプロイが行われるようにする。
3. 全シャード・スキーマ・ユーザーを作成する。
4. ロールと権限を付与する。
5. 新しいスキーマ・ユーザーとしてシャード・カタログ・データベースに再接続する。
6. セッションを変更してシャードDDLを有効にし、シャード・カタログ・データベースだけでなく、すべてのシャードでスキーマのデプロイが確実に行われるようにする。
7. 表領域セットを作成し、割当て制限を適切に設定する。
8. シャード表ファミリーを作成し、主キー、表領域セット、パーティション化スキーム、およびシャーディング・キーとして使用する表列を指定します。
9. 必要なファンクション索引を作成する（Oracle Text索引は、データのロード後に独自の章に作成されます）。

始める前に、シャードDDLの実行に関して、以下に述べる注意事項があります。

SQL*Plusを使用し、セッションでシャードDDLを有効にしてシャード・カタログPDBでDDLを実行すると、SQL実行はシャード・カタログでの実行が完了すると完了済みを返します。追加のDDL文に進む前に、シャードでの実行が完了したかどうか、およびいずれかのシャードでエラーが発生したかどうかを把握するには、さらに2つの手順が必要です。

次のDDL文を実行しても安全かどうかを最終的に把握するには、次のようにします。

1. 各DDLリクエストの間に、カタログPDBセッションでgsmadmin_internal.dbms_gsm_utility.wait_for_ddl()関数を実行して、続行する前にすべてのシャードでの実行が完了していることを確認する必要があります。この関数は、すべてのシャードでの実行が完了するまで待機してから、SQL*Plusに制御を返します。ただし、SQLが正常に完了したかどうか、またはシャードでエラーが発生したかどうかは通知されません。
2. すべてのシャードにわたってDDL実行時のエラーをチェックするには、シャード・ディレクタでGDSCCTL SHOW DDLコマンドを使用します。これでは問合せの完了は表示されず、エラーのみが表示されます。

この複雑さを回避するために、シャード・ディレクタ・ホストからGDSCCTLコマンドライン・ツールを使用してSQLを実行できます。ドキュメントの「[Oracle Globally Distributed DatabaseでのDDLの処理](#)」セクションを参照してください。

たとえば、次のとおりです。

```
[oracle@sharddirector1] $ gdsctl
GDSCCTL> connect sys@catalog.fqdn:1521/catpdb as sysdba
password:
Catalog connection is established

GDSCCTL> sql "GRANT CONNECT, RESOURCE, ALTER SESSION TO wikip"
The operation completed successfully
```

```
GDSCCTL> show ddl -count 1
ld      DDL Text                                     Failed shards
--      -
113     GRANT CONNECT, RESOURCE, ALTER SESSIO...
```

今回の例ではその目的を考慮して、SQL*Plusを使用しました。

SQL*Plusを介してシャード・カタログからスキーマ・ユーザーを作成するDDLの例：

DDL：シャード・カタログ・データベースを使用してシャード・スキーマ・ユーザーを作成する

REM SYSがSYSDBAとしてカタログ・データベースに接続します

```
SQL> CONNECT sys@catpdb as sysdba;
```

```
-- セッションのシャードDDLおよび/またはシャード操作の処理を設定します\
ALTER SESSION ENABLE SHARD DDL;
```

```
-- 念のために、今回のスキーマ・ユーザーを削除します
```

```
DROP USER wiki CASCADE;
```

```
exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

```
-- 今回のスキーマ・ユーザーを作成します
```

```
CREATE USER wiki IDENTIFIED BY "RedactedCredentialString";
```

```
exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

wait_for_ddl()関数の使用後にシャード・カタログ・データベースからDDLを実行するときは、その都度、どのシャードでもDDLが失敗していないことを必ず確認してから続行してください。シャード・ディレクトでGDSCTLコマンドを使用して、実行されたDDLを一覧表示し、そのDDLがいずれかのシャードで失敗したかどうかを示します。

すべてのシャードでのDDL完了のステータスを確認するためのGDSCTLの使用例：

GSM：障害が発生したシャードのDDLステータスを確認する

シャード・ディレクトでGDSCTLによってDDLステータスを調べ、成功か、シャードで障害が発生したかを確認します

-countパラメータを使用して、提示されるデータを必要に応じて最新のDDLのみに減らします

```
[oracle@sharddirector1] $ gdsctl show ddl -count 1
```

Id	DDL Text	Failed shards
88	CREATE USER wiki IDENTIFIED BY...	

必要なロールと権限の付与を続けます。必要なロールと権限は、個別の要件に応じて異なる場合があります。

DDL：今回の例のロールと権限を付与する

```
SQL>
```

```
-- 新しいスキーマ・ユーザーにロールを付与し、シャードが追いつくまで待機します
```

```
GRANT CONNECT, RESOURCE, ALTER SESSION TO wiki;
```

```
GRANT CREATE TABLE, CREATE PROCEDURE, CREATE TABLESPACE, CREATE MATERIALIZED VIEW TO wiki;
```

```
GRANT GSMADMIN_ROLE TO wiki;
```

```
GRANT GSM_POOLADMIN_ROLE TO wiki;
```

```
GRANT ANALYZE ANY TO wiki;
```

```
GRANT SELECT ANY DICTIONARY TO wiki;
```

```
GRANT SELECT ANY TABLE TO wiki;
```

```
GRANT SELECT_CATALOG_ROLE TO wiki;
```

```
GRANT EXECUTE ON DBMS_CRYPTO TO wiki;
```

```
GRANT EXECUTE ON DBMS_LOCK TO wiki;
```

```
GRANT EXECUTE ON DBMS_SODA TO wiki;
```

```
GRANT EXECUTE ON DBMS_SODA_ADMIN TO wiki;
```

```
GRANT EXECUTE ON DBMS_SYS_ERROR TO wiki;
```

```
GRANT EXECUTE ON EXEC_SHARD_PLSQL TO wiki;
```

```
GRANT CTXAPP TO wiki;
```

```
-- CTSパッケージのPL/SQLパッケージに対する実行権限を付与します
```

```
-- これらはctxappロールによっても付与されますが、ロール権限がPL/SQLプロシージャで常に機能するとは限りません
```

```
-- CTXAPPロールをすでに有するユーザーにそれらの権限を明示的に付与するのが、もっとも安全です
```

```
GRANT EXECUTE ON CTXSYS.CTX_ADM TO wiki;
```

```
GRANT EXECUTE ON CTXSYS.CTX_CLS TO wiki;
```

```
GRANT EXECUTE ON CTXSYS.CTX_DDL TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_DOC TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_OUTPUT TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_QUERY TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_REPORT TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_THES TO wikip;
GRANT EXECUTE ON CTXSYS.CTX_ULEXER TO wikip;
```

```
exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

```
GDSCTL> show ddl -count 24
```

```
Catalog connection is established
```

id	DDL Text	Failed shards
--	-----	-----
89	GRANT	CONNECT, RESOURCE, ALTER SES...
90	GRANT	CREATE TABLE, CREATE PROCEDU...
91	GRANT	GSMADMIN_ROLE TO wikip; ...
92	GRANT	GSM_POOLADMIN_ROLE TO wikip;...
93	GRANT	ANALYZE ANY TO wikip; ...
94	GRANT	SELECT ANY DICTIONARY TO wik...
95	GRANT	SELECT ANY TABLE TO wikip; ...
96	GRANT	SELECT_CATALOG_ROLE TO wikip...
97	GRANT	EXECUTE ON DBMS_CRYPTO TO wi...
98	GRANT	EXECUTE ON DBMS_LOCK TO wiki...
99	GRANT	EXECUTE ON DBMS_SODA TO wiki...
100	GRANT	EXECUTE ON DBMS_SODA_ADMIN T...
101	GRANT	EXECUTE ON DBMS_SYS_ERROR TO...
102	GRANT	EXECUTE ON EXEC_SHARD_PLSQL ...
103	GRANT	CTXAPP TO wikip; ...
104	GRANT	EXECUTE ON CTXSYS.CTX_ADM TO...
105	GRANT	EXECUTE ON CTXSYS.CTX_CLS TO...
106	GRANT	EXECUTE ON CTXSYS.CTX_DDL TO...
107	GRANT	EXECUTE ON CTXSYS.CTX_DOC TO...
108	GRANT	EXECUTE ON CTXSYS.CTX_OUTPUT...
109	GRANT	EXECUTE ON CTXSYS.CTX_QUERY ...
110	GRANT	EXECUTE ON CTXSYS.CTX_REPORT...
111	GRANT	EXECUTE ON CTXSYS.CTX_THES T...
112	GRANT	EXECUTE ON CTXSYS.CTX_ULEXER...

すべてのシャードでスキーマ・ユーザーが正常に作成されたら、新しいスキーマ・ユーザーとしてシャード・カタログ・データベースに再接続し、シャード・データベース実装用の表領域セットと表ファミリーを作成するためのDDLを実行します。

シャード・データベースでは、個々のシャード・データベースにデータセット・ペイロードの一部（1/シャード）が含まれることに注意してください。パーティションとデータファイルは、シャード領域の定義時に宣言した"チャンク"ごとに作成されます。デフォルトのチャンク数は120です。4シャード環境では、データベース・インスタンスごとに120個のパーティションがあり、そのうち30個にはデータが移入され、残りの90個のパーティションのデータファイルは空になります。

シャード表ファミリーの表領域を計画および構成する場合、DDLで指定したデータファイル・サイズが、（移入された）各パーティションのデータファイルに使用されます。4個のシャードと120個のチャンクがあり、データファイルの最大サイズが20 Gの新しい表領域セットは、最大で620 GBのストレージを消費します。予想されるデータのインポートと索引付けのペイロードに対応できるように、これを十分なサイズに設定します。

表領域セットと表ファミリーを作成するためのDDLおよびGSMコマンドの例は、次のとおりです。

DDL：シャード表領域セットを作成する

```
REM WIKIPスキーマ・ユーザーがカタログ・データベースおよびシャード・データベースに接続します
CONNECT wikip@catpdb;
```

```
-- セッションのシャードDDLおよび/またはシャード操作の処理を設定します
```

```
ALTER SESSION ENABLE SHARD DDL;
```

```
-- 今回の表領域セットを作成し、シャードが追いつくまで待機します
-- シャードあたり31個のチャンクがあり、それぞれの最大サイズが20 Gであるため、全体でおよそ620 GBとなります
CREATE BIGFILE TABLESPACE SET TSP_SET_WIKIP
  USING TEMPLATE (
    DATAFILE SIZE 10G
    AUTOEXTEND ON NEXT 250M maxsize 20G
    EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO
  );

exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

すべてのシャードでCREATE TABLESPACE SET DDLが完了したことを確認します。

GSM : 障害が発生したシャードのDDLステータスを確認する

```
# シャード・ディレクトでGDSCTLによってDDLステータスを調べ、成功か、シャードで障害が発生したかを確認します
# -countパラメータを使用して、提示されるデータを必要に応じて最新のDDLのみに減らします
```

```
[oracle@sharddirector1] $ gdsctl show ddl -count 1
```

Id	DDL Text	Failed shards
113	CREATE BIGFILE TABLESPACE SET...	

新しい表領域での新しいスキーマ・ユーザーの割当て制限を変更して無制限に設定し、sysdbaとして接続します。

DDL : ユーザーを変更して表領域ファミリーに無制限の割当て制限を設定する

```
REM WIKIPスキーマ・ユーザーがカタログ・データベースおよびシャード・データベースに接続します
CONNECT sys@catpdb as sysdba;
```

```
-- セッションのシャードDDLおよび/またはシャード操作の処理を設定します
ALTER SESSION ENABLE SHARD DDL;
```

```
ALTER USER wikip QUOTA UNLIMITED ON TSP_SET_WIKIP;
exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

すべてのシャードでALTER USER DDLが完了したことを確認します。

GSM : 障害が発生したシャードのDDLステータスを確認する

```
# シャード・ディレクトでGDSCTLによってDDLステータスを調べ、成功か、シャードで障害が発生したかを確認します
# -countパラメータを使用して、提示されるデータを必要に応じて最新のDDLのみに減らします
```

```
[oracle@sharddirector1] $ gdsctl show ddl -count 1
```

Id	DDL Text	Failed shards
90	ALTER USER wikip QUOTA UNLIMITED ON...	

表ファミリーの作成

今回の例では、Article_ID列を主キーとシャーディング・キーの両方として使用します。シャーディング・キーと、システム管理のデータ分散の使用は、PARTITION BY CONSISTENT HASH句で指定します。また、リレーショナル問合せ演算子を使用した問合せにおいて特定のJSON属性値の使用をサポートするために、いくつかの仮想列も組み込まれています。

DDL : シャード表ファミリーを作成する

REM WIKIPスキーマ・ユーザーがカタログ・データベースおよびシャード・データベースに接続します
CONNECT wikip@catpdb;

```
-- セッションのシャードDDLおよび/またはシャード操作の処理を設定します
ALTER SESSION ENABLE SHARD DDL;
-- 表ファミリーを作成し、すでに存在する場合は削除します
-- すべてのシャードで削除と作成の実行が完了するまで待機してから続行します
DROP TABLE ARTICLES CASCADE CONSTRAINTS PURGE;
exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
CREATE SHARDED TABLE Articles
(
  Article_JSON      JSON          NOT NULL,
  Article_ID        VARCHAR(10)   NOT NULL,
  Date_Loaded       TIMESTAMP(6)  DEFAULT SYS_EXTRACT_UTC(SYSTIMESTAMP) NOT NULL,
  Title             VARCHAR(512)   AS ( JSON_VALUE(Article_JSON, '$.title'
                                     returning varchar(512) null on error
                                     null on empty) ),
  Wiki_Title        VARCHAR(512)  AS ( JSON_VALUE(Article_JSON, '$.wikiTitle'
                                     returning varchar(512) null on error
                                     null on empty) ),
  Lang              VARCHAR(2)     AS ( JSON_VALUE(Article_JSON, '$.lang'
                                     returning varchar(2) null on error
                                     null on empty) ),
  Last_Modified     TIMESTAMP(0)   WITH TIME ZONE AS ( TO_TIMESTAMP_TZ(
                                     JSON_VALUE(Article_JSON, '$.timestamp' null on error null on empty),
                                     'YYYY-MM-DD"T"HH24:MI:SS TZH:TZM' ) ),
  PRIMARY KEY (Article_ID)
)
TABLESPACE SET tsp_set_wikip
PARTITION BY CONSISTENT HASH (Article_ID) PARTITIONS AUTO;

SQL> exec gsmadmin_internal.dbms_gsm_utility.wait_for_ddl();
```

GSM : シャードのステータスについてDDLステータスを確認する

シャード・ディレクタでGDSCTLによってDDLステータスを調べ、シャードが成功かを確認します
-countパラメータを使用して、提示されるデータを必要に応じて最新のDDLのみに減らします。

```
[oracle@sharddirector1] $ gdsctl show ddl -count 2
```

Id	DDL Text	Failed shards
--	-----	-----
91	DROP TABLE ARTICLES CASCADE CONSTRAIN...	
92	CREATE SHARDED TABLE Articles (Articl...	

表ファミリーを作成したら、同じSQLセッションを使用して表を記述し、仮想列を確認します。

```
SQL> desc articles
```

Name	Null?	Type
------	-------	------

```

-----
ARTICLE_JSON          NOT NULL  JSON
ARTICLE_ID            NOT NULL  VARCHAR2(10)
DATE_LOADED           NOT NULL  TIMESTAMP(6)
TITLE                 VARCHAR2(512)
WIKI_TITLE            VARCHAR2(512)
LANG                  VARCHAR2(2)
LAST_MODIFIED         TIMESTAMP(0) WITH TIME ZONE

```

```

SQL> -- 仮想列の詳細を問い合わせます
SQL> col column_name format a48;
SQL> col data_default format a80;
SQL> SELECT
        column_name,
        virtual_column,
        data_default
FROM
    all_tab_cols
WHERE owner = 'WIKIP'
    AND column_name not like 'SYS%';

```

COLUMN_NAME	VIR	DATA_DEFAULT
ARTICLE_JSON	NO	
ARTICLE_ID	NO	
DATE_LOADED	NO	SYS_EXTRACT_UTC(SYSTIMESTAMP)
TITLE	YES	JSON_VALUE("ARTICLE_JSON" FORMAT OSON, '\$.title' RETURNING VARCH
WIKI_TITLE	YES	JSON_VALUE("ARTICLE_JSON" FORMAT OSON, '\$.wikiTitle' RETURNING V
LANG	YES	JSON_VALUE("ARTICLE_JSON" FORMAT OSON, '\$.lang' RETURNING VARCHA
LAST_MODIFIED	YES	TO_TIMESTAMP_TZ(JSON_VALUE("ARTICLE_JSON" FORMAT OSON, '\$.timest

JSONデータのバルク・ロード

既存のJSONデータファイルからJSONドキュメントをスキーマに一括でロードするには、必要なデータファイルを各シャードにステージングし、続いて外部表を宣言し、Oracle SQL*Loaderを使用してJSONレコードから主キー/シャーディング・キー値を抽出し、外部表から選択したデータをシャード表ファミリーに挿入する必要があります。これは、すべてのシャードで同時に並行して実行できます。各シャードの範囲内のハッシュ値があるシャーディング・キーを持つJSONドキュメントのみが、そのシャードにロードされます。

データをロードするときに、挿入する必要がある列は、JSON列と主キー/シャーディング・キーのみであることに注意してください。今回の例の表で他のすべての列は仮想列であり、その値は仮想列宣言に基づいてJSON列から導出されます。

JSONデータファイルのデプロイ

外部表を使用すると、中央のJSONファイルから各シャード・データベースにデータを個別にロードできるようになります。JSONデータファイルは、各シャードのローカル・ファイル・システムに直接ロードでき、共有ストレージの場所またはOCI Object Storageからアクセスできます。

今回の例では単に、データファイルを各シャードに一時的にコピーし、外部表として公開しています。これはすべてのシャードで実行されます。データファイルenwiki.jsonを、各シャード・データベース・サーバーの/u01/wikip/dataフォルダにロードします。このフォルダはローカルに置くことも、すべてのシャードにマウントされた共有ファイル・システム上に置くこともできます。フォルダが存在するようになると、シャードPDBでsysdba権限を持つsysとして接続している間、次のようにしてシャード・データベースでそのフォルダを使用できるようになります。

サンプル・コード：シャード・データ・ディレクトリの登録

```
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> DROP DIRECTORY wikip_data_dir;
SQL> CREATE OR REPLACE DIRECTORY wikip_data_dir AS '/u01/wikip/data';
SQL> GRANT ALL on DIRECTORY wikip_data_dir TO wikip;
```

ディレクトリが構成されたら、全シャード・ユーザー（今回の場合は“wikip”）として接続している間、Oracle Loaderと外部表を使用してそのディレクトリ内のデータファイルを参照できます。

Oracle Loaderが長いJSONドキュメント・レコードを正確に解析できるように、データファイル内の個々のJSONドキュメント・レコードのうちもっとも長いものの長さ（バイト単位）を使用して、次の文のREADSIZEアクセス・パラメータを調整する必要があります。

```
SQL> CONNECT WIKIP@SHDPDB0
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> DROP TABLE Articles_Ext;
SQL> CREATE TABLE Articles_Ext
      (Article_JSON JSON NOT NULL)
      ORGANIZATION EXTERNAL(
        TYPE ORACLE_LOADER
        DEFAULT DIRECTORY wikip_data_dir
        ACCESS PARAMETERS (
          RECORDS DELIMITED BY 0x'0A'
          READSIZE 444100
          BADFILE wikip_data_dir: 'JSON_DUMPFILE_CONTENTS.bad'
          LOGFILE wikip_data_dir: 'JSON_DUMPFILE_CONTENTS.log'
          FIELDS(Article_JSON char(444100))
        )
      LOCATION (wikip_data_dir:'enwiki.json')
)
PARALLEL
REJECT LIMIT UNLIMITED;
```

```
-- 行をカウントして、データセット全体が利用可能であることを確認します
select count(*) from Articles_ext;
```

JSONドキュメントのロード

JSONデータが外部表のJSON列として利用可能になると、シャード表ファミリーに挿入するための問合せを実行できます。この問合せは、各シャードで直接、並行して実行できます。insert文により、主キーの“article_id”列と“article_json”列のみに値が移入されます。“article_id”列の値とシャード・キーの値は両方とも、JSON_VALUE()関数を使用して設定され、各JSONレコードから\$.wid要素値が抽出されます。この概念は、アプリケーションの実行時に挿入されたレコードにも適用されます。新しいJSONドキュメントを送信する場合は、“article_id”主キー列の値も送信する必要があり、シャード・キーと同じ値を指定できます。

注：このセクションが完了し、データ・ロードが検証されると、外部表は削除できます。

これを全シャード・ユーザーとしてすべてのシャードで並行して実行します。

```
SQL> CONNECT WIKIP@SHDPDB0
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> ALTER SESSION ENABLE SHARD OPERATIONS;
SQL> ALTER SESSION ENABLE PARALLEL DML;
SQL> INSERT /*+ NOLOGGING APPEND PARALLEL (6) */ INTO ARTICLES (article_id, article_json)
  (SELECT json_value(ext.article_json,'$.wid'), ext.article_json
   FROM articles_ext ext
   WHERE SHARD_CHUNK_ID('WIKIP.ARTICLES',
                        json_value(ext.article_json,'$.wid')) IS NOT NULL
  );
SQL> COMMIT;
```

INSERT AS SELECT文は、パラレルDMLを有効にし、INSERT文に適切なPARALLEL(x)ヒントを含めることにより、データベースで使用可能なCPUスレッドの数に基づいてパラレル実行を利用できます。

この問合せのパラレル処理にすべてのCPUコアを使用することは理想的ではない可能性があり、デフォルト値の“1”のままにすることも理想的ではない可能性があります。データベース・インスタンスのハードウェア・リソース可用性に基づき、必要に応じてこの挿入の並列度（DOP）を調整します。パフォーマンス上の理由から、バルク挿入時のロギングを無効にすることもベスト・プラクティスです。

最後に、各シャードに正しい行のみが挿入されるようにするには、SHARD_CHUNK_ID()関数を使用してWHERE句を組み込み、各行のシャード・キー値のシステム管理ハッシュを評価し、その行がこのシャードのチャンクに属しているかどうかを判定する必要があります。SHARD_CHUNK_ID()関数には、表ファミリーの名前とシャード・キーの値という2つの値が必要です。SHARD_CHUNK_ID()関数の詳細については、[Oracle Database 23ai SQL言語リファレンス](#)を参照してください。

現時点で、システム管理のデータ分散を使用してシャード・データベースを実装する場合は、外部表を介してすべての行を一貫して処理できるように、すべてのシャードでデータファイル全体を利用できるようにすることをお勧めします。オラクルのコンシステント・ハッシュ・モデルの値に基づいて、現在のシャードに割り当てられたチャンクに挿入するのに有効なシャードID値を持つ行を識別するのは、SHARD_CHUNK_ID関数の役割です。詳細については、Oracle Globally Distributed Databaseドキュメントの「[システム管理のシャード・シャード](#)」を参照してください。

統計の収集

すべてのシャードにデータが完全に挿入されたら、シャード・スキーマ・ユーザーのスキーマ統計を収集します。

シャード・データベースで統計を正確に収集するには、いくつかの手順が必要です。統計は各シャードで収集され、コーディネータ（シャード・カタログ・データベース）に“プル”される必要があります。シャードで収集された統計をコーディネータがプルできるようにするには、すべてのシャード・データベースで統計プリファレンス・パラメータCOORDINATOR_TRIGGER_SHARDをtrueに設定する必要があります。

注：マルチテナント・データベース環境にデプロイする場合、次に示す手順を、シャード・カタログ・データベースおよびシャード・プラガブル・データベース（PDB）に対して実行する必要があります。

シャード表で手動統計収集を行う手順は次のとおりです。

1. 全シャード・スキーマ・ユーザーに代わって、すべてのシャードPDBでsysとしてCOORDINATOR_TRIGGER_SHARDをTRUEに設定します。これはシャードごとに1回だけ実行する必要があります。


```
SQL> CONNECT sys/shdpdb0 AS SYSDBA
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> EXECUTE DBMS_STATS.SET_SCHEMA_PREFS('WIKIP','COORDINATOR_TRIGGER_SHARD','TRUE');
```
2. 全シャード・スキーマ・ユーザーとして接続されている、すべてのシャードPDB上の全シャード・ユーザーのスキーマ統計を収集します。SYS.DBMS_STATS.GATHER_SCHEMA_STATS()プロシージャは、各シャードで直接実行することも、セッションでSHARD DDLを有効にし、SYS.EXEC_SHARD_PLSQL()関数を使用してシャード・カタログからすべてのシャードで一度に実行することもできます。

```
SQL> CONNECT wikip@catpdb
SQL> ALTER SESSION ENABLE SHARD DDL;
SQL> EXEC SYS.EXEC_SHARD_PLSQL('SYS.DBMS_STATS.GATHER_SCHEMA_STATS(
OWNNAME => "WIKIP", OPTIONS => "GATHER")');
```

3. すべてのシャードで統計が収集された後、スキーマのシャード・カタログPDBで統計を収集します。

```
SQL> CONNECT wikip@catpdb
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> EXEC SYS.DBMS_STATS.GATHER_SCHEMA_STATS(OWNNAME => 'WIKIP', OPTIONS => 'GATHER');
```

4. 各シャードの統計を確認します。

```
SQL> CONNECT wikip@shdcdb0
SQL> ALTER SESSION DISABLE SHARD DDL;
SQL> SELECT t.table_name AS "Table Name",
       t.num_rows AS "Rows",
       t.avg_row_len AS "Avg Row Len",
       Trunc((t.blocks * p.value)/1024) AS "Size KB",
       sharded,
       duplicated,
       to_char(t.last_analyzed,'DD/MM/YYYY HH24:MM:SS') AS "Last Analyzed"
FROM user_tables t,
     v$parameter p
WHERE table_name not like 'MLOG%'
      AND table_name not like 'RUPD%'
      AND table_name not like 'USLOG%'
      AND table_name not like 'DR$%'
      AND p.name = 'db_block_size'
ORDER by t.table_name
/
```

Table Name	Rows	Avg Row Len	Size KB	S	D	Last Analyzed
ARTICLES	1524331	2860	5952480	Y	N	30/05/2024 10:05:27
ARTICLES_EXT				N	N	

統計が収集されて検証されたら、自動統計収集を構成できます。シャード・データベースでの自動統計収集の構成方法などの詳細については、「[シャード表のオブティマイザ統計の収集](#)」を参照してください。統計収集パラメータの追加的なチューニング、関連する最適化、およびメンテナンス計画のチューニングは、この例の範囲を超えていますが、必要に応じて組み込むことができます。

シャードJSONデータの索引付け

データへのアクセスを最適化するために、いくつかの索引が作成されます。主キーのファンクション索引と、各JSONドキュメントのカテゴリ配列の複数値索引の作成について示します。次に、各JSONドキュメント内の値の全文に対してOracle Text検索索引を作成します。複数値索引は、JSONデータベース機能の用途を示すとともに、カテゴリ・ファセット値に基づいてフィルタする検索結果問合せの実行を最適化するのに役立ちます。

Oracle Text検索索引は、JSONコードとの照合ではなく、JSONドキュメントの要素値のコンテキストによる全文検索をサポートします。

これらの索引は、GDSCTL SQLコマンドを使用してシャード・カタログまたはシャード・ディレクトラから作成され、すべてのシャードに索引が作成されるようになります。シャード・ディレクトラでGDSCTL SHOW DDLを使用して各DDL実行を検証し、障害を検出します。PL/SQLが必要な場合は、シャード・カタログからSYS.EXEC_SHARD_PLSQL()関数を使用して、実行をすべてのシャードに伝播できます。

シャード・カタログでSQLを介してDDLを実行した場合、シャード・カタログに索引が作成されるとすぐにプロンプトが返されます。ただし、各シャードへの索引の移入には数分か、場合によってはさらに長い時間がかかる場合があります。

索引作成のステータスを監視するには、各シャードで「Oracle Text索引のステータスの検証」セクション（後述）のSQLを定期的に行って、索引作成のステータスを確認します。GDSCTL SHOW DDLを使用して各DDL実行を検証すると、各シャードで正常に完了したことが表示されるようになることに注意してください。

JSONデータの索引付けに関する詳しいドキュメントについては、「[JSONデータの索引](#)」を参照してください。

主キーの索引の作成

主キー/シャーディング・キー列の索引を追加しました。これは標準的なリレーショナル・データベース操作であり、デプロイメントの完全な表現を示すために含まれています。

```
SQL> CREATE INDEX article_id_idx ON articles(article_id) LOCAL;
```

JSON配列値の複数値索引の作成

複数値索引（MVI）の主な用途は、配列内のスカラー値に索引を付けることですが、オブジェクト配列要素のスカラー・フィールド値を含めることもできます。

複数の値をターゲットとする問合せの場合、複数値索引はBツリー索引やビットマップ索引よりもパフォーマンスが良好な可能性があります。単一の値がターゲットの場合、Bツリーまたはビットマップ索引の方がMVIよりもパフォーマンスが優れている可能性があります。SQLオプティマイザは、SELECT文のWHERE句にJSON_EXISTS()関数の使用が含まれている場合にのみMVIを使用します。シャード・データベース環境では、MVIはローカル索引として作成する必要があります。

今回のサンプル・データの場合、Articles_JSON列に保存されている各JSONドキュメントには、複数の配列要素を含む\$.categories[]配列があります。各配列要素には、カテゴリの名前を含む.id値があります。これらの\$.categories[].id値は、カテゴリごとのドキュメント数を表示するために、カテゴリ・ファセット集計データ問合せで使用されます。ユーザーが1つ以上のカテゴリを選択して検索を絞り込む場合、検索結果問合せには、選択したカテゴリごとに、JSON_EXISTS()を使用したWHERE句フィルタ条件が含まれている必要があります。ここでMVIが重要になります。

例：カテゴリ配列ID要素値の複数値索引を作成するDDL

```
SQL> CREATE MULTIVALUE INDEX mvi_article_categories ON Articles a  
      (a.article_json.categories.id.string()) local;
```

Oracle Text索引の作成

JSONコンテンツの全文検索を容易にするために、JSON列にOracle Text検索索引を作成します。まず、索引を作成する前に、必要に応じてテキスト・プリファレンスを作成します。Oracle Text索引は、必要なプリファレンス属性を設定することで、個別のニーズを満たすように調整できます。Oracle Textプリファレンスは、一度定義すると、すべてのシャードに一貫して適用されます。

詳しいドキュメントについては、[JSON開発者ガイド](#)の「[非定型の問合せおよび全文検索のためのJSON検索索引](#)」セクションと、[Oracle Textアプリケーション開発者ガイド](#)の「[Oracle Text索引の作成](#)」セクションを参照してください。

例：Oracle Textプリファレンスを作成するためのSQL

SQL> begin

```
sys.exec_shard_plsql('ctxsys.ctx_ddl.create_preference("wl", "BASIC_WORDLIST"));
sys.exec_shard_plsql('ctxsys.ctx_ddl.set_attribute("wl", "FUZZY_MATCH",          "ENGLISH"));
sys.exec_shard_plsql('ctxsys.ctx_ddl.set_attribute("wl", "FUZZY_SCORE",        "60"));
sys.exec_shard_plsql('ctxsys.ctx_ddl.set_attribute("wl", "FUZZY_NUMRESULTS",   '5000'));
sys.exec_shard_plsql('ctxsys.ctx_ddl.set_attribute("wl", "WILDCARD_INDEX",      'TRUE'));
```

end;

/

Oracle Textのワードリスト・プリファレンスを設定したら、Oracle Text検索索引を作成します。

Oracle Text索引を作成するためのSQLの例：

SQL> create search index articles_idx on Articles(article_json) for json

```
parameters (' wordlist wl
             search_on text_value_string
             memory 170M
             maintenance auto
             optimize (auto_daily parallel 1)
             ')
parallel 6
local;
```

ワードリスト・プリファレンスに加えて、いくつかのパラメータはCREATE SEARCH INDEX呼出しで直接設定されます。これらは、各実装の個別のニーズに合わせて適切に分析および調整する必要があります。詳細については、「[Oracle Text索引の作成](#)」のドキュメントを参照してください。

示されているパラメータと値については、次のとおりです。

- SEARCH_ON TEXT_VALUE_STRING - 完全なJSONコードではなく、JSON要素値に対して索引が作成されます。
- MEMORY 170M - ハードウェアへの適度な割当て。指定した並列度を念頭に置きながら、ハードウェアとデータセットをテストして調整します。
- MAINTENANCE AUTO - 個別のスケジュールでスケジュールされたジョブを使用する従来の“手動”アプローチではなく、シャード・データのパーティション全体での索引の自動同期を実現します。これはOracle Database 23aiの新機能です。
- OPTIMIZE (auto_dailyParallel 1) - 今回のデータセットがエンドユーザーから更新を取得していないことを考慮すると、ベースライン構成は今回のテスト・シナリオには十分過ぎます。
- PARALLEL 6 - 今回のテスト環境データベースには8つのCPUコアがあり、他の使用はありません。いくつかのプロセスを索引付けから切り離すことで、CPU競合なしに他のデータベース・プロセスやシステム・プロセスをスケジュールできるようになります。空きCPUリソースを評価し、それに応じて調整します。

Oracle Text索引のステータスの検証

シャード上でいくつかの問合せを実行すると、索引パーティション、表、ジョブが適切に作成され、正しく機能していることを検査および検証するのに役立ちます。

デフォルトのシャード・データベース構成では、トポロジに追加されるシャードの数に関係なく、シャード・データ用に120個の“チャンク”が作成されて割り当てられることに注意してください。表領域セットが作成されたとき、すべてのシャードに表領域とパーティションがチャンクごとに作成されましたが、そのシャードに割り当てられたチャンクのみデータが移入されます。それでも、Oracle Text索引は、表領域セット内のすべての表領域のすべてのパーティションを処理する必要があります。

今回の例では、4つのシャードとデフォルトの120個のチャンクにより、表領域セットには任意のシャード上のデータの25 %のみが含まれ、Oracle Text索引により、120個のローカル索引パーティションのうち30個のみデータが移入されます。

これらのSQL文を各シャードで実行する場合は、SQLセッションを変更してシャードDDLを無効にします。

```
SQL> ALTER SESSION DISABLE SHARD DDL;
```

表領域のサイズの問合せ

デフォルトのシステム表領域に加えて、明示的に宣言された表領域セットに対応して表示されている1行と、シャードに割り当てられた“チャンク”ごとの追加の表領域がいくつかあります。

今回の場合の例のシャードでは、これにより、TSP_SET_WIKIPが1個追加され、チャンクごとの表領域（C001～C00UTSP_SET_WIKIP）が30個追加されています。2番目のシャードにもTSP_SET_WIKIPがあり、表領域C00VTSP_SET_WIKIPから始まる、チャンクごとの表領域がさらに30個あります。

```
SQL> column "Tablespace" format a24
SQL> column "Used MB" format 99,999,999
SQL> column "Free MB" format 99,999,999
SQL> column "Total MB" format 99,999,999
SQL> select
    fs.tablespace_name                    "Tablespace",
    (df.tablespace - fs.freespace)        "Used MB",
    fs.freespace                          "Free MB",
    df.tablespace                         "Total MB",
    round(100 * (fs.freespace / df.tablespace)) "Pct.Free"
from
    (select tablespace_name, (sum(bytes) / 1048576) TotalSpace
     from dba_data_files
     group by tablespace_name
    ) df,
    (select tablespace_name, round(sum(bytes) / 1048576) FreeSpace
     from dba_free_space
     group by tablespace_name
    ) fs
where
    df.tablespace_name = fs.tablespace_name
order by df.tablespace_name;
```

Tablespace	Used MB	Free MB	Total MB	Pct.	Free
C001TSP_SET_WIKIP	1,616	158	1,774		9
C002TSP_SET_WIKIP	1,616	158	1,774		9
C003TSP_SET_WIKIP	1,540	234	1,774		13
(24行を省略…)					
C00STSP_SET_WIKIP	1,532	242	1,774		14
C00TTSP_SET_WIKIP	1,608	166	1,774		9
C00UTSP_SET_WIKIP	1,608	166	1,774		9
SYSAUX	1,015	75	1,090		7
SYSTEM	328	2	330		1
TSP_SET_WIKIP	13,118	7,362	20,480		36
UNDOTBS1	277	17,117	17,394		98
USERS	1	4	5		80

スキーマ・ユーザーの表、ステータス、行数のリストの問合せ

表ファミリー内の表、データ・ロードに使用された外部表（削除されていない場合）、およびOracle Text索引のいくつかの表（索引に指定したプリファレンスによって異なります）に対応して返される行が存在する必要があります。

これらの“ドル記号”表にはさまざまなタイプの索引データが含まれていますが、その詳細については、この技術概要では説明しません。表が“VALID”（有効）ステータスで作成されており、このセクションの他の問合せで障害なしと示される場合は、すべて問題ありません。

```
SQL> COL OWNER FORMAT A16
SQL> COL TABLE_NAME FORMAT A32
SQL> SELECT
      OWNER, TABLE_NAME, IOT_TYPE, STATUS, NUM_ROWS
    FROM DBA_TABLES
   WHERE OWNER='WIKIP'
   ORDER BY TABLE_NAME;
```

OWNER	TABLE_NAME	IOT_TYPE	STATUS	NUM_ROWS
WIKIP	ARTICLES		VALID	1524331
WIKIP	ARTICLES_EXT		VALID	
WIKIP	DR\$ARTICLES_IDX\$B		VALID	
WIKIP	DR\$ARTICLES_IDX\$C		VALID	
WIKIP	DR\$ARTICLES_IDX\$G		VALID	
WIKIP	DR\$ARTICLES_IDX\$I		VALID	
WIKIP	DR\$ARTICLES_IDX\$K		VALID	
WIKIP	DR\$ARTICLES_IDX\$KG		VALID	
WIKIP	DR\$ARTICLES_IDX\$N		VALID	
WIKIP	DR\$ARTICLES_IDX\$Q		VALID	
WIKIP	DR\$ARTICLES_IDX\$SN		VALID	
WIKIP	DR\$ARTICLES_IDX\$ST		VALID	
WIKIP	DR\$ARTICLES_IDX\$SV		VALID	
WIKIP	DR\$ARTICLES_IDX\$U		VALID	

14 rows selected.

索引のステータスの問合せ

どのステータス値もFAILEDとして報告されていないことを確認します。

```
SQL> SELECT
      OWNER, INDEX_NAME, INDEX_TYPE, TABLE_NAME,
      STATUS, DOMIDX_STATUS, DOMIDX_OPSTATUS, FUNCIDX_STATUS,
      DROPPED, INDEXING, DEGREE, INSTANCES, PARTITIONED, BUFFER_POOL
    FROM SYS.DBA_INDEXES
   WHERE INDEX_NAME = 'ARTICLES_IDX'
   ORDER BY 1, 2;
```

OWNER	INDEX_NAME	INDEX_TYPE	TABLE_NAME	STATUS
WIKIP	ARTICLES_IDX	DOMAIN	ARTICLES	VALID
DOMIDX_STATUS DOMIDX_OPSTATUS FUNCIDX_STATUS				
VALID	VALID			
DROPPED	INDEXING	DEGREE	INST	PAR BUFFER_POOL
NO	FULL	6	1	YES DEFAULT

パーティション・ステータスの問合せ

この問合せにより、ステータス列の値によってグループ化されたパーティションの数を取得します。

すべてのパーティションが、ステータスはUSABLE、DOMIDX_OPSTATUSはVALIDまたはINPROCESSと報告する必要があります。
パーティションの合計数は、シャード・データベースに構成されたチャンクの数と一致する必要があります。

```
SQL> SELECT INDEX_NAME, STATUS, DOMIDX_OPSTATUS, COUNT(*)
      FROM DBA_IND_PARTITIONS
      WHERE INDEX_NAME IN ('ARTICLES_IDX')
      GROUP BY INDEX_NAME, STATUS, DOMIDX_OPSTATUS
      ORDER BY 1,2,3;
```

INDEX_NAME	STATUS	DOMIDX_OPSTATUS	COUNT(*)
ARTICLES_IDX	USABLE	VALID	120

索引パーティションと詳細のリストの問合せ

この問合せにより、120個すべての索引パーティション、そのステータス、索引付きドキュメント数、および同期の詳細を一覧表示します。

“MAINTENANCE AUTO”を使用する場合、手動でスケジュールされた同期ジョブに関する列は省略できます。

これには、IXP_SYNC_PARA_DEGREE、IXP_SYNC_INTERVAL、およびIXP_SYNC_JOB_NAMEが含まれます。

このシャードでは、パーティション1〜30にドキュメントが移入されていることに注意してください。これは、シャード・データベース構成でシャードに割り当てられたチャンクに対応します。これらの割り当てられたチャンクは、シャード・ディレクタからGDSCTL CONFIG CHUNKSを使用して表示できます。

```
SQL> SELECT IXP_ID, IXP_INDEX_OWNER, IXP_INDEX_NAME, IXP_INDEX_PARTITION_NAME,
           IXP_STATUS, IXP_DOCID_COUNT, IXP_SYNC_TYPE, IXP_SYNC_MEMORY
           FROM CTXSYS.CTX_INDEX_PARTITIONS
           WHERE IXP_INDEX_NAME IN ('ARTICLES_IDX')
           ORDER BY 2, 3, 1;
```

IXP_ID	IXP_INDEX_OWNER	IXP_INDEX_NAME	IXP_INDEX_PARTIT	IXP_STATUS	IXP_DOCID_COUNT	IXP_SYNC_TYPE	IXP_SYNC_MEM
1	WIKIP	ARTICLES_IDX	ARTICLES_P4	INDEXED	50999	MANUAL	178257920
2	WIKIP	ARTICLES_IDX	ARTICLES_P3	INDEXED	50746	MANUAL	178257920
3	WIKIP	ARTICLES_IDX	ARTICLES_P6	INDEXED	51040	MANUAL	178257920
(4行目から28行目を省略)							
29	WIKIP	ARTICLES_IDX	ARTICLES_P29	INDEXED	50833	MANUAL	178257920
30	WIKIP	ARTICLES_IDX	ARTICLES_P30	INDEXED	50687	MANUAL	178257920
31	WIKIP	ARTICLES_IDX	ARTICLES_P31	INDEXED	0	MANUAL	178257920
(32行目から119行目を省略)							
120	WIKIP	ARTICLES_IDX	ARTICLES_P120	INDEXED	0	MANUAL	178257920

検索問合せの設計と例

完全に実現されたテキスト検索ソリューションには、検索リクエストをフィルタリングして絞り込むためのいくつかの機能が含まれる場合があります。

今回の例では、いくつかのユースケースを概説し、UI設計のモックアップを示し、実装に必要なSQL問合せの設計について伝えるための基礎として、いくつかのユーザー・エクスペリエンスについて説明します。問合せの例と、それらがさまざまなユースケースの要件を満たすためにどのように変更されるかが、推奨されるベスト・プラクティスも含めて示されます。

検索フォームの例

前に示した抽象的な検索フォームのモックアップは、ここで詳しく説明しています。この設計には、ドキュメント・セットの特定のカテゴリと言語メタデータに基づいた、基本的なテキスト検索機能、日付フィルタリング、ファセット検索が含まれています。

集計されたファセット選択リストは、キーワードによる最初の検索が送信されてから移入する必要があります。送信された日付も、問合せに対する追加のフィルタ条件として機能します。カスタマイズ用にページ区切りパラメータも提供されます。ページ・サイズと現在のページ番号は、リクエストごとに取得されるデータを制限するために、検索結果問合せで使用されます。

The mock-up shows a search interface with the following components:

- Header:** A box labeled "LOGO" and the title "Faceted Search UI Mock-Up".
- Search Bar:** Contains the text "Orange and fruit" and a magnifying glass icon. To its right is a "Date Range" section with two date input fields (format MM/DD/YYYY) and a "to" separator, each with a calendar icon.
- Filters (Left Panel):**
 - Categories (31):** A list of checkboxes for Category A through J, with a "(more)" link below.
 - Language (4):** A list of checkboxes for English, French, German, and Hindi, each followed by a count in parentheses.
 - Update Results:** A button at the bottom of the filter panel.
- Results (Right Panel):**
 - View:** A link to "View (previous 20 | next 20)" and a list of view counts: "(20 | 50 | 100 | 250 | 500)".
 - Item 1:** "Orange (fruit)" with a placeholder image, description, and "last modified: 18:36, 12 August 2023".
 - Item 2:** "Mandarin orange" with a placeholder image, description, and "last modified: 18:36, 12 August 2023".
 - Item 3:** "Blood orange" with a placeholder image, description, and "last modified: 09:23, 19 July 2023".
 - Item 4:** "Bitter orange" with a placeholder image, description, and "last modified: 08:58, 21 July 2023".
 - Item 5:** "Bergamot orange" with a placeholder image, description, and "last modified: 23:28, 19 July 2023".

検索フォームのレイアウトとパラメータ：

UIには、今回の問合せ設計に関連するいくつかのフォーム・パラメータがある3つのパネルが含まれています。フォームを送信するメカニズムは、アプリケーション開発チームに任されています。

1. 3つのフォーム要素がある水平検索バー：
 - a. 自由形式入力用の“search_term”テキスト・ボックス
 - b. 日付を提供するための“start_date”要素（テキスト/カレンダー・ポップアップなど）
 - c. 日付を提供するための“end_date”要素（テキスト/カレンダー・ポップアップなど）
2. 2つの‘ファセット’検索フィルタのフォーム要素が含まれる、左側の検索フィルタ・パネル：
 - a. カテゴリ別に集計された結果数
 - i. ヘッダーと、見つかったカテゴリの合計数が表示されます
 - ii. カテゴリごとにチェック・ボックスがあり、それぞれの記事の集計数が表示されます
 - b. 言語別に集計された結果数
 - i. ヘッダーと、見つかった言語の合計数が表示されます
 - ii. 言語ごとにチェック・ボックスがあり、それぞれの記事の集計数が表示されます
3. 右側の検索結果パネル
 - a. Oracle Textの“スコア”ランキングによってソートされた検索結果の部分的なリストが表示されます。各検索結果には以下が表示されます
 - i. Title
 - ii. 検索語句を含むテキスト・スニペット
 - iii. 最終変更日
 - iv. 検索結果のスコア（オプションであり、ここには表示されません。ソートにのみ使用されます）

ユースケース

ここで示す、サンプル・アプリケーションによるテキスト検索のユースケース・シナリオでは、多くの検索インタフェースに共通する機能を紹介しています。これらの機能により、今回の問合せの設計面を明確に伝えることができます。

1. 基本的なキーワードまたはフレーズを使用して検索する

追加のフィルタを使用せずに、単純なテキスト検索フレーズを送信します。初期ファセットと検索結果データが表示されます。

2. 日付フィルタを使用して検索する

検索語句に加えて、開始日フィルタと終了日フィルタの一方または両方を構成し、フォームを送信します。更新されたファセットと検索結果データが表示されます。

3. ファセット検索

ファセットから1つ以上の値を選択し、フォームを送信します。アプリケーションにより、検索結果が更新されます。

4. 結果のページング

検索結果ページのナビゲーション・リンクをクリックして、表示される結果ページを変更します。

アプリケーションにより、リクエストした結果ページで検索結果が更新されます。

5. 結果の数を変更する

検索結果パネルで優先ページ・サイズを更新します。

アプリケーションにより、同じ箇所から開始して、リクエストした結果数で検索結果が更新されます。

6. ドキュメント・ビュー

特定のドキュメントの検索結果リンクをクリックすると、JSONドキュメントのレンダリング・ビューが表示されます。

ページが更新され、検索フォームがドキュメント・ビューに置き換えられます。

問合せの設計

アプリケーションの検索UIフォーム要素の送信により、アプリケーション・ロジックが駆動し、指定したパラメータに基づいてWHERE句フィルタ条件が変更されます。SQL問合せは、ユーザー入力に基づき、バインド変数を使用して設計されています。

今回の検索問合せは“複数シャード問合せ”であり、必ずシャード・データベース全体にわたって結果を検索します。これは、システム管理のシャード・データベースに適しています。すべての複数シャード問合せは、シャード・カタログ・データベースのコーディネータに送信されます。

例外は、ユーザーが、別の方法で処理される特定のドキュメント・リンクをクリックする、6番目のユースケース・シナリオです。シャード・キーに基づいて単一の記事を取得するための最終的な検索結果のクリックスルー問合せでは、シャード・ディレクタを通じて構成された、読取り/書込みまたは読取り専用のOLTPサービスを使用できます。

最初の5つの各ユースケースでは、特定の組合せのフィルタ条件を使用して、さまざまな組合せの問合せを送信する必要があります。フィルタ条件の構文は、すべての問合せで同じになります。SQL問合せの例では、効率化を目的として、ユーザーがユースケース1、2、3を反復済みであるかのように、すべてのフィルタ条件を所定の位置で表示しています。

それぞれの問合せの簡単な説明、SQLコードの例、結果の例、例示した機能と手法の説明を示しています。SQL問合せでは、すべてのフィルタ条件が表示され、それにはWHERE句にさまざまな条件が必要とされるさまざまなユースケースの並べ替えについて説明するコメントがあります。

注：ビューのページ区切りで検索結果を管理するユースケース4と5。UIの合計ドキュメント数とページングの詳細を取得するための問合せを実装するかどうかは、読者の判断に委ねられています。

ユーザーが検索語句を更新し、ファセット値または日付を選択し、ファセット検索ページ・フォームを再送信すると、アプリケーションは必ず検索結果問合せを動的に導出し、送信された入力値に特に一致する適切な句を取得します。

SQL問合せの概要

主要な問合せが3つあり、表示される各パネルに1つずつ対応しています。各問合せは、各着信検索リクエストのプロパティに基づいて、適切なフィルタ条件を含めるように調整する必要があります。

まず、各問合せとそれぞれのフィルタ句の包括的な概要を説明します。コンテキスト全体を示したら、使用される個々の句とSQLの機能および関数について、ベスト・プラクティスの推奨事項とともに詳しく説明します。

主要な問合せには、次のものがあります。

1. 検索結果の問合せ - スコアリングされた関連性ランキングとドキュメント・スニペットを含む
2. ファセット検索の問合せ - 集計されたドキュメント数を含むJSONカテゴリ配列値
3. ファセット検索の問合せ - 集計されたドキュメント数を含むJSON言語要素値

追加のフィルタリング・オプションが提供される場合、問合せにはアプリケーション・コード内でリクエストごとに追加のフィルタを含める必要があります。これらのフィルタ条件は主要な問合せの例に含まれており、個別に説明されています。これらには、次によるフィルタリングが含まれます。

- A. 日付
- B. カテゴリ語句
- C. 言語

バインド変数

バインド変数を使用して問合せフィルタ条件で使用する値を宣言すると、ユースケースごとに各SQL文で単一のカーソルを使用できるようになります。これにより、本質的に同じであるものの、パラメータ値のハードコードされたデータでn個の組合せを持つSQL文にメモリを割り当てたり、その最適化に時間を費やしたりする必要がなくなります。ユーザー集団が送信できるすべての検索語句と日付範囲の組合せの数を想像してみてください。リテラル問合せからこれらの値を削除すると、カーソルの数、メモリと処理のオーバーヘッド、そして最終的にはパフォーマンスとスケーラビリティに非常に大きな影響を与えます。参考として、「[Oracle Real-World Performanceのアプリケーションの設計](#)」のドキュメントを参照してください。

注：アプリケーションは、SQL実行用のバインド変数値を設定する前に、厳密なフィールド・データ検証を行う必要があります。

今回のソリューション例において、検索問合せの最適化は、複数のバインド変数宣言の利点を生かしています。ユーザーによって送信されたフォーム・データから直接解析された単純な値もあれば、特定のSQL関数パラメータ句または計算値に構成された、より複雑な値もあります。

- 検索語句と日付ベースの変数は、明示的な入力データの単純な文字列値です。
- 検索結果問合せに通知するページ区切りパラメータは、同様に単純に送信された値である場合もあれば、要件に応じて計算される場合もあります。たとえば、問合せで必要なオフセット値は、ページあたりの行数の値とページ番号の積によって算出されます。
- ファセット・パラメータは複数の値の場合があるため、単一のバインド変数内で使用するには何らかの書式設定が必要です。このような場合、アプリケーションでは、必要なSQLに基づいて完全なフィルタ句パラメータまたはパス式文を構築することをお勧めします。
 - 言語仮想列の場合、引用符付き文字列値の単純なカンマ区切りリストを必要とする“IN”句を使用できます。
 - カテゴリ・ファセット値の場合は、送信されたフォームcategory_id値を使用して、フィルタを含む完全修飾パス式を生成します。
 - 例：SQL*Plusのバインド変数宣言

REM ユーザー・リクエスト・フォーム入力に基づいて今回の検索問合せのバインド変数を設定します

REM

REM 自由形式の検索語句テキスト入力を含むメイン検索ヘッダー・フォーム・データと、
開始日および終了日のセレクト

```
define s_searchterm = 'Orange and fruit';
```

```
define s_datestart = '2021-11-10';
```

```
define s_dateend = '2021-11-20';
```

REM カテゴリ・ファセット値

REM フォーム要素: "FacetCategories"チェック/ボックスには以下が含まれます

```
REM "Taxonomy_articles_created_by_Polbot"
```

REM および

```
REM "Taxa_named_by_Carl_Linnaeus"
```

REM categories[]配列のidフィールドに完全一致する各個別値（排他OR）を取得します

REM JSON_EXISTS()のフィルタを含むパス式として書式設定します

```
define s_filtercat = '$.categories[*]?(@.id == "Taxonomy_articles_created_by_Polbot" || @.id == "Taxa_named_by_Carl_Linnaeus")';
```

REM

REM 言語ファセット値

REM フォーム要素: "FacetLang"チェック/ボックスには以下が含まれます

```
REM "EN"
```

REM および

```
REM "HI"
```

REM "IN"フィルタ句で使用する複数の値のバインド変数値を書式設定します

```
define s_filterlang = '"EN","HI"';
```

REM

REM ページング・パラメータ

REM 表示する行数

REM オフセットする行数（アプリ内で、表示する行数×ページ番号により計算します）

```
define n_pgrows = 3;
```

```
define n_pgoffset = 0;
```

検索結果の問合せ

今回の例の検索問合せでは、仮想列とJSON列の両方からデータを選択し、Oracle Textを使用して必要な結果を取得します。記事IDとタイトル（仮想列）データの基本的な選択に加えて、問合せは、ランク付けされた検索スコアと、各検索結果の各ドキュメントのコンテンツに関連する短いHTMLスニペットを返します。

アプリケーションの検索フォームで更新を送信した後、アプリケーションは、提供された入力フォーム要素の値に基づいて適切な句を含む検索結果問合せを構築し、問合せを実行して、結果ビューに表示するために結果を書式設定する必要があります。条件句を含めて、提供された特定の組合せの入力に基づいてバインド変数を設定するには、問合せを動的に組み立てる必要があります。

以下の問合せには2つのセクションがあります。

- 1) 内側のSELECT文は、一致するすべての行を返し、検索ランク・スコアを設定し、ドキュメント・スニペットのコンテンツを構築します。これらの“内側の”問合せ操作は、処理のためにコーディネータから各シャードに並列で送信されます。
- 2) もっとも外側のSELECT文操作は、結果セットのページングを目的とした最終的なソートとフィルタリングを実行します。これには、内側の問合せ操作が完了してデータがコーディネータに返された後にすべてのシャードから返される結合データが含まれます。

例：スコアリング済み検索結果の問合せ - すべてのパラメータ

```
SELECT /*+ QB_NAME(qb_paging)*/
  article_id,
  title,
  last_modified,
  search_score,
  doc_snippet
FROM
  (
    SELECT /*+ QB_NAME(search_result_uc3) DOMAIN_INDEX_SORT FIRST_ROWS(100) PARALLEL(4) */
      a.article_id,
      a.title,
      a.last_modified,
      greatest(score(1),score(2)) AS search_score,
      ctx_doc.policy_snippet('articles_idx',
                             json_serialize(a.article_json.paragraphs RETURNING clob),
                             '&s_searchterm') AS doc_snippet
    FROM
      articles a
    WHERE
      (
        JSON_TEXTCONTAINS(a.article_json, '$.title', '&s_searchterm', 1)
        OR JSON_TEXTCONTAINS(a.article_json, '$.paragraphs', '&s_searchterm', 2)
      )
      AND a.last_modified BETWEEN to_utc_timestamp_tz('&s_datestart')
                             AND to_utc_timestamp_tz('&s_dateend')
      AND JSON_EXISTS ( a.article_json, '&s_filtercat' )
      AND a.lang IN ( &s_filterlang )
    ORDER BY
      greatest(score(1),score(2)) DESC
  )
OFFSET &n_pgoffset ROWS FETCH NEXT &n_pgrows ROWS ONLY;
```

例：返されたスコアリング済み検索結果データ

ARTICLE_ID	TITLE	DOC_SNIPPET	SEARCH_SCORE
191214	Mandarin orange	Short_description, Small citrus fruit]\nTEMPLATE[speciesbox, name = Mandarin orange, image = Citrus reticulata April...mandarine, is a small citrus tree fruit.Treated as a distinct species of orange, it is usually eaten plain or in fruit salads.TEMPLATE[cite	100
58875	Maclura pomifera	Osage orange, image...caption = Foliage and [[multiple fruit]], genus = Maclura, species...commonly known as the Osage orange, horse apple, hedge, or hedge...15, m, ft, -1] tall.The distinctive fruit, a multiple fruit, is roughly spherical, bumpy	100
68763241	Worldwide breakfast	and fresh fruit, including...accompanied by coffee, tea and orange juice.A typical Israeli meal...and drink milk, hot chocolate or fruit juice.Japanese adults (especially younger...)They often drink coffee or orange juice.Traditional Japanese inns	100

返されるデータ

この問合せは、JSONデータとスキーマ設計に反映されているとおり、ビジネス要件に基づいて検索結果内の各ドキュメントの表示要素を返します。今回の例では、これには次のものが含まれます。

- ドキュメント・タイトルのテキストと記事ID
これらは、各ドキュメントへのハイパーリンクを提供するために使用されます。正しいシャードから記事データを効率的に取得するには、リクエストにパラメータとしてシャード・キー（article_ID）を含める必要があります。この例では、シャード・キーはデータの主キーと一致します。
注：場合によっては、シャード・キーが主キーではないことがあります。その場合、主キーとシャード・キーの両方の列値がこの問合せによって返される必要があります。
- ドキュメントの最終変更日
- ソート目的で使用される、ドキュメントの検索結果のランク/スコア
- \$.paragraphs[]配列からのドキュメント本文テキストの"スニペット"
これにより、検索語句が強調表示された、ドキュメント・コンテンツの短いセクションが表示されます。

注目すべきSQL機能

リクエストにはさまざまな検索フォーム要素パラメータが含まれるため、結果を状況に応じて絞り込むには、対応するSQL句が問合せ内に存在する必要があります。

検索語句のテキスト

アプリケーション・ユーザーは、自由形式のテキストで検索語句を入力できます。条件演算子は使用してもしなくても構いません。そのため、SQL問合せへの値の挿入に先立って、SQLインジェクション攻撃を回避するために、アプリケーション層での慎重な検証が必要となります。

今回のサンプル・データには、検索対象のコンテンツ関連テキストを含む2つの要素（title要素とparagraph要素）がJSONデータ内に含まれています。paragraph要素は、各記事の本文コンテンツを含む、文字列要素の配列であり、title要素は単一文字列です。

アプリケーションから送信された検索語句のテキスト文字列は、JSON_TEXTCONTAINS()関数で指定された検索語句バインド・パラメータ値に使用する必要があります。JSON_TEXTCONTAINS()フィルタ条件の実行計画操作は、Oracle Text検索索引を使用するように最適化されます。

OracleではCONTAINS()関数も提供していることに注意してください。JSON型付き列を評価する場合は、このユースケース向けに特に最適化されたJSON_TEXTCONTAINS()関数を使用することがベスト・プラクティスです。

JSON_TEXTCONTAINS()関数の評価で一致を示す、ゼロ以外の結果が返される場合、その行は結果セットに追加されます。JSON_TEXTCONTAINS()関数のパラメータには、処理するJSONフィールド、“パス式”と呼ばれる論理文、およびこれらの一致のランク付けされたスコアの基になる数値スコアリング・オブジェクトが含まれます。ベスト・プラクティスは、WHERE句に必要なJSON要素ごとに、別々のJSON_TEXTCONTAINS()関数を含めることです。複数の要素値のロジックを含むパス式を作成することは可能ですが、そのようにした場合、SQLエンジンが問合せを最適化して利点を最大限に生かすことはできない可能性があります。

例：複数のJSON_TEXTCONTAINS()句

```
WHERE
    (
        JSON_TEXTCONTAINS(a.article_json, '$.title',      '&s_searchterm', 1)
    OR JSON_TEXTCONTAINS(a.article_json, '$.paragraphs', '&s_searchterm', 2)
    )
```

複数のJSON_TEXTCONTAINS()関数を使用すると、結果のランク・スコアリングは要素ごとに行われます。したがって、結果をソートするときにすべてのスコアを考慮する必要があります。複数のスコアを指定してGREATEST関数を使用すると、期待どおりの結果が得られます。

例：実際の最大スコアを取得するSQL SELECT句とORDER BY句

```
SELECT greatest(score(1),score(2)) AS search_score
ORDER BY greatest(score(1),score(2)) DESC
```

WHERE句の順序と優先順位は慎重にテストする必要があります。全文検索用に複数のJSON要素にJSON_TEXTCONTAINS()を組み込み、AND演算子を含む他の句を使用する場合は、JSON_TEXTCONTAINS()を括弧でグループ化し、OR演算子を使用する必要があります。これにより、任意の行の検索結果はどちらでも一致ようになります。追加の検索条件がAND演算子に含まれる場合、JSON_TEXT_CONTAINS()句を括弧で囲む必要があります。AND演算子はORよりも優先されます。たとえば、次の抽象化されたWHERE句について考えてみましょう。

1. WHERE A or B and C and D
これは次のように評価されます。WHERE A or (B and C and D)
2. WHERE (A or B) and C and D
目標としていたのはこれです。ここではAとBがJSON_TEXTCONTAINS()であり、それらは他のフィールドでフィルタリングされる前に常に一緒に評価されます。

検索語句を強調表示した、検索結果のHTML"スニペット"

検索語句を含む記事コンテンツの関連HTML"スニペット"の表示は、Oracle TextのCTX_DOC.POLICY_SNIPPET()関数の結果を返すことで実現できます。CTX_DOC.POLICY_SNIPPETは、問合せ語句の出現箇所を前後のテキストとともに返す、ドキュメントの"コンコーダンス"を生成します。[Oracle Text Application開発者ガイド、23ai](#)を参照してください

POLICY_SNIPPET()関数は、必要な機能に基づいて多くのパラメータを受け取ることができます。今回の基本的な例では、検索する索引、表示するデータ、検索語句の3つのパラメータが必要です。表示するJSONデータを返すとき、この例では、記事本文のコンテンツを含むparagraphs配列からスニペットを返すことを選択しています。このJSON配列データはCLOBとして返され、JSON_SERIALIZE()で処理されて、確実に表示可能な結果が得られます。[Oracle Textリファレンス、23aiドキュメント](#)のCTX_DOCのセクションを参照してください。

例：検索語句を強調表示してドキュメント・コンテンツのHTMLスニペットを取得する方法

```
ctx_doc.policy_snippet('articles_idx',
                        json_serialize(a.article_json.paragraphs RETURNING clob),
                        '&s_searchterm') AS doc_snippet
```

開始日および/または終了日フィルタ

すべての日付フォーム・フィールド値は、問合せを送信する前に検証され、日付書式にキャストされる必要があります。日付が送信されない場合、日付比較のためのWHERE句のフィルタ条件は、当然のこととして、送信前に問合せから除外する必要があります。

今回のサンプル・スキーマでは、JSON文字列値を適切な日付書式に自動的に変換するArticles.last_modified仮想列を実装しました。

日付が1つだけ送信される場合、WHERE句のフィルタ条件には単に「大なり」または「小なり」の比較が含まれます。両方の値が必要な場合は、BETWEEN()関数の方が効率的です。

ユースケース・シナリオごとに、次に示す3つのフィルタ句のうちいずれかを使用します。

```
AND a.last_modified > TO_UTC_TIMESTAMP_TZ('&s_datestart')
AND a.last_modified < TO_UTC_TIMESTAMP_TZ('&s_dateend')
AND a.last_modified BETWEEN TO_UTC_TIMESTAMP_TZ('&s_datestart')
                           AND TO_UTC_TIMESTAMP_TZ('&s_dateend')
```

ファセット・チェック・ボックスの選択された値

ユーザーは、最初の検索を完了し、ファセット問合せを実行してファセット・バケット値とドキュメント数を移入すると、任意の組合せのファセット・バケット値を選択して検索結果をさらに絞り込むことができます。ファセット値のチェック・ボックスをオンにして検索フォームを再送信した後、バケット値が返されたファセットに対し、追加のWHERE句を使用して検索結果ドキュメントの問合せを実行する必要があります。

今回の例では、フォーム上にfacetCategoriesとfacetLangという2つのファセットがあります。最初の検索後、これらにデータが移入され、以前の検索結果のドキュメントのカテゴリと言語の選択ボックス要素/値が含まれるようになります。今回の例のユーザーが、検索コンテキストを制限するために"Category:A"と"Category:B"を選択し、その後、検索フォームを再度送信する前に、英語（EN）とヒンディー語（HI）の2つの言語を選択したとします。

この場合、検索結果ドキュメントの問合せには追加のSQL WHERE句を含める必要があり、必要なJSONパス式はアプリケーションによって事前に組み立てられ、バインド変数値に設定されている必要があります。

```
define s_filtercat = '$.categories[*]?(@.id == "Category:A" || @.id == "Category:B");'
define s_filterlang = '"EN","HI"';

AND JSON_EXISTS ( a.article_json, '&s_filtercat' )
AND a.lang IN ( &s_filterlang )
```

カテゴリ・ファセット・チェック・ボックスの値の場合、各ドキュメントは、送信された2つの値のいずれかに一致する各\$.categories[].id値のJSON値によってフィルタされます。これを行うには、JSON_EXISTS()関数で使用するフィルタを構築します。このフィルタは、「すべての\$.categories配列要素について、含まれる@.id要素がリクエストされた値のいずれかと等しい場合にドキュメントと一致する」と解釈できます。articles_json.categories.id要素値に複数値索引を追加すると、パフォーマンスの最適化に役立ちます。

言語ファセットの場合、バインド変数値はJSON配列ではなく単純なスカラーJSON要素と比較されるため、仮想列とより単純なリレーショナル句を使用できます。Articles.lang仮想列の実装により、問合せ内でJSON_VALUE()関数を明示的に使用する必要がなくなりました。

ページ区切り

ユーザーは、表示する結果の数を選択したり、表示したい結果'ページ'を変更したりすることができます。すべてのシャードから返された完全マージ済み行セットのソートとページ区切りは、メイン問合せの周囲の“ラッパー”SELECT文を使用してシャード・カタログで実行されます。ページ区切りを行うには、“オフセット”機能を使用します。

内側の問合せの例には、`/* FIRST_ROWS(100) */`というオプティマイザ・ヒントがあることに注意してください。この値はおそらく、問合せのリモート・シャード実行が適切に最適化されていない場合、`(&n_pgoffset+1)*&n_pgrows`の値に設定された独自のバインド変数を使用して動的に変更する必要があります。

カテゴリ・ファセット問合せ

カテゴリ・ファセットのデータを集計する問合せは、一致するすべてのドキュメントの個別の`$category[].id`値を、ドキュメント数、次にカテゴリの順に返します。個別のカテゴリIDの単純なリストを取得するには、`$category[].id`値の列を1つ含むJSON表を宣言する必要があります。

`JSON_TABLE()`を使用する場合、操作するJSON列、データを解析するJSON配列、およびその配列への相対的なJSONパスを使用する列定義を指定します。これにより、この問合せによって返されたすべての行のJSONデータ内で見つかったすべての`$category[].id`値がJSON表の行に明確に移入されます。`article_id`数を取得するには、JSON表の`catid`列でグループ化します。

この問合せのWHERE句は、検索結果問合せと同じロジックと機能に従いますが、検索語句と日付値のみを含める必要があります。今回の例では、提示されているカテゴリ・ファセット・データと言語ファセット・データは独立しています。

例：カテゴリ・ファセットSQL問合せ

```
SELECT /*+ QB_NAME(facet_cat_uc3) FIRST_ROWS(10) PARALLEL(4) */
       jt.catid,
       COUNT(a.article_id) AS doc_count
FROM   Articles a,
       JSON_TABLE(Article_JSON,
                  '$.categories[*]' COLUMNS (catid varchar2(64) PATH '$.id'
                  null on error)) jt
WHERE  (
        JSON_TEXTCONTAINS(a.article_json, '$.title',      '&s_searchterm')
      OR JSON_TEXTCONTAINS(a.article_json, '$.paragraphs', '&s_searchterm')
      )
AND    a.last_modified BETWEEN to_utc_timestamp_tz('&s_datestart')
                          AND to_utc_timestamp_tz('&s_dateend')

GROUP BY jt.catid
ORDER BY doc_count DESC, jt.catid;
```

例：カテゴリ・ファセット・データセット

CATID	DOC_COUNT
Taxa_named_by_Carl_Linnaeus	246
Flora_of_New_South_Wales	200
Flora_of_Queensland	179
Fungi_of_North_America	148
Taxonomy_articles_created_by_Polbot	144

言語ファセット問合せ

言語ファセット問合せの設計はカテゴリ・ファセット問合せの設計と同じですが、`JSON_TABLE`は、ドキュメントごとに単一の文字列値を含む`$.lang` JSON要素に直接構築されます。オプションで、言語ファセット問合せは、リレーショナル問合せ用に以前に作成したlang仮想列を利用することもできます。これにより、`JSON_TABLE()`関数評価の呼出しと解析の複雑さが解消され、クライアント側のコードが簡素化されます。両方のアプローチが提供されます。

例：JSON_TABLE()を使用した言語ファセットSQL問合せ

```

SELECT /*+ QB_NAME(facet_lang_uc3a) PARALLEL(4) */
      jt.lang,
      COUNT(a.article_id) AS doc_count
FROM   Articles a,
      JSON_TABLE(Article_JSON,'$.lang' COLUMNS (lang PATH '$' null on error)) jt
WHERE
      (
        JSON_TEXTCONTAINS(a.article_json, '$.title',      '&s_searchterm')
      OR JSON_TEXTCONTAINS(a.article_json, '$.paragraphs', '&s_searchterm')
      )
      AND a.last_modified BETWEEN to_utc_timestamp_tz('&s_datestart')
                                AND to_utc_timestamp_tz('&s_dateend')

GROUP BY jt.lang
ORDER BY doc_count desc jt.lang;

```

例：カテゴリ・ファセット・データセット

LANG	DOC_COUNT
EN	851
HI	1

例：仮想列を使用した言語ファセットSQL問合せ

```

SELECT /*+ QB_NAME(facet_lang_uc3b) PARALLEL(4) */
      a.lang,
      COUNT(a.article_id) AS doc_count
FROM   Articles a,
WHERE
      (
        JSON_TEXTCONTAINS(a.article_json, '$.title',      '&s_searchterm')
      OR JSON_TEXTCONTAINS(a.article_json, '$.paragraphs', '&s_searchterm')
      )
      AND a.last_modified BETWEEN to_utc_timestamp_tz('&s_datestart')
                                AND to_utc_timestamp_tz('&s_dateend')

GROUP BY a.lang
ORDER BY doc_count desc, a.lang;

```

例：言語ファセット・データセット

LANG	DOC_COUNT
EN	851
HI	1

まとめ

オラクルのネイティブな問合せ機能を使用して大規模な全文検索を実現するために、データをOracleデータベースに保存し、Oracle Text、索引、JSON検索機能、およびOracle Globally Distributed Databaseアーキテクチャの能力を組み合わせる方法について説明しました。

これにより、次のことができるようになります。

- データ・リクエストと検索リクエストに単一の接続モデルを使用することで、アプリケーションの複雑さを軽減
- 他のサードパーティのドキュメント・ストアの要件を排除し、運用コストを削減
- 大規模なデータセットと極端なワークロードをサポートするためのスケール・アウト
- Oracle Databaseソリューションに固有の高可用性を活用

付録A : 高可用性とディザスタ・リカバリ

Oracle Maximum Availability Architectureベスト・プラクティスは、Globally Distributed Databaseに適用されます。

レプリケーション・テクノロジーの観点では、Oracle Globally Distributed Databaseには、Raftコンセンサス・アルゴリズムを使用した、組み込みのRaftレプリケーション・テクノロジーが含まれています。Oracle Data Guardに緊密に統合されてもいます。シャード・データベースにはどちらのレプリケーション・テクノロジーも選択できます。Oracle Data Guardは、シャード・カタログ・データベースのレプリケーションに使用できます。

参考資料 :

- [Oracle Globally Distributed Databaseでのレプリケーション](#)
- [シャード・データベースのデプロイの概要](#)
- [Raftレプリケーションの構成および管理](#)
- [Oracle MAAベスト・プラクティス - Oracle Database](#)

付録B：シャード・データの分散方法

Oracle Globally Distributed Databaseは、“シャード”と呼ばれる複数の独立した物理データベースにわたってのデータの水平パーティション化に基づくスケーリング手法を使用します。データの分散はアプリケーションに対して透過的に行われます。

シャード表としてデPLOYされた表は、標準のOracle Databaseパーティション化機能と同様の方法でパーティション化されますが、それに加えて、明確に分離されたシャード・データベース全体の特定のパーティションにデータが分散されます。データをシャード全体に分散する方法は、設計の目標と要件に合わせて選択できます。包括的な概要については、『Oracle Globally Distributed Databaseガイド』ドキュメントの「[概要](#)」セクションを参照してください。

Oracle Globally Distributed Databaseは表パーティション化に基づいているため、Oracle Databaseが提供するすべてのサブパーティション化方法もOracle Globally Distributed Databaseによりサポートされています。

データ分散方法によって、シャード上のデータの配置が制御されます。Oracle Globally Distributed Databaseは、システム管理のシャーディング、ユーザー定義のシャーディング、ディレクトリベースのシャーディング、コンポジット・シャーディングの各方法をサポートしています。

- **システム管理**：データをシャードにマッピングする必要はありません。データは、コンシステント・ハッシュを使用したパーティション化により、自動的にシャード全体に分散されます。パーティション化アルゴリズムにより、シャード全体に均一かつランダムにデータが分散されます。

詳細については、『Oracle Globally Distributed Databaseガイド』ドキュメントの「[システム管理のシャーディング](#)」セクションを参照してください。

- **ユーザー定義**：個々のシャードへのデータのマッピングを明示的に指定できます。これは、パフォーマンス、規制、またはその他の理由により、特定のデータを特定のシャードに保存する必要があり、管理者がシャード間のデータ移動を完全に制御する必要がある場合に使用されます。

詳細については、『Oracle Globally Distributed Databaseガイド』ドキュメントの「[ユーザー定義のシャーディング](#)」セクションを参照してください。

- **コンポジット**：2レベルのパーティション化を使用できます。まず、データは範囲またはリストによってパーティション化され、次に、コンシステント・ハッシュによってさらにパーティション化されます。

多くのユースケース、特にデータ主権とデータ近接性の要件の場合、コンポジットの方法はシステム管理の方法とユーザー定義の方法の両方の長所を提供し、目的とする自動化と、データ配置に対して必要とする制御が実現します。

詳細については、『Oracle Globally Distributed Databaseガイド』ドキュメントの「[コンポジット・シャーディング](#)」を参照してください。

- **ディレクトリベース**：ユーザー定義の方法の拡張機能。シャーディング・キーに関連付けられたデータ・レコードの場所は、ユーザー・プリファレンスに基づいて実行時に動的に指定されます。キーの場所情報はディレクトリに保存され、このディレクトリには数十万ものキー値の大規模なセットを保持できます。

個々のキー値のある場所から別の場所に自由に移動させたり、データやロード・バランシングのために自由に一括移動を行ってスケールアップまたはスケールダウンを図ったりすることができます。場所情報には、シャード・データベース情報とパーティション情報を含めることができます。

詳細については、『Oracle Globally Distributed Database Guide』ドキュメントの「[ディレクトリベースのシャーディング](#)」を参照してください。

参考資料：

オラクルのJSON機能：

- [23aiにおけるJSONドキュメントのシャーディングの概要](#)
- [23aiドキュメント・リファレンス：JSON開発者ガイド](#)
- [新しいJSONデータタイプに関するオラクルのブログ記事：](#)
- [JSONドキュメントをシャード環境に保存する方法に関するオラクルのブログ記事：](#)

Oracle Text：

- [技術概要：Oracle DatabaseのOracle Textの新しいユーザー・ガイド](#)
- [23ai Oracle Textアプリケーション開発者ガイド](#)

Globally Distributed Database：

- [23ai Globally Distributed Databaseのドキュメント](#)
- [シャーディング用のTLSを使用したTCPSの構成](#)
- [Oracle Globally Distributed Databaseのデプロイ](#)
- [コマンド・リファレンス](#)

Global Data Services：

- [Oracle Database Global Data Services概要および管理ガイド](#)
-

Connect with us

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

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

Copyright © 2024, Oracle and/or its affiliates.本文書は情報提供のみを目的として提供されており、ここに記載されている内容は予告なく変更されることがあります。本文書は、その内容に誤りがないことを保証するものではなく、また、口頭による明示的保証や法律による黙示的保証を含め、商品性ないし特定目的適合性に関する黙示的保証および条件などのいかなる保証および条件も提供するものではありません。オラクルは本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。本文書はオラクルの書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。

Oracle、Java、MySQLおよびNetSuiteは、Oracleおよびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。