Oracle® Database Testing ガイド 19c

F16126-04(原本部品番号:E96279-04)

2022年12月

タイトルおよび著作権情報

Oracle Database Testingガイド, 19c

F16126-04

原本著者: Sunil Surabhi

原本協力著者: Immanuel Chan

原本協力者: Ashish Agrawal、Waleed Ahmed、Helen Altmar、Lance Ashdown、Pete Belknap、Supiti Buranawatanachoke、Romain Colle、Karl Dias、Kurt Engeleiter、Leonidas Galanis、Veeranjaneyulu Goli、Prabhaker Gongloor、Prakash Gupta、Shantanu Joshi、Prathiba Kalirengan、Karen McKeen、Mughees Minhas、Konstantinos Morfonios、Valarie Moore、Ravi Pattabhi、Bert Rich、Yujun Wang、Keith Wong、Qinyi Wu、Khaled Yagoub、Hailing Yu、Yury Berezin

目次

表一覧

はじめに

内容は次のとおりです。

対象読者

このマニュアルでは、Oracle Real Application Testingを使用してデータベース変更の整合性を保持し、テスト・データを管理する方法について説明します。このマニュアルは、Oracle Databaseを現実的な環境でテストするデータベース管理者、アプリケーション設計者、およびプログラマを対象としています。

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

このマニュアルで説明する一部の項目の詳細は、Oracle Databaseリリース12.1ドキュメント・セットの次のマニュアルを参照してください。

- Oracle Database 2日でデータベース管理者
- Oracle Database 2日でパフォーマンス・チューニング・ガイド
- Oracle Database管理者ガイド
- Oracle Database概要
- Oracle Databaseパフォーマンス・チューニング・ガイド
- Oracle Database SQLチューニング・ガイド

表記規則

このマニュアル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。

本リリースでのOracle Database Testingガイドの変更点

この序文では、『Oracle Database Testingガイド』の変更点をあげます。

Oracle Databaseリリース19c、バージョン19.1での変更点

次に、Oracle Database 19c、バージョン19.1の『Oracle Database Testingガイド』の変更点をあげます。

新機能

この項では、Oracle Databaseリリース19c、バージョン19.1のSQLパフォーマンス・アナライザとDBリプレイの新機能について説明します。

- 現在のプラガブル・データベース(PDB)に対して、ワークロード取得の有効化とワークロード・リプレイの開始が可能です。

  関連項目:

  PDBでのワークロードの取得とリプレイ

- ワークロード・リプレイ・レポートを使用すると、遅いワークロード・リプレイの診断が可能になります。

  関連項目:

  ワークロードのリプレイ・レポートの確認

- DBA_RAT_CAPTURE_SCHEMA_INFOビューを問い合せると、ワークロード取得でSQL文が記録されたときに有効になっていたlogin_schemaとcurrent_schemaを取得できます。

  関連項目:

  ワークロードのリプレイの開始

Oracle Databaseリリース18c、バージョン18.1での変更点

次に、Oracle Database 18c、バージョン18.1の『Oracle Database Testingガイド』の変更点をあげます。

新機能

この項では、Oracle Databaseリリース18c、バージョン18.1のSQLパフォーマンス・アナライザとDBリプレイの新機能について説明します。

- 新しいENCRYPTIONパラメータを使用して、ワークロード取得の暗号化を有効化できます。

  関連項目:

  - ワークロードの取得の開始

    - リプレイ・クライアントの開始

- 既存のワークロード取得を暗号化および復号化できます。

  関連項目:

  APIの使用による既存のワークロード取得の暗号化および復号化

- 新しいTEST_EXECUTE_DOPパラメータを使用して、SQLパフォーマンス・アナライザでの同時SQL実行を有効化できます。

  関連項目:

  分析タスクの並列度の構成

- 新しいCOMPARE_RESULTSETパラメータを使用して、SQL結果セットを検証できます。

  関連項目:

  SQLパフォーマンス・アナライザの使用によるSQL結果セットの検証

- 必要なDatabase Vault権限がある場合は、Database Vault環境でデータベース・リプレイ操作を実行できます。

  関連項目:

  Database Vault環境でのデータベース・リプレイの使用の詳細は、『Oracle Database Vault管理者ガイド』を参照してください。

Oracle Database 12cリリース2 (12.2.0.1)での変更点

次に、Oracle Database 12cリリース2 (12.2.0.1)の『Oracle Database Testingガイド』の変更点をあげます。

Oracle Database 12cリリース2 (12.2.0.1)の新機能

この項では、Oracle Database 12cリリース2 (12.2.0.1)のSQLパフォーマンス・アナライザとDBリプレイの新機能について説明します。

- SQLパフォーマンス・アナライザの分析タスクでトリガーの実行を有効または無効にできます。

  「分析タスクでのトリガーの実行の構成」を参照してください。

- SQLパフォーマンス・アナライザの分析タスクのコールで返される日付を構成できます。

  「分析タスクでコールにより返される日付の構成」を参照してください。

- SQLパフォーマンス・アナライザの分析タスクでフェッチする行数を構成できます。

  「分析タスクでフェッチする行数の構成」を参照してください。

- DBリプレイを使用したワークロードの取得とワークロードのリプレイ中のユーザーによるPL/SQLコールの処理方法を指定できます。

詳細は、次の各項を参照してください。

- ワークロードの取得の開始
- APIを使用したデータベース・ワークロードの事前処理
- リプレイ・データの初期化

「APIを使用したデータベース統合リプレイの初期化」で説明されているように、plsql_modeパラメータはINITIALIZE_CONSOLIDATED_REPLAYプロシージャで使用することもできます。

Oracle Database 12cリリース2 (12.2.0.1)でのその他の変更点

ここでは、Oracle Database 12c リリース2 (12.2.0.1)用の『Oracle Database Testingガイド』における、その他の変更点について説明します。

Oracle Database 12c リリース2 (12.2.0.1)用の『Oracle Database Testingガイド』では、次に示す変更も実施されています。

- このマニュアルの第III部で説明されていた、Oracle Database 12cリリース1のテストデータ管理機能は削除されました。これに該当する機能は、別のマニュアル(『Oracle Data Masking and Subsettingガイド』)に記載されることになりました。

Oracle Database 12cリリース1 (12.1)での変更点

次に、Oracle Database 12cリリース1(12.1)の『Oracle Database Testingガイド』の変更点をあげます。

新機能

このリリースの新機能は次のとおりです。

- SQLパフォーマンス・アナライザ・クイック・チェック(SPAクイック・チェック)

  一部のOracle Enterprise Manager Cloud Controlデータベース管理ページで、SPAクイック・チェックは、変更を加える前にデータベース・ワークロードへのシステム変更の影響を検証できます。

  「SPAクイック・チェックの使用」を参照してください。

- SQLパフォーマンス・アナライザによるマルチテナント・アーキテクチャのサポート

  非CDBからマルチテナント・コンテナ・データベース(CDB)に転送されたSQLチューニング・セットを、CDB識別子を再マッピングすることによりSQLパフォーマンス・アナライザへの入力ソースとして使用することができます。

  「APIを使用した分析タスクでのマルチテナント・コンテナ・データベース識別子の再マッピング」を参照してください。

- ワークロード・インテリジェンス

  ワークロード・インテリジェンスとは、取得したワークロードに格納されているデータを分析するJavaプログラムのスイートです。

  「ワークロード・インテリジェンスの使用」を参照してください。

- データベース統合リプレイ

  データベース統合リプレイでは、1つ以上のシステムから取得した複数のワークロードを統合して、Oracle Exadata

Machineなど1つのテスト・システムで同時にリプレイできます。

「データベース統合リプレイの使用」を参照してください。

- ワークロード・スケールアップ

  データベース・リプレイでは、様々な使用事例やシナリオで、スケールアップ・テストやストレス・テストを実行できます。

  「ワークロード・スケールアップの使用」を参照してください。

その他の変更

このリリースでの追加変更は次のとおりです。

- 新規のマニュアル

  『Oracle Database Testingガイド』に、『Oracle Database Real Application Testingユーザーズ・ガイド』が含まれました。

- 新しい章

  『Oracle Database Testingガイド』に、Oracle Databaseのテスト・データの管理機能を説明する第III部「テスト・データの管理」が追加されました。

1 Oracle Database Testingの概要

Oracle DatabaseのOracle Real Application Testingオプションでは、データベースに対する変更の整合性を保証したり、テスト・データを管理できます。

Oracle Real Application Testingオプションを使用すると、Oracle Databaseを現実的な環境でテストできます。Oracle Real Application Testingでは、本番環境にデプロイメントを行う前に、本番環境のワークロードを取得し、こらのワークロードに対するシステムの変更の影響を評価することで、システム変更が原因で不安定になるリスクを最小限にします。Oracle Real Application Testingの主要コンポーネントは、SQLパフォーマンス・アナライザおよびデータベース・リプレイです。テストするシステム変更の性質とその影響、およびテストするシステムの種類に応じて、テストの実行にいずれかまたは両方のコンポーネントを使用できます。

この章の構成は、次のとおりです。

- SQLパフォーマンス・アナライザ
- データベース・リプレイ

> ノート:
>
> SQL パフォーマンス・アナライザおよびデータベース・リプレイの使用には、Oracle Real Application Testing のライセンス・オプションが必要です。詳細は、『Oracle Database ライセンス情報ユーザー・マニュアル』を参照してください。

SQLパフォーマンス・アナライザ

システム変更(データベースのアップグレードや索引の追加など)を行うと、SQL文の実行計画が変更され、SQLパフォーマンスに大きな影響を与える場合があります。システム変更によってSQL文が低下し、パフォーマンスの低下につながる場合もあります。システム変更によってSQLパフォーマンスが改善される場合もあります。システム変更がSQLパフォーマンスに与える可能性がある影響を正確に予測できると、SQL文の低下に備えてシステムを事前にチューニングしたり、SQL文のパフォーマンスが改善された場合のパフォーマンスの向上を検証し、評価することができます。

SQLパフォーマンス・アナライザは、SQL文ごとのパフォーマンスの相違を識別することで、すべてのSQLワークロードに対する変更の全体的影響を評価するプロセスを自動化します。レポートには、変更によるワークロード・パフォーマンスへの最終的な影響が示されます。SQLパフォーマンス・アナライザでは、パフォーマンスが低下しているSQL文に適した詳細な実行計画や推奨の調整方法を提供します。これによって、エンド・ユーザーに影響が及ぶ前に、望ましくない結果を修正することができます。また、時間とコストを大幅に節約しながら、本番環境のシステム変更が最終的なパフォーマンス向上につながることを検証できます。

SQLパフォーマンス・アナライザを使用すると、次に示す変更を含むあらゆるタイプのシステム変更のSQLパフォーマンスに対する影響を分析できます。

- データベース・アップグレード
- プラガブル・データベース(PDB)および手動で統合したスキーマのデータベース統合テスト

- オペレーティング・システムまたはハードウェアの構成変更
- スキーマの変更
- データベース初期化パラメータの変更
- オプティマイザ統計のリフレッシュ
- SQLチューニング・アクションの検証

関連項目:

- SQLパフォーマンス・アナライザの使用については、「SQLパフォーマンス・アナライザの概要」を参照してください。

データベース・リプレイ

ハードウェアやソフトウェアのアップグレードなどのシステム変更を行う前に、通常は変更内容を検証するためにテスト環境で広範囲なテストが実行されます。ただし、テストを実行しても、テストには実際のワークロードを使用していないため、新規システムの本番運用が開始されると、しばしば予期しない動作が発生します。テスト中に実際のワークロードをシミュレートできないことは、システム変更を検証する際の最大の問題点の1つです。

データベース・リプレイでは、テスト・システムで本番環境のワークロードを実質的に再現し、システム変更の現実的なテストを可能にします。データベース・リプレイを使用すると、本番システムのワークロードを取得して、それを元のワークロードとまったく同じタイミング、同時実行性およびトランザクション特性に従ってテスト・システムでリプレイできます。これにより、変更の影響(望ましくない結果、新しい競合ポイント、計画の品質低下など)を詳細に評価できます。広範な分析およびレポートを利用して、新しく発生したエラーやパフォーマンスの相違など、起こりうる問題の特定に役立てることができます。

データベース・リプレイでは、外部データベース・クライアントのワークロードをデータベース・レベルで取得するもので、パフォーマンス・オーバーヘッドはごくわずかです。本番ワークロードを取得することで、シミュレーション・ワークロードやスクリプトを開発する必要がなくなり、コストと時間を大幅に節約できます。データベース・リプレイを使用すると、以前であれば負荷シミュレーション・ツールを使用して何か月もかかっていた複雑なアプリケーションの現実的なテストが、数日で完了します。これにより、より高度な信頼性をより少ないリスクで確保しながら、変更を迅速にテストして新しいテクノロジを導入できます。

データベース・リプレイを使用すると、次のような重要なシステム変更をテストできます。

- データベースおよびオペレーティング・システムのアップグレード
- PDBおよび手動で統合したスキーマのデータベース統合テスト
- ワークロードのスケールアップの様々なシナリオのオーサリングおよび実験
- 構成の変更(シングル・インスタンスからOracle Real Application Clusters(Oracle RAC)環境へのデータベースの変換など)
- ストレージ、ネットワークおよびインターコネクトの変更
- オペレーティング・システムおよびハードウェアの移行

関連項目:

- データベース・リプレイの使用の詳細は、「データベース・リプレイの概要」を参照してください。

第I部 SQLパフォーマンス・アナライザ

SQLパフォーマンス・アナライザでは、SQL文のSQLレスポンス時間に対するシステムの変更の影響を評価できます。

第I部は、SQLパフォーマンス・アナライザについて説明し、次の章で構成されています。

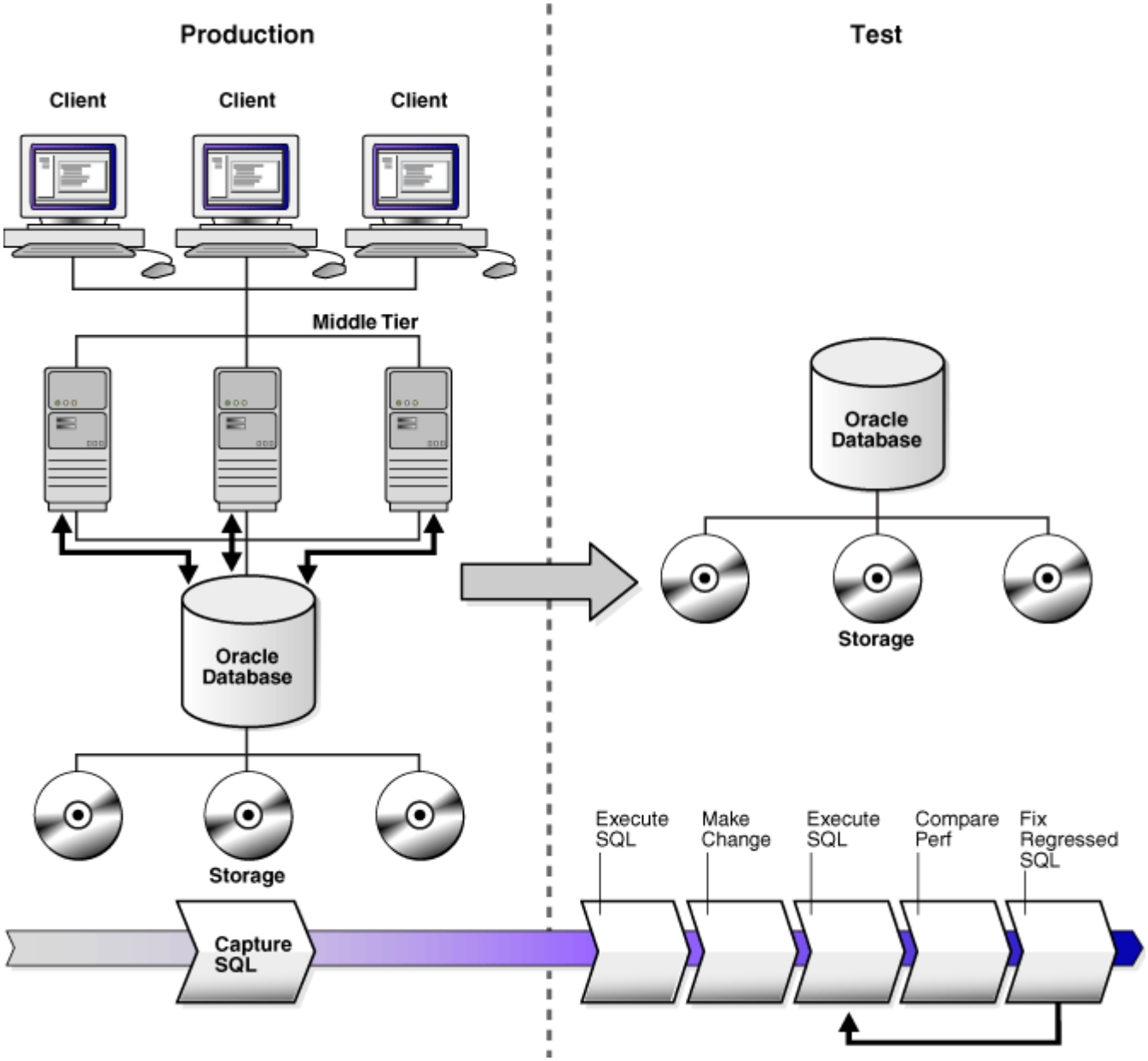
- SQLパフォーマンス・アナライザの概要
- 分析タスクの作成
- 変更前のSQL試行の作成
- 変更後のSQL試行の作成
- SQL試行の比較
- SPAクイック・チェックの使用
- データベースのアップグレードのテスト

2 SQLパフォーマンス・アナライザの概要

SQLパフォーマンス・アナライザは、本番システムまたは本番システムによく似たテスト・システムで実行できます。SQLパフォーマンス・アナライザではテストするSQL文を実行する必要があるため、本番システムでシステム変更をテストすると、システムのスループットに影響します。パフォーマンスへの影響をテストするためにシステムで行うグローバル変更によって、システムのその他のユーザーが影響を受ける可能性もあります。システム変更が多数のセッションまたはSQL文に影響しない場合は、本番システムでSQLパフォーマンス・アナライザを実行できます。ただし、データベースのアップグレードなどのシステム全体の変更では本番システムを使用しないことをお薦めします。かわりに、本番システムに影響を与えずにシステム変更の影響をテストできるように、別のテスト・システムでSQLパフォーマンス・アナライザを実行することを検討してください。テスト・システムを使用すると、本番システムで実行されているその他のワークロードがSQLパフォーマンス・アナライザによって実行される分析に影響することもなくなります。推奨される方法はテスト・システムでSQLパフォーマンス・アナライザを実行する方法であり、その方法をここで説明します。本番システムでSQLパフォーマンス・アナライザを実行する場合は、適宜テスト・システムを本番システムに置き換えてください。

SQLパフォーマンス・アナライザを使用してシステム変更によるSQLパフォーマンスへの影響を分析する場合は、図2-1に示すステップを実行します。

図2-1 SQLパフォーマンス・アナライザのワークフロー



1. 分析対象のSQLワークロードを取得して、SQLチューニング・セットに格納します。詳細は、「SQLワークロードの取得」を参照してください。
2. 本番システムとは切り離されたテスト・システムを使用する場合は、次のステップを実行します。
    a. 可能なかぎり本番環境と一致するようにテスト・システムを設定します。
    b. SQLチューニング・セットをテスト・システムに転送します。
3. テスト・システムで、SQLパフォーマンス・アナライザのタスクを作成します。詳細は、「SQLパフォーマンス・アナライザのタスクの作成」を参照してください。
4. SQLチューニング・セットに格納されているSQL文のテスト実行または実行計画の生成を行って、変更前のSQL試行を作成します。詳細は、「変更前のSQLパフォーマンスの測定」を参照してください
5. システム変更を実行します。詳細は、「システム変更の実行」を参照してください
6. 変更後のテスト・システムでSQLチューニング・セット内のSQL文を再実行して、変更後のSQL試行を作成します。詳細は、「変更後のSQLパフォーマンスの測定」を参照してください
7. 変更前バージョンと変更後バージョンのパフォーマンス・データを比較および分析し、レポートを作成して、システム変更後にパフォーマンスが改善されたSQL文、パフォーマンスの変更がなかったSQL文またはパフォーマンスが低下したSQL

文を特定します。詳細は、「パフォーマンス測定値の比較」を参照してください

8. 特定されたパフォーマンスが低下したSQL文をチューニングします。詳細は、「パフォーマンスが低下したSQL文の修正」を参照してください。
9. パフォーマンスの目標を達成するまで、ステップ6から8を繰り返し、チューニングしたSQL文のパフォーマンスが許容範囲内であることを確認します。

   それぞれの比較において、以前のSQL試行を変更前のSQL試行として使用し、現在のSQL試行を変更後のSQL試行として使用できます。たとえば、最初のSQL試行を現在のSQL試行と比較して、すべての変更を評価したり、最新のSQL試行を現在のSQL試行と比較して、最新の変更のみを評価する場合があります。

ノート:

Oracle Enterprise Manager では、このプロセスを簡単にするため、ステップ 3 から 9 が自動化されたワークフローが提供されています。

ノート:

SQL パフォーマンス・アナライザでプラガブル・データベース(PDB)を使用する場合、データの可視性および必要な権限が異なる場合があります。

関連項目:

- 「テスト・システムの設定」
- マルチテナント・コンテナ・データベース(CDB)で、SQLパフォーマンス・アナライザなどの管理性機能がどのように動作するかの詳細は、『Oracle Database管理者ガイド』を参照してください

SQLワークロードの取得

SQLパフォーマンス・アナライザを実行する前に、分析対象のSQLワークロードを表すSQL文のセットを本番システムで取得します。

取得したSQL文には、次の情報が含まれています。

- SQLテキスト
- 実行環境
  - SQL文を実行し、正確な実行統計を生成するために必要なバインド値であるSQLバインド
  - SQL文をコンパイルできる解析スキーマ
  - SQL文が実行される、初期化パラメータが含まれているコンパイル環境
- SQL文の実行回数

SQLワークロードの取得は、本番システムのパフォーマンスにごくわずかな影響を与えますが、スループットには影響を与えません。

より多くのSQL文が含まれているSQLワークロードでは、アプリケーションまたはデータベースの状態がよりよく表示されます。これによって、システム変更がSQLワークロードに与える可能性がある影響をSQLパフォーマンス・アナライザでより正確に予測できるようになります。そのため、可能なかぎり多くのSQL文を取得する必要があります。アプリケーションでコールされているすべてのSQL文またはデータベースで実行されているすべてのSQL文のいずれかを取得することをお薦めします。

取得したSQL文をSQLチューニング・セットに格納し、SQLパフォーマンス・アナライザの入力ソースとして使用できます。SQLチューニング・セットは、1つ以上のSQL文がその実行統計および実行コンテキストとともに含まれているデータベース・オブジェクトです。SQL文は、カーソル・キャッシュ、自動ワークロード・リポジトリ(AWR)、SQLトレース・ファイル、既存のSQLチューニング・セットなどの様々なソースからSQLチューニング・セットにロードできます。SQLチューニング・セットを使用してSQLワークロードを取得すると、次のことができます。

- 単一の永続データベース・オブジェクトへのSQLテキストおよび必要な補助情報の格納
- SQLチューニング・セット内の取得済SQL文の移入、更新、削除および選択
- 自動ワークロード・リポジトリ(AWR)やカーソル・キャッシュなどの様々なデータ・ソースからのコンテンツのロードおよびマージ
- SQLワークロードを取得したシステムからのSQLチューニング・セットのエクスポート、および別のシステムへのSQLチューニング・セットのインポート
- SQLチューニング・アドバイザやSQLアクセス・ アドバイザなどの他のアドバイザへの入力ソースとしてのSQLワークロードの再使用

関連項目:

- Oracle Enterprise Managerを使用してSQLチューニング・セットを作成する方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
- APIを使用してSQLチューニング・セットを作成する方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください

テスト・システムの設定

本番システムでSQLチューニング・セットにSQLワークロードを取得したら、ワークロードを取得したデータベースと同じデータベースまたは異なるデータベースでSQLパフォーマンス・アナライザの分析を実行できます。分析ではリソースが多く消費されるため、本番データベースでワークロードを取得して、分析を実行できる別のテスト・データベースに転送することをお薦めします。これを行うには、SQLチューニング・セットを本番システムからエクスポートし、システム変更をテストする別のシステムにインポートします。

テスト・データベースを作成する方法はいくつもあります。たとえば、Recovery Manager(RMAN)のDUPLICATEコマンド、Oracle Data Pumpまたはトランスポータブル表領域を使用できます。RMANではテスト・データベースを既存のバックアップまたはアクティブな本番データ・ファイルから作成できるため、RMANを使用することをお薦めします。本番データベースとテスト・データベースは、同じホストまたは異なるホストのいずれにも存在できます。

可能なかぎり本番システムのデータベース環境と一致するようにテスト・データベース環境を構成する必要があります。これによって、システム変更がSQLパフォーマンスに与える影響をSQLパフォーマンス・アナライザでより正確に予測できるようになります。

テスト・システムを適切に構成したら、SQLチューニング・セットを本番システムからステージング表にエクスポートし、次にステージ

ング表からテスト・システムにインポートします。

関連項目:

- RMANを使用したデータベースの複製については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- Oracle Enterprise Managerを使用してSQLチューニング・セットを転送する方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
- APIを使用してSQLチューニング・セットを転送する方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください

SQLパフォーマンス・アナライザ・タスクの作成

SQLワークロードを取得してテスト・システムに転送し、初期化データベース環境を適切に構成したら、SQLパフォーマンス・アナライザを実行して、システム変更がSQLパフォーマンスに与える影響を分析できます。

SQLパフォーマンス・アナライザを実行するには、まずSQLパフォーマンス・アナライザのタスクを作成する必要があります。タスクは、SQLパフォーマンス・アナライザの完全な分析に関するすべてのデータがカプセル化されているコンテナです。SQLパフォーマンス・アナライザの分析は、2つ以上のSQL試行と1つの比較で構成されています。SQL試行で、特定の環境条件下でのSQLチューニング・セットの実行パフォーマンスがカプセル化されます。SQLパフォーマンス・アナライザのタスクを作成する場合は、入力ソースとしてSQLチューニング・セットを選択する必要があります。テスト実行または実行計画の方法でSQL試行を構築する場合、SQL文のソースとしてSQLチューニング・セットが使用されます。SQLパフォーマンス・アナライザは、2つの試行間での環境の相違による影響を表示します。

関連項目:

- SQLパフォーマンス・アナライザ・タスクの作成方法は、「分析タスクの作成」 を参照

変更前のSQLパフォーマンスの測定

システム変更を行う前に、変更前のSQL試行を作成します。SQLパフォーマンス・アナライザでのSQL試行に必要なパフォーマンス・データは、次に示す方法で生成できます。

- テスト実行

  この方法では、SQLパフォーマンス・アナライザを介してSQL文をテスト実行します。SPAパフォーマンス・アナライザを実行しているデータベースまたはリモート・データベースで実行できます。

- 実行計画

  この方法では、SQLパフォーマンス・アナライザを介してSQL文に対してのみ実行計画を生成します。SPAパフォーマンス・アナライザを実行しているデータベースまたはリモート・データベースで実行できます。EXPLAIN PLAN文と異なり、実行計画の方法を使用するSQL試行ではバインド値が考慮され、実際の実行計画が生成されます。

- SQLチューニング・セットの変換

この方法では、SQLチューニング・セットに格納されている実行統計と計画を変換します。この方法はAPIでのみサポートされています。

テスト実行の方法では、ワークロードに含まれている各SQL文が完了するまで実行されます。実行中、SQLパフォーマンス・アナライザによってワークロードのSQL文ごとに実行計画が生成され、実行統計が計算されます。SQLチューニング・セット内の各SQL文は、SQL文の初期の実行順序または同時実行性を維持せずに、その他のSQL文とは別に実行されます。これは、実行タイムアウトが発生するまで可能なかぎり多く(最大10回)、SQL文ごとに2回以上実行されます。最初の実行は、バッファ・キャッシュの準備のために使用されます。以降のすべての実行は、平均に基づいてSQL文の実行時の実行統計を計算するために使用されます。SQL文が実行される実際の回数は、SQL文の実行時間の長さによって異なります。実行時間が長いSQL文は2回だけ実行され、この実行から得られた実行統計が使用されます。その他の(比較的時間の短い)SQL文は複数回実行され、これらの実行から実行統計の平均が計算されます(最初の実行で得られた統計は計算に使用されません)。複数の実行の統計の平均を求めることで、SQLパフォーマンス・アナライザは各SQL文の実行統計をより正確に計算できます。データベースが受ける可能性がある影響を回避するために、DDLはサポートされていません。デフォルトでは、DMLの問合せ部分のみが実行されます。APIを使用する場合は、EXECUTE_FULLDMLタスク・パラメータを使用することでDML全体を実行できます。パラレルDMLはサポートされておらず、パラレル・ヒントが削除されないかぎり、その問合せ部分は実行されません。

サイズによっては、SQLワークロードの実行に時間およびリソースが大量に消費される可能性があります。実行計画の方法では、実行統計を収集せずに、実行計画のみの生成を選択することができます。この方法によって、試行を実行する時間が短縮され、システム・リソースへの影響が減少しますが、分析時に使用できるのは実行計画のみのため、包括的なパフォーマンスの分析は実行できません。ただし、EXPLAIN PLANコマンドで計画を生成した場合とは異なり、SQLパフォーマンス・アナライザから、実行計画の生成時にオプティマイザにバインド値が提供されるため、SQL文の実行時に計画の結果に関してより信頼性の高い予測が得られます。

どちらの場合も、データベース・リンクを使用して、別のデータベースでSQLワークロードをリモートで実行することができます。SQLパフォーマンス・アナライザによって、データベース・リンクを介してリモート・データベースへの接続が確立され、そのデータベースでSQL文が実行され、SQL文ごとの実行計画およびランタイム統計が収集されて、ローカル・データベースのSQL試行に以降の分析で使用可能な結果が格納されます。この方法は、次の操作を行う場合に有効です。

- データベースのアップグレードのテスト
- 別のバージョンのOracle Databaseが実行されているシステムでのSQLワークロードの実行
- 別のテスト・システムへのSQLパフォーマンス・アナライザの分析結果の格納
- ハードウェア構成が異なる複数のシステムでのテストの実行
- 本番システムで古いバージョンのOracle Databaseを使用している場合のSQLパフォーマンス・アナライザの最新機能の使用

SQLワークロードを実行すると、生成された実行計画およびランタイム統計がSQL試行に格納されます。

SQLチューニング・セットに格納されている実行統計および計画を使用して、SQL試行を作成することもできます。この方法は、APIでのみサポートされていますが、ワークロードを実行する別の方法(データベース・リプレイまたは別のアプリケーション・テスト・ツール)があり、テスト・システムでワークロードを実行するためにSQLパフォーマンス・アナライザが必要ない場合に有効なことがあります。このような場合でも、テストの実行中にSQLチューニング・セットを取得すると、SQLパフォーマンス・アナライザを使用してこれらのSQLチューニング・セットからSQL試行を作成し、より包括的な分析レポートを表示することができます。標準のSQLパフォーマンス・アナライザ・レポート(各試行内の実行計画は1つのみで、1セットのバインドでSQL文を実行して生成される実行

統計は1セット)とは異なり、SQLチューニング・セットから作成したSQL試行を比較するレポートを生成して、複数の実行にわたって潜在的に多数の異なるバインド・セットが存在する2つの試行からすべての実行計画を表示することができます。

関連項目:

- 変更前のパフォーマンスを測定する方法は、「変更前のSQL試行の作成」を参照
- リモート・システムでSQLワークロードを実行して、データベースのアップグレードをテストする方法の詳細は、「データベースのアップグレードのテスト」を参照してください

システム変更の実行

測定対象のSQLパフォーマンスにかかわる変更を行います。SQLパフォーマンス・アナライザでは、様々なタイプのシステム変更による影響を分析できます。たとえば、データベースのアップグレード、新しい索引の作成、初期化パラメータの変更、オプティマイザ統計のリフレッシュなどをテストできます。本番システムでSQLパフォーマンス・アナライザを実行する場合は、残りのシステムへの影響を回避するためにプライベート・セッションを使用して変更を行うことを検討してください。

変更後のSQLパフォーマンスの測定

システム変更の実行後、変更後のSQL試行を作成します。変更後のSQL試行は、変更前のSQL試行と同じ方法を使用して作成することを強くお薦めします。作成すると、変更後のSQL試行に新しいパフォーマンス・データセットが生成され、変更前のバージョンと比較可能になります。この結果は、新しいSQL試行または変更後のSQL試行に保存されます。

関連項目:

- 変更後のパフォーマンスを測定する方法は、「変更後のSQL試行の作成」を参照

パフォーマンス測定値の比較

SQLパフォーマンス・アナライザでは、変更前と変更後のSQL文のパフォーマンスが比較され、SQL文の実行計画またはパフォーマンスでの変更を特定するレポートが生成されます。

SQLパフォーマンス・アナライザは、SQLワークロード全体の実行時間とワークロード内の個々のSQL文のレスポンス時間について、システム変更による影響を測定します。デフォルトでは、SQLパフォーマンス・アナライザは、比較のメトリックに経過時間を使用します。また、次のSQL実行統計の中から比較のメトリックを選択できます。

- CPU時間
- ユーザーI/O時間
- バッファ読取り
- 物理I/O
- オプティマイザ・コスト
- I/Oインターコネクト・バイト
- 式形式でのこれらメトリックの組合せ

SQL試行で実行計画のみを生成する場合、SQLパフォーマンス・アナライザではSQL実行計画に格納されているオプティマイザ・コストが使用されます。

比較が完了すると、変更前と変更後のSQLパフォーマンスを比較するSQLパフォーマンス・アナライザ・レポートに結果のデータが生成されます。SQLパフォーマンス・アナライザ・レポートは、HTML、テキストまたはアクティブ・レポートとして表示できます。アクティブ・レポートには、インタラクティブなユーザー・インタフェースを使用するきめ細かなレポート機能が用意されており、データベースやOracle Enterprise Managerに接続されていないときでも詳細な分析を実行できます。

関連項目:

- パフォーマンス測定値の比較とレポートの詳細は、「SQL試行の比較」を参照してください

低下したSQL文の修正

SQLパフォーマンス・アナライザが実行したパフォーマンス分析でSQL文の低下が見つかった場合は、変更を行って問題を解決します。たとえば、SQLチューニング・アドバイザを実行するか、またはSQL計画ベースラインを使用して、パフォーマンスが低下したSQLを修正できます。その後、SQL文の実行プロセスを繰り返し、そのパフォーマンスを最初の実行と比較できます。満足できる分析結果になるまで、これらのステップを繰り返します。

関連項目:

- 低下したSQL文の修正方法は、「SQL試行の比較」を参照

3 分析タスクの作成

分析するSQLワークロードをSQLチューニング・セット(STS)に取得したら、SQLパフォーマンス・アナライザを実行して、システム変更がSQLパフォーマンスに与える影響を分析できます。SQLパフォーマンス・アナライザを実行するには、まずSQLパフォーマンス・アナライザのタスクを作成する必要があります。タスクは、SQLパフォーマンス・アナライザの完全な分析に関するすべてのデータがカプセル化されているコンテナです。SQLパフォーマンス・アナライザの分析は、2つ以上のSQL試行と1つの比較で構成されています。SQL試行は特定の環境条件下でSQLチューニング・セットの実行パフォーマンスを取得するもので、次のいずれかの方法でSQLパフォーマンス・アナライザを使用して自動的に生成できます。

- SQL文のテスト実行
- SQL文の実行計画の生成
- SQLチューニング・セットで取得した実行統計および計画の参照

SQLパフォーマンス・アナライザのタスクを作成する場合は、入力ソースとしてSQLチューニング・セットを選択する必要があります。このSQLチューニング・セットがSQL試行のテスト実行または実行計画生成のソースとして使用されます。このため、試行間のパフォーマンスの相違は、環境の相違が原因となります。

この章では、SQLパフォーマンス・アナライザのタスクの作成方法について説明します。内容は次のとおりです。

- Enterprise Managerを使用した分析タスクの作成
- APIを使用した分析タスクの作成
- APIを使用した分析タスクの構成

> ノート:
>
> SQL パフォーマンス・アナライザを実行するための主要なインタフェースは、Oracle Enterprise Manager です。なんらかの理由で Oracle Enterprise Manager を使用できない場合は、DBMS_SQLPA PL/SQL パッケージを使用して SQL パフォーマンス・アナライザを実行できます。

関連項目:

「SQLパフォーマンス・アナライザ・タスクの作成」

Enterprise Managerを使用した分析タスクの作成

Oracle Enterprise Managerでは、いくつかのワークフローでSQLパフォーマンス・アナライザ・タスクを作成できます。

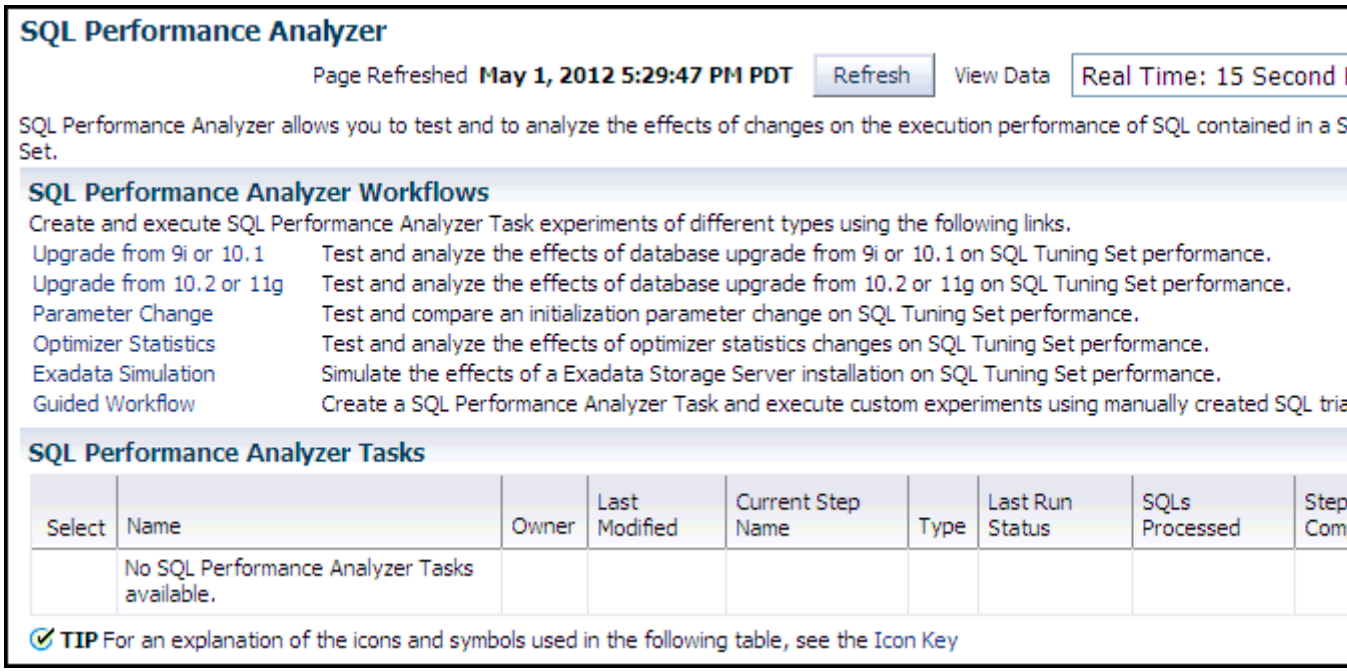
SQLパフォーマンス・アナライザを実行する前に、パフォーマンスの分析に使用するSQLワークロードを本番システムのSQLチューニング・セットに取得して、パフォーマンスの分析を実行するテスト・システムに転送します。詳細は、「SQLワークロードの取得」を参照してください。

Enterprise Managerを使用して分析タスクを作成するには:

1. 「パフォーマンス」メニューから、「SQL」、「SQLパフォーマンス・アナライザ」の順に選択します。

「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。



2. 「SQLパフォーマンス・アナライザ・ワークフロー」で、目的のタイプの分析タスクを作成するワークフローを選択します。
   - 9iまたは10.1からのアップグレード

     9iまたは10.1のワークフローからのアップグレードは、Oracle9i DatabaseまたはOracle Database 10gリリース1から、Oracle Database 10gリリース2以上のリリースへのデータベース・アップグレードをテストする場合に使用します。詳細は、「Oracle9i DatabaseおよびOracle Database 10gリリース1からのアップグレード」を参照してください。

   - 10.2または11gからのアップグレード

     10.2または11gのワークフローからのアップグレードは、Oracle Database 10gリリース2またはOracle Database 11gから、それ以上のリリースへのデータベース・アップグレードをテストする場合に使用します。詳細は、「Oracle Database 10gリリース2以上のリリースからのアップグレード」を参照してください。

   - パラメータの変更

     パラメータ変更ワークフローは、データベース初期化パラメータの変更によるSQLパフォーマンスへの影響を確認する場合に使用します。詳細は、「パラメータ変更ワークフローの使用」を参照してください。

   - オプティマイザ統計

     オプティマイザ統計ワークフローは、オプティマイザ統計への変更がSQLのパフォーマンスに与える影響を分析する場合に使用します。詳細は、「オプティマイザ統計ワークフローの使用」を参照してください。

   - Exadataシミュレーション

     Exadataシミュレーション・ワークフローは、Oracle Exadataを使用した場合にSQLのパフォーマンスに与える影響をシミュレートする場合に使用します。詳細は、「Exadataシミュレーション・ワークフローの使用」を参照してください。

   - ガイド付きワークフロー

     ガイド付きワークフローは、その他のすべてのタイプのシステム変更でのSQLパフォーマンスを比較する場合に使

用します。詳細は、「ガイド付きワークフローの使用」を参照してください。

パラメータ変更ワークフローの使用

パラメータ変更ワークフローでは、1つの環境初期化パラメータの値を変更した場合のSQLワークロードへのパフォーマンスの影響をテストできます。たとえば、OPTIMIZER_FEATURES_ENABLE初期化パラメータ・セットを10.2.0.4と12.1.0.1に設定することによって、SQLパフォーマンスを比較できます。

SQLチューニング・セットおよび比較メトリックを選択すると、SQLパフォーマンス・アナライザによってタスクが作成され、初期化パラメータが元の値に設定された試行が実行されます。その後、ALTER SESSION文を発行することによって、SQLパフォーマンス・アナライザで、パラメータが変更された値に設定された2番目の試行が実行されます。したがって、変更の影響はテスト・セッションにローカルに限定されます。パフォーマンスの低下または変更は、システムによって生成されたSQLパフォーマンス・アナライザ・レポートに報告されます。

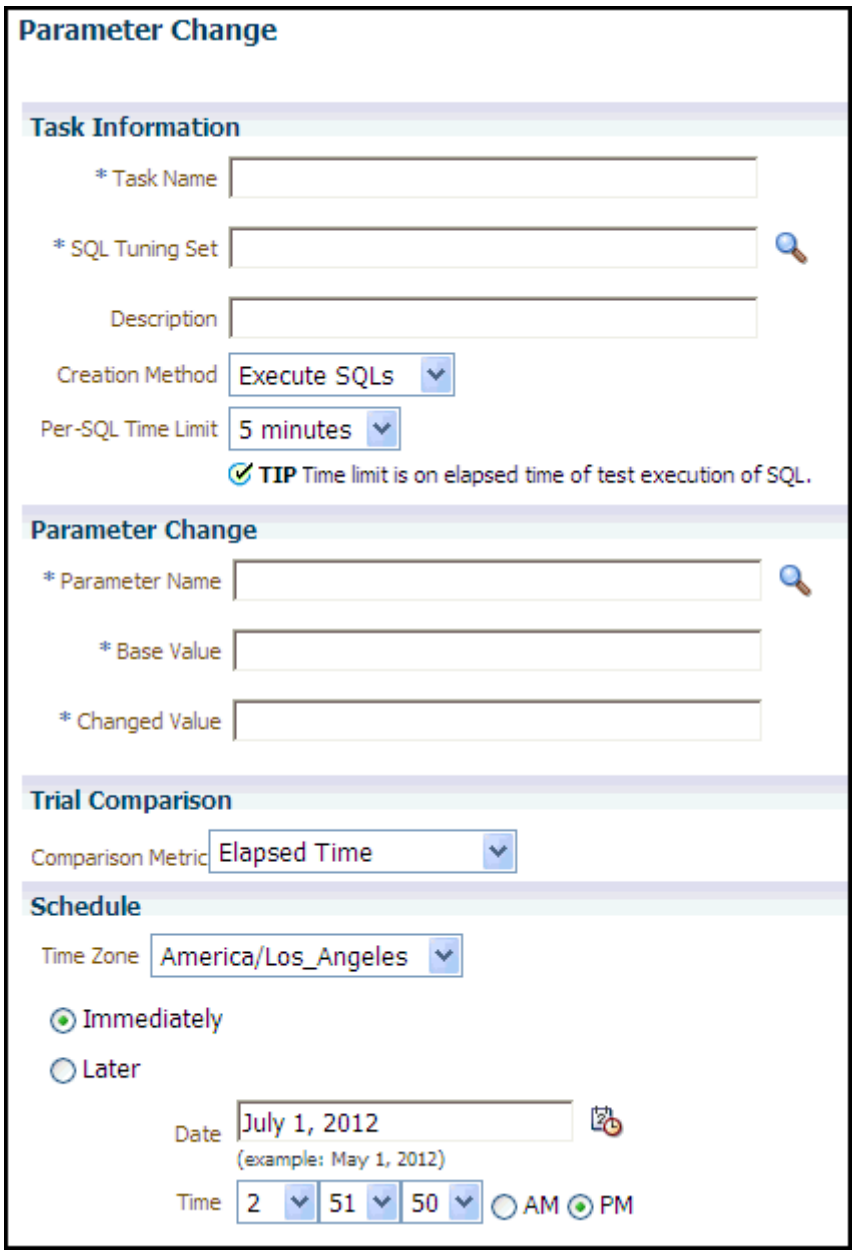
ノート:

他のタイプのシステム変更の分析タスクを作成する場合は、ガイド付きワークフローをかわりに使用してください。詳細は、「ガイド付きワークフローの使用」を参照してください。

SQLパフォーマンス・アナライザのパラメータの変更ワークフローを使用するには:

1. 「SQLパフォーマンス・アナライザ」ホーム・ページの「SQLパフォーマンス・アナライザ・ワークフロー」で、「パラメータの変更」をクリックします。

   「パラメータの変更」ページが表示されます。



2. 「タスク名」フィールドに、タスクの名前を入力します。
3. 「SQLチューニング・セット」フィールドに、分析するSQLワークロードが含まれているSQLチューニング・セットの名前を入力します。

   または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを検索します。

   選択したSQLチューニング・セットが「SQLチューニング・セット」フィールドに表示されます。
4. 「説明」フィールドに、オプションでタスクの説明を入力します。
5. 「作成方法」リストで、次のいずれかのアクションを実行して、SQL試行の作成方法および生成されるコンテンツを確認します。

   - 「SQLの実行」を選択します。

     このSQL試行では、実際にSQL文を実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。
   - 「計画の生成」を選択します。

このSQL試行では、オプティマイザが起動され、実際にSQL文が実行されない場合にのみ実行計画が作成されます。

6. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。

   - 「5分」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。

   - 「無制限」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。

   - 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。

7. 「パラメータの変更」セクションで、次のステップを実行します。

   - 「パラメータ名」フィールドに、変更する初期化パラメータの名前を入力するか、または検索アイコンをクリックして、「検索と選択: 初期化パラメータ」ウィンドウを使用して初期化パラメータを選択します。
   - 「ベース値」フィールドに、初期化パラメータの現在の値を入力します。
   - 「変更された値」フィールドに、初期化パラメータの新しい値を入力します。

8. 「比較メトリック」リストで、分析に使用する比較メトリックを選択します。

   - ステップ5で「計画の生成」を選択した場合は、「オプティマイザ・コスト」を選択します。
   - ステップ5で「SQLの実行」を選択した場合は、次のいずれかのオプションを選択します。
     - 経過時間
     - CPU時間
     - ユーザーI/O時間
     - バッファ読取り
     - 物理I/O
     - オプティマイザ・コスト
     - I/Oインターコネクト・バイト

   複数の比較メトリックを使用して比較分析を実行するには、異なるメトリックを使用してこの手順を繰り返すことによって比較分析を別々に実行します。

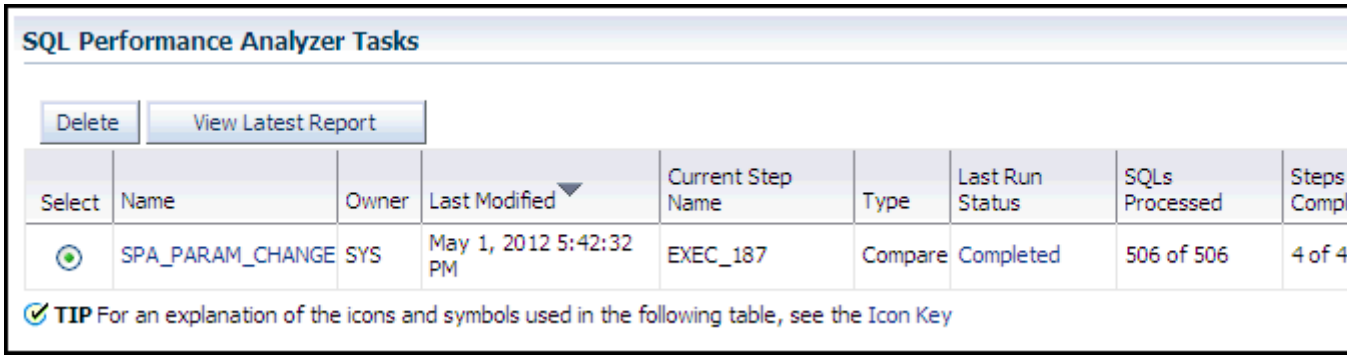
9. 「スケジュール」セクションで、次の手順を実行します。

   - 「タイムゾーン」リストで、タイムゾーン・コードを選択します。
   - 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。

10. 「発行」をクリックします。

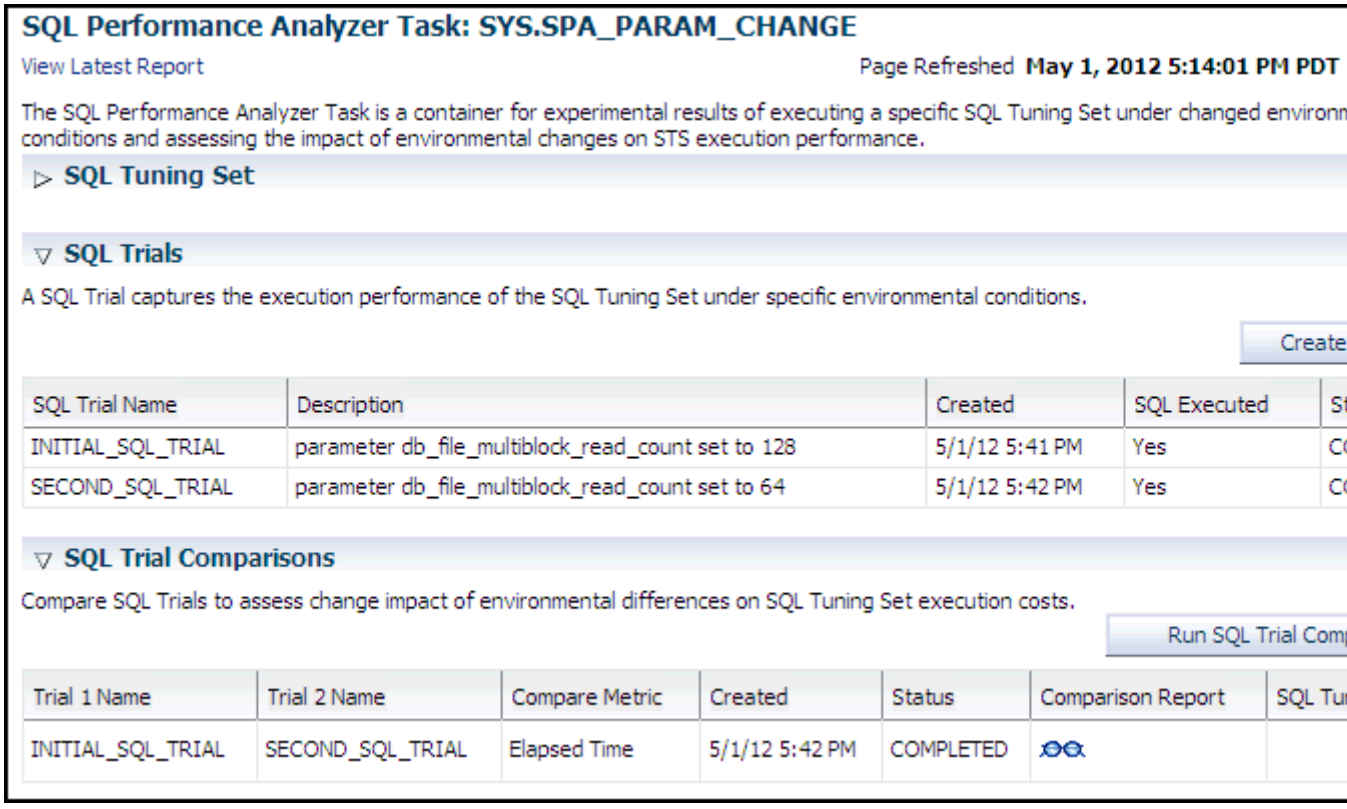
「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。

「SQLパフォーマンス・アナライザのタスク」セクションに、このタスクのステータスが表示されます。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。タスクが完了すると、「ステータス」フィールドが「完了」に変更されます。



11. 「SQLパフォーマンス・アナライザのタスク」セクションで、タスクを選択して「名前」列のリンクをクリックします。

「SQLパフォーマンス・アナライザのタスク」ページが表示されます。



このページには、次のセクションが含まれています。

- SQLチューニング・セット

  このセクションには、SQLチューニング・セットに関する情報(名前、所有者、説明、SQLチューニング・セットに含まれているSQL文の数など)の概要が表示されます。

- SQL試行

  このセクションには、SQLパフォーマンス・アナライザのタスクで使用されるSQL試行を示す表が含まれています。

- SQL試行比較

  このセクションには、SQL試行比較の結果を示す表が含まれています。

12. 「比較レポート」列のアイコンをクリックします。

「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。

13. パフォーマンス分析の結果を確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

14. パフォーマンスの低下が特定された場合は、「SQLチューニング・レポート」列のアイコンをクリックしてSQLチューニング・レポートを表示します。

オプティマイザ統計ワークフローの使用

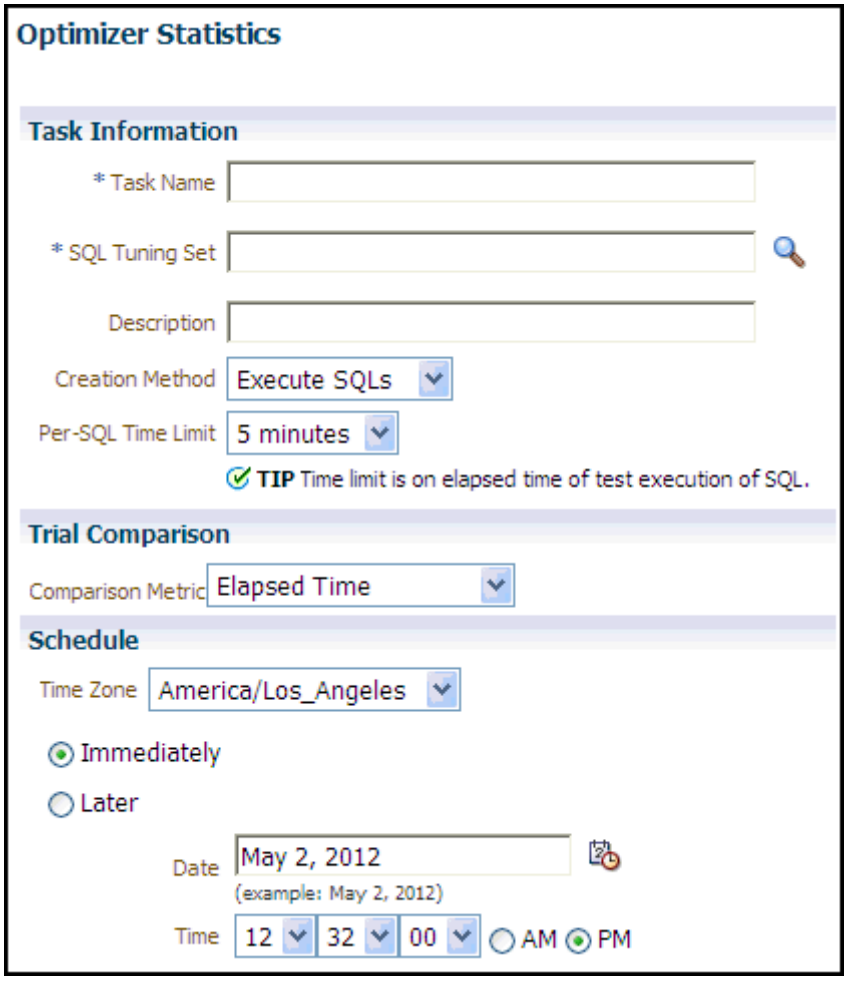
オプティマイザ統計ワークフローを使用すると、オプティマイザ統計の変更によるSQLワークロードのパフォーマンスに対する影響を分析できます。

SQLパフォーマンス・アナライザは、テスト・セッションで保留中のオプティマイザ統計を使用可能にすることで、新しいオプティマイザ統計の影響をテストします。最初のSQL試行では、SQLチューニング・セットのベースラインのパフォーマンスが測定され、2番目のSQL試行では、保留中のオプティマイザ統計が使用されます。続いて、この2つのSQL試行に対して比較レポートを実行できます。

オプティマイザ統計ワークフローを使用するには:

1. 「SQLパフォーマンス・アナライザ」ホーム・ページの「SQLパフォーマンス・アナライザ・ワークフロー」で、「オプティマイザ統計」をクリックします。

「オプティマイザ統計」ページが表示されます。



2. 「タスク名」フィールドに、タスクの名前を入力します。
3. 「SQLチューニング・セット」フィールドに、分析するSQLワークロードが含まれているSQLチューニング・セットの名前を入力します。

   または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを検索します。

   選択したSQLチューニング・セットが「SQLチューニング・セット」フィールドに表示されます。
4. 「説明」フィールドに、オプションでタスクの説明を入力します。
5. 「作成方法」リストで、次のいずれかのアクションを実行して、SQL試行の作成方法および生成されるコンテンツを確認します。
   - 「SQLの実行」を選択します。

     このSQL試行では、実際にSQL文を実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。
   - 「計画の生成」を選択します。

     このSQL試行では、オプティマイザが起動され、実際にSQL文が実行されない場合にのみ実行計画が作成されます。
6. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。
   - 「5分」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。
   - 「無制限」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。
   - 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。
7. 「比較メトリック」リストで、比較分析に使用する比較メトリックを選択します。
   - 経過時間
   - CPU時間
   - ユーザーI/O時間
   - バッファ読取り
   - 物理I/O
   - オプティマイザ・コスト
   - I/Oインターコネクト・バイト

   SQL試行で実行計画のみを生成するように選択した場合、選択可能な比較メトリックは「オプティマイザ・コスト」のみ

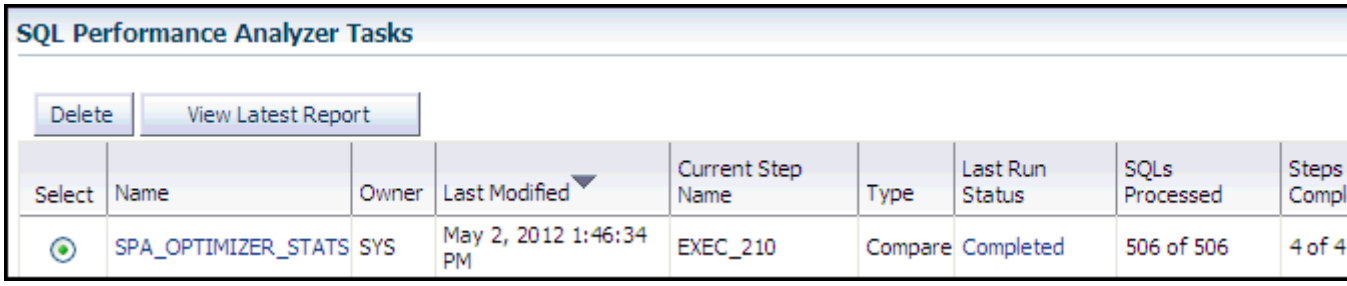
です。

複数の比較メトリックを使用して比較分析を実行するには、異なるメトリックを使用してこの手順を繰り返すことによって比較分析を別々に実行します。

8. 保留中のオプティマイザ統計が収集されていることを確認し、「収集された保留中のオプティマイザ統計」を選択します。
9. 「スケジュール」セクションで、次の手順を実行します。
   - 「タイムゾーン」リストで、タイムゾーン・コードを選択します。
   - 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。
10. 「発行」をクリックします。

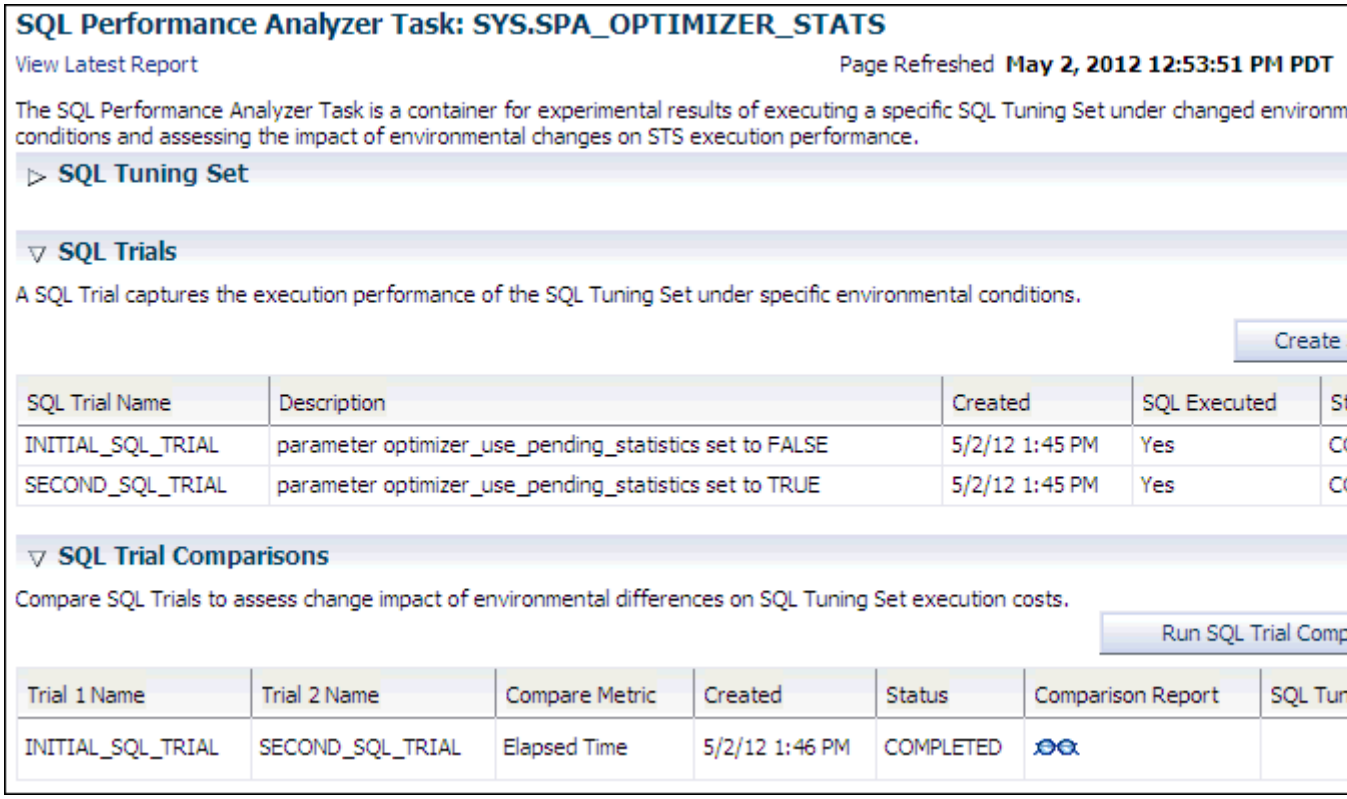
「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。

「SQLパフォーマンス・アナライザのタスク」セクションに、このタスクのステータスが表示されます。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。タスクが完了すると、「ステータス」フィールドが「完了」に変更されます。



11. 「SQLパフォーマンス・アナライザのタスク」セクションで、タスクを選択して「名前」列のリンクをクリックします。

「SQLパフォーマンス・アナライザのタスク」ページが表示されます。



このページには、次のセクションが含まれています。

- SQLチューニング・セット

  このセクションには、SQLチューニング・セットに関する情報(名前、所有者、説明、SQLチューニング・セットに含まれているSQL文の数など)の概要が表示されます。

- SQL試行

  このセクションには、SQLパフォーマンス・アナライザのタスクで使用されるSQL試行を示す表が含まれています。

- SQL試行比較

  このセクションには、SQL試行比較の結果を示す表が含まれています。

12. 「比較レポート」列のアイコンをクリックします。

「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。

13. パフォーマンス分析の結果を確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

パフォーマンスの低下が見つかった場合は、SQL計画ベースラインとSQLチューニング・アドバイザを使用して修正できます。保留中のオプティマイザ統計で得られるパフォーマンスが満足できるものであれば、公開して使用することができます。

Exadataシミュレーション・ワークフローの使用

Exadataシミュレーション・ワークフローを使用すると、SQLワークフローのパフォーマンスに対するExadata Storage Serverインストールの影響をシミュレートできます。

Oracle Exadataは、非常に大きなI/O帯域幅を備えているうえ、データベースからストレージへのSQL処理のオフロード機能があります。これによって、Oracle Databaseでは、I/Oインターコネクトを介して送信されるデータ量を大幅に削減可能になると同時に、Exadataストレージ・セルへのCPUリソースのオフロードが可能になります。

SQLパフォーマンス・アナライザは、Exadata Storage Serverインストールをシミュレートし、SQLワークロードに対するI/Oインターコネクトの使用率の減少を測定することで、Exadata SQLオフロード処理の効果を分析することができます。

Exadataシミュレーションを実行するためにシステムのハードウェアや構成を変更する必要はありません。SQLチューニング・セットを選択した後、SQLパフォーマンス・アナライザによりタスクが作成され、Exadata Storage Serverシミュレーションが無効の状態で最初の試行が実行されます。SQLパフォーマンス・アナライザでは次に、Exadata Storage Serverシミュレーションが有効な状態で2回目の試行が実行されます。その後、「I/Oインターコネクト・バイト」比較メトリックを使用して2つの試行が比較され、Oracle Exadataを使用している場合は、Exadataストレージ・セルからデータベースに送信する必要のないデータの量を見積るSQLパフォーマンス・アナライザ・レポートが生成されます。両方のSQL試行で、SQL文が完了するまで実行され、I/Oインターコネクト・バイトの測定値が取得されます(最初の試行が実際のExadataの値で、2番目の試行がシミュレートされたExadataの値です)。I/Oインターコネクト・バイトで測定された変化を確認することによって、Exadataストレージ・セルで実行可能なフィルタリングの量、および通常このデータの処理に使用されるCPUの量のうちデータベースからオフロード可能な量を的確に見積もることができます。

ノート:

Exadata シミュレーションを使用しても計画の変更は発生しません。シミュレーションが I/O インターコネクトの使用

状況の向上に関する測定に重点を置いているため、Exadata Storage Server インストール内で実行計画が変更されることはありません。また、Oracle Exadata はデータベースに送信されるデータ量のみを削減するため、データ圧縮が使用されている場合(次のノートを参照)を除いて、I/O インターコネクト・バイトが増加することはありません。

ノート:

I/O インターコネクト・バイトは、Exadata Storage Server インストールを使用した場合のパフォーマンス変化の影響を測定するための唯一のメトリックであるため、Oracle Exadata をデータ圧縮とともに使用した場合は適切に機能しません。Exadata ストレージ・セルはデータの圧縮解除も行うため、データが圧縮されている場合、Oracle Exadata を使用した状態での SQL 文(2 番目の SQL 試行)の I/O インターコネクト・バイトが、Oracle Exadata を使用しない状態の I/O インターコネクト・バイトよりも増加することがあります。実際にはそうでないにもかかわらず、SQL 文のパフォーマンスが低下したと報告されるため、この比較は誤解を与えます。

ノート:

Exadata シミュレーションのワークフローは、Exadata 以外のハードウェアで Exadata Storage Server インストールをシミュレートするために使用されます。Exadata ハードウェアで変化をテストするには、標準の SQL パフォーマンス・アナライザのワークフローを使用します。

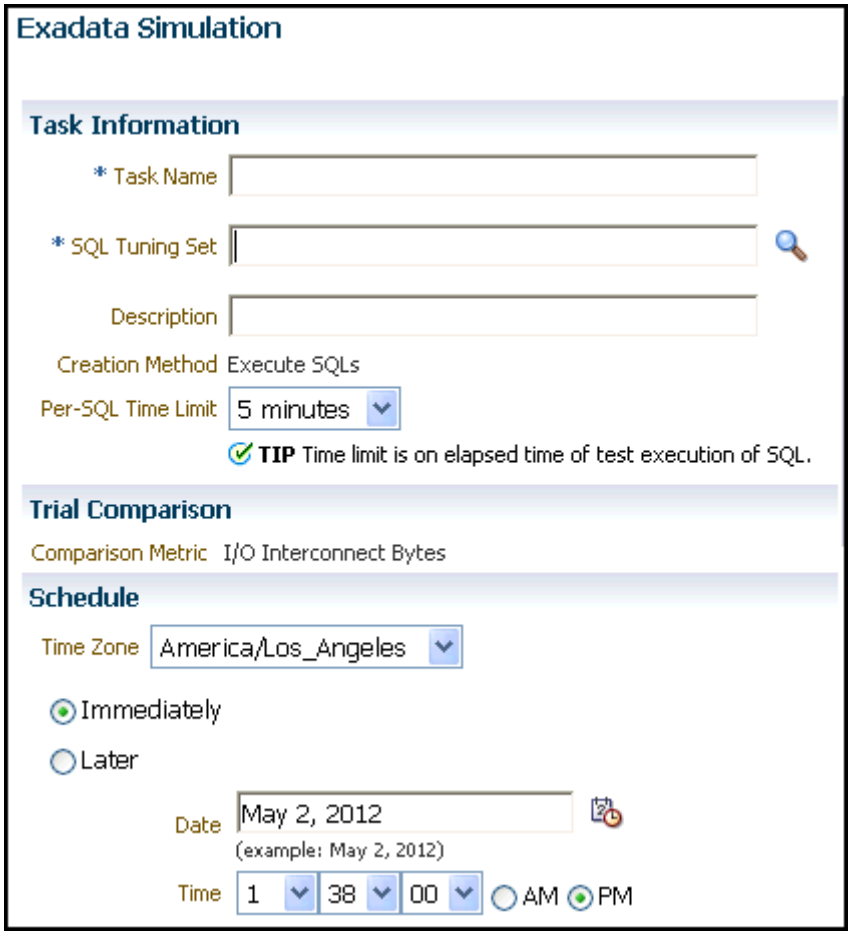
ノート:

Exadata シミュレーションがサポートされているのは、DSS およびデータ・ウェアハウスのワークロードに対してのみです。

SQLパフォーマンス・アナライザのExadataシミュレーション・ワークフローを使用するには:

1. 「SQLパフォーマンス・アナライザ」ホーム・ページの「SQLパフォーマンス・アナライザ・ワークフロー」で、「Exadataシミュレーション」をクリックします。

   「Exadataシミュレーション」ページが表示されます。



2. 「タスク名」フィールドに、タスクの名前を入力します。
3. 「SQLチューニング・セット」フィールドに、分析するSQLワークロードが含まれているSQLチューニング・セットの名前を入力します。

   または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを検索します。

   選択したSQLチューニング・セットが「SQLチューニング・セット」フィールドに表示されます。
4. 「説明」フィールドに、オプションでタスクの説明を入力します。
5. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。

   - 「5分」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。
   - 「無制限」を選択します。

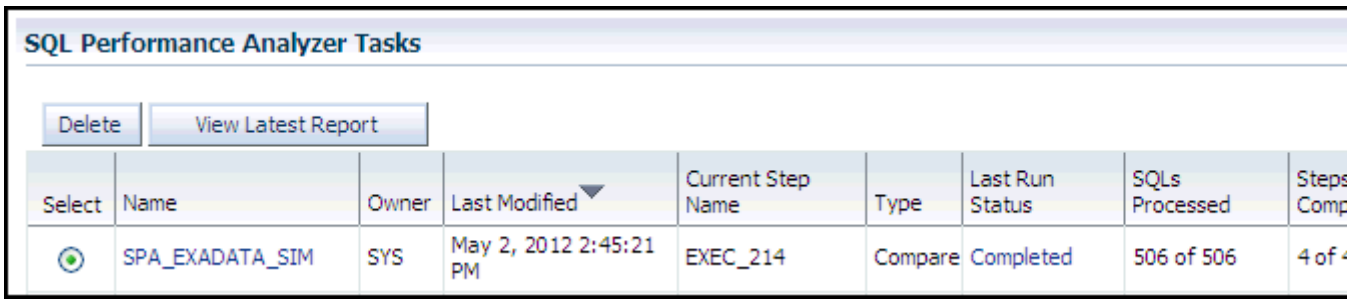
     この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。
   - 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。
6. 「スケジュール」セクションで、次の手順を実行します。

- 「タイムゾーン」リストで、タイムゾーン・コードを選択します。
- 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。

7. 「発行」をクリックします。

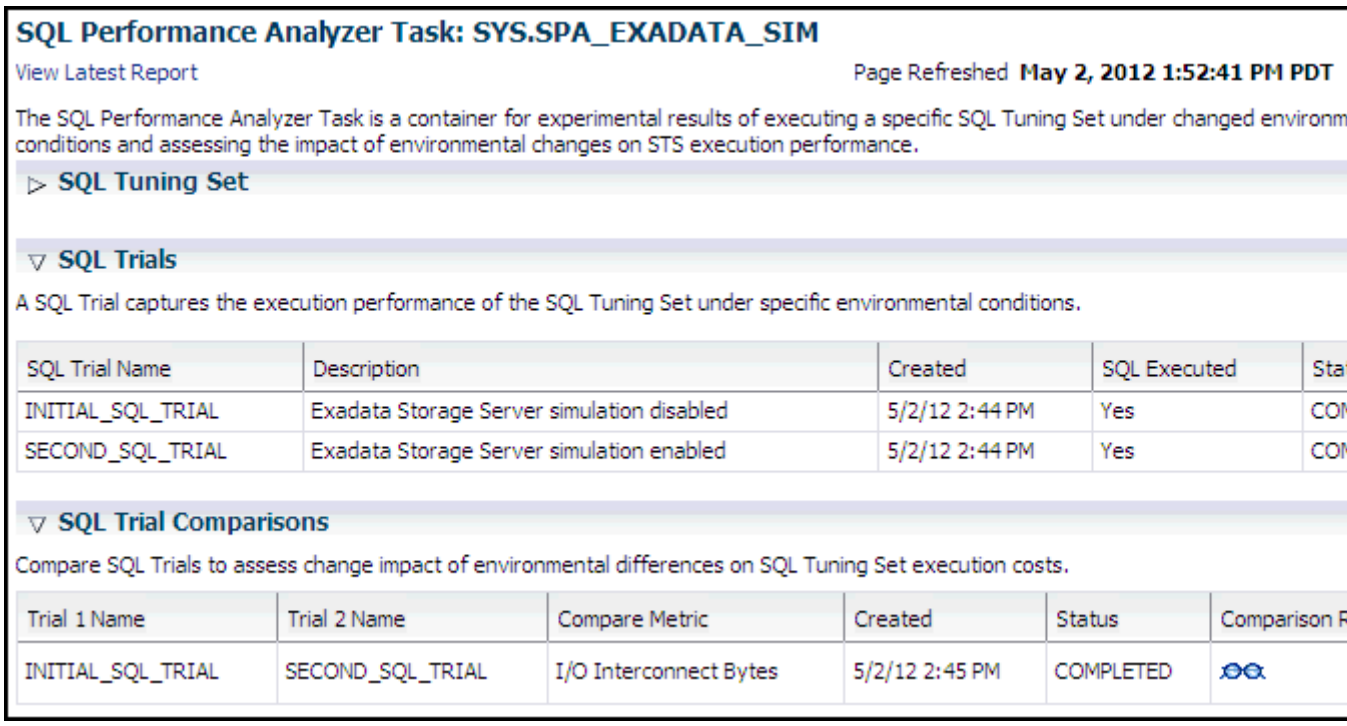
「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。

「SQLパフォーマンス・アナライザのタスク」セクションに、このタスクのステータスが表示されます。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。タスクが完了すると、「ステータス」フィールドが「完了」に変更されます。



8. 「SQLパフォーマンス・アナライザのタスク」セクションで、タスクを選択して「名前」列のリンクをクリックします。

「SQLパフォーマンス・アナライザのタスク」ページが表示されます。



このページには、次のセクションが含まれています。

- SQLチューニング・セット

  このセクションには、SQLチューニング・セットに関する情報(名前、所有者、説明、SQLチューニング・セットに含まれているSQL文の数など)の概要が表示されます。

- SQL試行

  このセクションには、SQLパフォーマンス・アナライザのタスクで使用されるSQL試行を示す表が含まれています。

- SQL試行比較

  このセクションには、SQL試行比較の結果を示す表が含まれています。

9. 「比較レポート」列のアイコンをクリックします。

   「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。

10. パフォーマンス分析の結果を確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

    最初と2番目の試行間のExadataシミュレーションでのSQLパフォーマンスの向上は、このレポートに取得されます。一般に、SQLワークロードに多数の行をスキャンする問合せ、または表の列の小さなサブセットをスキャンする問合せが含まれている場合、より大きな効果を期待できます。逆に、索引付き表または行数が少ない表を問合せるSQLワークロードは、Exadataシミュレーションからの効果は少なくなります。

ガイド付きワークフローの使用

ガイド付きワークフローでは、SQLワークロードに対するすべてのタイプのシステム変更のパフォーマンスの影響をテストできます。SQLパフォーマンスに影響を与える可能性のあるシステム変更のリストについては、「SQLパフォーマンス・アナライザ」を参照してください。

ノート:

分析タスクを作成してデータベース初期化パラメータの変更をテストする場合は、簡略化されたパラメータ変更ワークフローをかわりに使用してください。詳細は、「パラメータ変更ワークフローの使用」を参照してください。

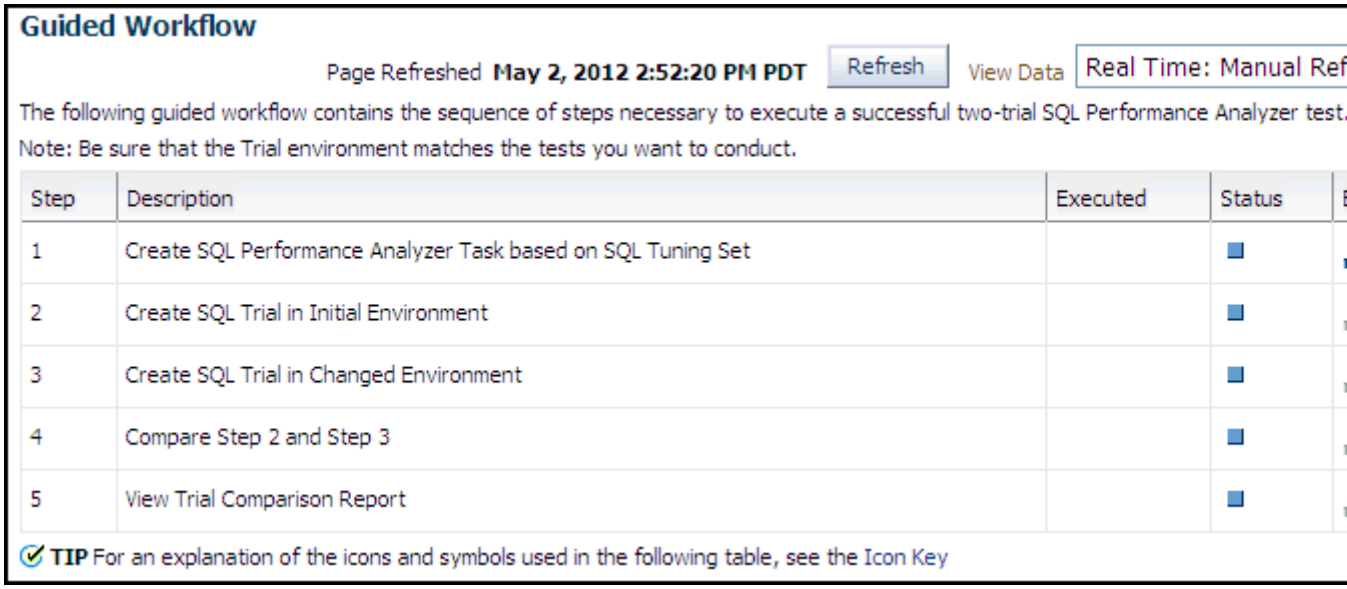
SQLパフォーマンス・アナライザ・タスクのガイド付きワークフローを使用するには:

1. 「SQLパフォーマンス・アナライザ」ホーム・ページの「SQLパフォーマンス・アナライザ・ワークフロー」で、「ガイド付きワークフロー」をクリックします。

   「ガイド付きワークフロー」ページが表示されます。

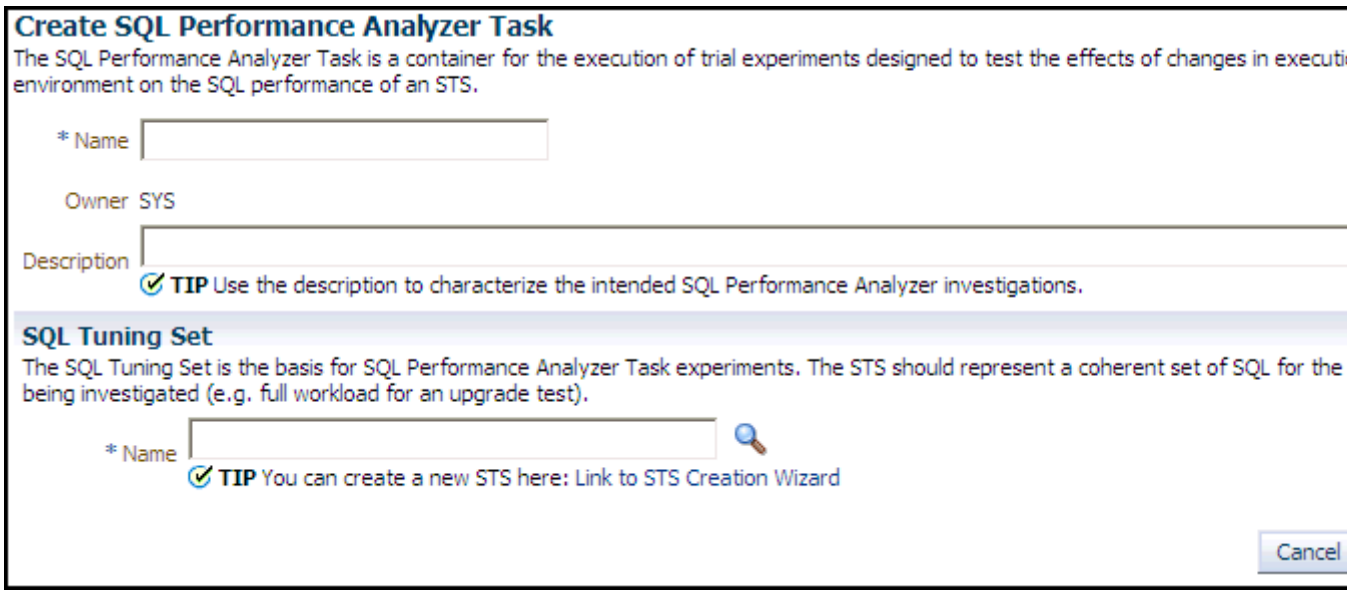
   ガイド付きワークフローを使用すると、任意のタイプのシステム変更を実行した場合のSQLワークロードへのパフォーマンスの影響をテストできます。詳細は、「SQLパフォーマンス・アナライザ」を参照してください。

   このページには、SQLパフォーマンス・アナライザのタスクで必要なステップが順に表示されます。各ステップを順に完了してから、次のステップを開始する必要があります。



2. 「ガイド付きワークフロー」ページで、ステップ1の「SQLチューニング・セットに基づくSQLパフォーマンス・アナライザのタスクの作成」の「実行」アイコンをクリックします。

   「SQLパフォーマンス・アナライザのタスクの作成」ページが表示されます。



3. 「名前」フィールドに、タスクの名前を入力します。
4. 「説明」フィールドに、オプションでタスクの説明を入力します。
5. 「SQLチューニング・セット」の下の「名前」フィールドに、分析するSQLワークロードが含まれているSQLチューニング・セットの名前を入力します。

   または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを選択します。

6. 「作成」をクリックします。

   「ガイド付きワークフロー」ページが表示されます。

   このステップの「ステータス」アイコンがチェック・マークに変わり、次のステップの「実行」アイコンが有効になります。

7. 分析タスクが作成されると、「変更前のSQL試行の作成」の説明に従って、SQLチューニング・セットのSQL文を実行して変更前のパフォーマンス・データを作成できます。

APIを使用した分析タスクの作成

この項では、DBMS_SQLPA.CREATE_ANALYSIS_TASKファンクションを使用してSQLパフォーマンス・アナライザのタスクを作成する方法について説明します。タスクは、SQLパフォーマンス・アナライザの実行時の入力および結果が格納されるデータベース・コンテナです。

操作を進める前に、パフォーマンスの分析に使用するSQLワークロードを本番システムのSQLチューニング・セットに取得して、パフォーマンスの分析を実行するテスト・システムに転送します。詳細は、「SQLワークロードの取得」を参照してください。

分析タスクを作成するには:

- 次のパラメータを使用してCREATE_ANALYSIS_TASKファンクションをコールします。
  - task_nameを設定して、SQLパフォーマンス・アナライザのタスクの任意の名前を指定します。
  - sqlset_nameをSQLチューニング・セットの名前に設定します。
  - sqlset_ownerをSQLチューニング・セットの所有者に設定します。デフォルトは現在のスキーマ所有者です。
  - basic_filterを、SQLチューニング・セットからSQLを除外するために使用するSQL述語に設定します。
  - order_byを設定して、SQL文が実行される順番を指定します。

    このパラメータを使用すると、時間制限に達しても、より重要なSQL文が処理され、スキップされないようにすることができます。
  - top_sqlを設定して、除外およびランク付け後にSQL文の上位の番号のみが考慮されるようにします。

  次の例は、ファンクション・コールを示しています。

```
VARIABLE t_name VARCHAR2(100);
EXEC :t_name := DBMS_SQLPA.CREATE_ANALYSIS_TASK(sqlset_name => 'my_sts', -
       task_name => 'my_spa_task');
```

分析タスクが作成されると、「変更前のSQL試行の作成」の説明に従って、SQLチューニング・セットのSQL文を実行して変更前のパフォーマンス・データを作成できます。

関連項目:

- DBMS_SQLPA.CREATE_ANALYSIS_TASKファンクションについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用した分析タスクの構成

この項では、作成されたSQLパフォーマンス・アナライザのタスクを構成する方法について説明します。
DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャを使用してパラメータを指定することによって、分析タスクを構成できます。

この項では、次の項目について説明します。

- APIを使用した分析タスクでのマルチテナント・コンテナ・データベース識別子の再マッピング
- 分析タスクでのトリガーの実行の構成
- 分析タスクでコールにより返される日付の構成
- 分析タスクでフェッチする行数の構成
- 分析タスクの並列度の構成
- SQLパフォーマンス・アナライザの使用によるSQL結果セットの検証

APIを使用した分析タスクの実行計画の比較方法の構成

SQLパフォーマンス・アナライザ・タスクが実行計画の行ごとの比較をいつ行うかを決定する比較方法を構成できます。デフォルトでは、SQLパフォーマンス・アナライザ・タスクは、計画のハッシュ値が不明の場合にのみ、実行計画の行ごとの比較を実行します。

分析タスクの実行計画の比較方法を構成するには:

- SET_ANALYSIS_TASK_PARAMETERプロシージャを使用して、PLAN_LINES_COMPARISONパラメータの値を設定します。

  表3-1は、PLAN_LINES_COMPARISONパラメータの有効な値を示しています。

表3-1 SQLパフォーマンス・アナライザ・タスクの実行計画方法

方法	説明
ALWAYS	分析タスクで常に実行計画の行ごとの比較が実行されます。
AUTO	最初の SQL 試行の計画のハッシュ値の計算が変更された場合、または 2 番目の SQL 試行が使用できない場合にのみ、分析タスクで実行計画の行ごとの比較が実行されます。
NONE	計画のハッシュ値が不明の場合にのみ、分析タスクで実行計画の行ごとの比較が実行されます。これがデフォルト値です。

次の例に、分析タスクの実行計画方法をAUTOに設定する方法を示します。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
        parameter => 'PLAN_LINES_COMPARISON', -
        value => 'AUTO');
```

関連項目:

- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

APIを使用したExadataシミュレーションの分析タスクの構成

SQLパフォーマンス・アナライザがOracle Exadataシミュレーションを実行するように構成できます。SQLパフォーマンス・アナライザが、SQLワークロードのパフォーマンスに関するExadata Storage Serverインストールの影響をどのようにシミュレートするか

については、「Exadataシミュレーション・ワークフローの使用」を参照してください。

分析タスクでExadataシミュレーションを有効にするには:

- 次に例を示すとおり、変更後のSQL試行を作成する前に、SET_ANALYSIS_TASK_PARAMETERプロシージャをコールします。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'CELL_SIMULATION_ENABLED', -
       value => 'TRUE');
```

  この操作によって、変更後のSQL試行を作成する際に、Exadataシミュレーションが有効になります。作成した変更後のSQL試行は、Exadataシミュレーションを無効にして作成した変更前のSQL試行と比較することができます。

または、tcellsim.sqlスクリプトを使用してExadataシミュレーションを実行することもできます。

tcellsim.sqlを使用してExadataのシミュレーションを実行するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/tcellsim.sql
```

2. 使用するSQLチューニング・セットの名前および所有者を入力します。

```
Enter value for sts_name: MY_STS
Enter value for sts_owner: IMMCHAN
```

   スクリプトによって次の4つのステップが自動的に実行されます。

   - SQLパフォーマンス・アナライザのタスクの作成
   - Exadataシミュレーションを無効にした状態でのSQL文のテスト実行
   - Exadataシミュレーションを有効にした状態でのSQL文のテスト実行
   - パフォーマンスの比較と分析レポートの生成

関連項目:

- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

APIを使用した分析タスクでのマルチテナント・コンテナ・データベース識別子の再マッピング

取得したSQL文をSQLチューニング・セットに格納し、SQLパフォーマンス・アナライザ・タスクの作成時に入力ソースとして使用できます。その後、SQLパフォーマンス・アナライザはこのSQLチューニング・セットをSQL試行のテスト実行または実行計画生成のソースとして使用します。

非CDBからマルチテナント・コンテナ・データベース(CDB)に転送されたSQLチューニング・セットを入力ソースとして使用する場合、SQLチューニング・セット内のSQL文のCDB識別子を再マッピングして、STSをCDB内で使用できるようにする必要があります。CDB識別子を再マッピングすると、SQLチューニング・セット内の各SQL文はCDB内の対応するプラガブル・データベース(PDB)に再マッピング可能なCDB識別子に関連付けられます。

通常CDB識別子は、SQLチューニング・セットを非CDBからCDBに転送する際に再マッピングする必要があります。この場合、単純にSQLチューニング・セットをSQLパフォーマンス・アナライザの入力ソースとして使用します。ただし、CDB識別子がまだ再マッピングされていないSQLチューニング・セットを使用している場合、SQLパフォーマンス・アナライザのタスク・プロパティとして再マッピングを指定できます。

分析タスク用にCDB識別子を再マッピングするには:

- 次の例のように、SET_ANALYSIS_TASK_PARAMETERプロシージャを使用します。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'non_cdb_spa1', -
       parameter => 'CON_DBID_MAPPING', -
       value => '1234:5678,1357:2468');
```

  この例では、CDB識別子の1234と1357は、それぞれ5678と2468に再マッピングされます。

CDB識別子が再マッピングされると、古いCDB識別子と一致する新しいCDB識別子をSQLパフォーマンス・アナライザは使用して、CDB内の適切なPDBでSQL文を実行します。

関連項目:

- SQLチューニング・セットの転送の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

分析タスクでのトリガーの実行の構成

分析タスクでトリガーを実行するかどうかを構成できます。デフォルトでは、トリガーは、SQLパフォーマンス・アナライザで実行されます。

分析タスクでのトリガーの実行を構成するには:

- SET_ANALYSIS_TASK_PARAMETERプロシージャを使用して、EXECUTE_TRIGGERSパラメータの値を設定します。

  表3-2は、EXECUTE_TRIGGERSパラメータの有効な値を示しています。

表3-2 EXECUTE_TRIGGERSパラメータの有効な値

値	説明
FALSE	TEST EXECUTE の EXECUTE_FULLDML モードであってもトリガーは SQL パフォーマンスアナライザで実行されません。これがデフォルト値です。
TRUE	すべてのトリガーは、SQL パフォーマンス・アナライザで実行されます。

次に、EXECUTE_TRIGGERSパラメータの値をFALSEに設定する例を示します。この設定では、トリガーはSQLパフォーマンス・アナライザで実行されません。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'EXECUTE_TRIGGERS', -
       value => 'FALSE');
```

関連項目:

- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

分析タスクでコールにより返される日付の構成

分析タスクでSYSDATEを参照するSQL文の処理方法を構成できます。

分析タスクでSYSDATEのコールによって返される日付を構成するには:

REPLACE_SYSDATE_WITHパラメータを設定すると、タスク実行内のSYSDATEへのすべてのコールは、このパラメータで指定された日付を返します。これは、SPAタスクへの入力がSQLチューニング・セット(STS)である場合に使用できます。

- SET_ANALYSIS_TASK_PARAMETERプロシージャを使用して、REPLACE_SYSDATE_WITHパラメータの値を設定します。

表3-3は、REPLACE_SYSDATE_WITHパラメータの有効な値を示しています。

表3-3は、REPLACE_SYSDATE_WITHパラメータの有効な値を示しています。

値	説明
CURRENT_SYSDATE	タスク実行内での SYSDATE へのすべてのコールは現在の SYSDATE を返します。これはデフォルトです。
SQLSET_SYSDATE	SYSDATE コールを含むすべての SQL 文で、SQL パフォーマンス・アナライザは、その値を、該当にの SQL 文に対する DBA_SQLSET_STATEMENTS ビューの LAST_EXEC_START_TIME 列の値で置き換えます。

ノート:

このパラメータの設定は、SQL パフォーマンス・アナライザのタスクの実行以外の SYSDATE へのコールには影響しません。

次に、REPLACE_SYSDATE_WITHパラメータの値をSQLSET_SYSDATEに設定する例を示します。これにより、タスクの実行内のSYSDATEへのコールでは、SQLチューニング・セットでSYSDATEが返されます。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'REPLACE_SYSDATE_WITH', -
       value => 'SQLSET_SYSDATE');
```

関連項目:

- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBA_SQLSET_STATEMENTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

分析タスクでフェッチする行数の構成

分析タスクでSQL文に対してフェッチする行数を構成できます。

分析タスクでフェッチする行数を構成するには:

- SET_ANALYSIS_TASK_PARAMETERプロシージャを使用して、NUM_ROWS_TO_FETCHパラメータの値を設定します。

  表3-4に、NUM_ROWS_TO_FETCHパラメータの有効な値を示します。

表3-4 NUM_ROWS_TO_FETCHパラメータの有効な値

値	説明
ALL_ROWS	SQL のすべての行をフェッチします。これがデフォルト値です。
AUTO	結果行の数は、SQL チューニング・セットで取得されたオプティマイザ環境の OPTIMIZER_MODE パラメータの値を使用して決定されます。OPTIMIZER_MODE の値が ALL_ROWS の場合、すべての結果行がフェッチされます。値が FIRST_ROWS_n の場合、n 個の結果行が SQL パフォーマンス・アナライザによってフェッチされます。
AVERAGE	結果行の数は、処理される行数の合計および SQL チューニング・セット内の各 SQL の合計実行回数の比率として計算されます。
有効な数値	結果行の数は、この指定値と等しくなり、フェッチする行数が少ない場合はこの値より少なくなります。

次に、NUM_ROWS_TO_FETCHパラメータの値をALL_ROWSに設定する例を示します。これにより、SQLのすべての行がフェッチされます。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'NUM_ROWS_TO_FETCH', -
       value => 'ALL_ROWS');
```

関連項目:

- DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- OPTIMIZER_MODEデータベース初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

分析タスクの並列度の構成

並列度の設定および同時SQL実行の有効化が可能です。

分析タスクの並列度を構成するには:

- SET_ANALYSIS_TASK_PARAMETERプロシージャを使用して、TEST_EXECUTE_DOPパラメータの値を設定します。

次の表は、TEST_EXECUTE_DOPパラメータの有効な値を示しています。

表3-5 TEST_EXECUTE_DOPパラメータの有効な値

値	説明
0	これはデフォルト値です。タスクは順次実行されます。
以上 2	同時実行が有効になっています。

次に、TEST_EXECUTE_DOPパラメータの値を4に設定して同時実行を有効化する方法の例を示します。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'TEST_EXECUTE_DOP', -
       value => 4);
```

ノート:

同時実行は、EXPLAIN PLAN および TEST EXECUTE 実行タイプでのみサポートされています。

関連項目:

DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

SQLパフォーマンス・アナライザの使用によるSQL結果セットの検証

2つのSQLパフォーマンス・アナライザ(SPA)試行のテスト実行時に、SQL結果セットの比較がサポートされるようになりました。

SQL結果セットが検証され、行または値の数が一致していない場合は、SQLパフォーマンス・アナライザ・レポートに記録されます。SQL結果セット検証は、COMPARE_RESULTSETパラメータによって制御されます。

SET_ANALYSIS_TASK_PARAMETERプロシージャは、COMPARE_RESULTSETパラメータの値を設定するのに使用されます。

次の表は、COMPARE_RESULTSETパラメータの有効な値を示しています。

表3-6 COMPARE_RESULTSETパラメータの有効な値

値	説明
TRUE	SQL 結果セットの比較が実行されます。

値	説明
FALSE	SQL 結果セットの比較は実行されません。

次に、COMPARE_RESULTSETパラメータの値をTRUEに設定する方法の例を示します。

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'COMPARE_RESULTSET', -
       value => 'TRUE');
```

関連項目:

DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETERプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

4 変更前のSQL試行の作成

SQLパフォーマンス・アナライザのタスクを作成し、入力ソースとしてSQLチューニング・セットを選択したら、テスト・システムの初期環境を設定する必要があります。テスト・システムのデータベース環境を設定するには、SQLの最適化およびパフォーマンスに影響する必要な環境変更を手動で行う必要があります。これらの変更には、初期化パラメータの変更、オプティマイザ統計の収集または設定、索引の作成などがあります。可能なかぎり本番システムと同様にテスト・システムを構築することをお薦めします。Enterprise Managerの専用ワークフローでは、両方のSQL試行が自動作成され、テスト・セッションに限定された変更が行われるため、このプロセスが簡単になっています。

ノート:

SQL 試行は、パブリック・データベース・リンクにアクセスすることで、リモート・システム上で実行することもできます。リモートの SQL 試行を実行する場合は、SQL 文が実行されるリモート・データベースのデータベース・バージョンが、その接続先のデータベースのデータベース・バージョン以下であることが必要です。Oracle Database リリース 11.2.0.2 以上では、Oracle Active Data Guard インスタンスなどの読取り専用データベースをリモート・データベースとして指定できます。

テスト・システムの環境が適切に構成されたら、変更前のバージョンのパフォーマンス・データを構築した後で、システム変更を実行できます。次のいずれかの方法で、SQLパフォーマンス・アナライザを使用してSQL試行を作成できます。

- ワークロード内のSQL文の実行
- ワークロード内のSQL文に対する実行計画の生成
- SQLチューニング・セットからのパフォーマンス・データおよび実行計画のロード(APIのみ)

この章では、変更前のSQL試行の作成方法について説明します。内容は次のとおりです。

- Enterprise Managerを使用した変更前のSQL試行の作成
- APIを使用した変更前のSQL試行の作成

ノート:

変更前の SQL 試行を作成するための主要なインタフェースは、Oracle Enterprise Manager です。なんらかの理由で Oracle Enterprise Manager を使用できない場合は、DBMS_SQLPA PL/SQL パッケージを使用して変更前の SQL 試行を作成できます。

関連項目:

- 「テスト・システムの設定」
- 「変更前のSQLパフォーマンスの測定」

Enterprise Managerを使用した変更前のSQL試行の作成

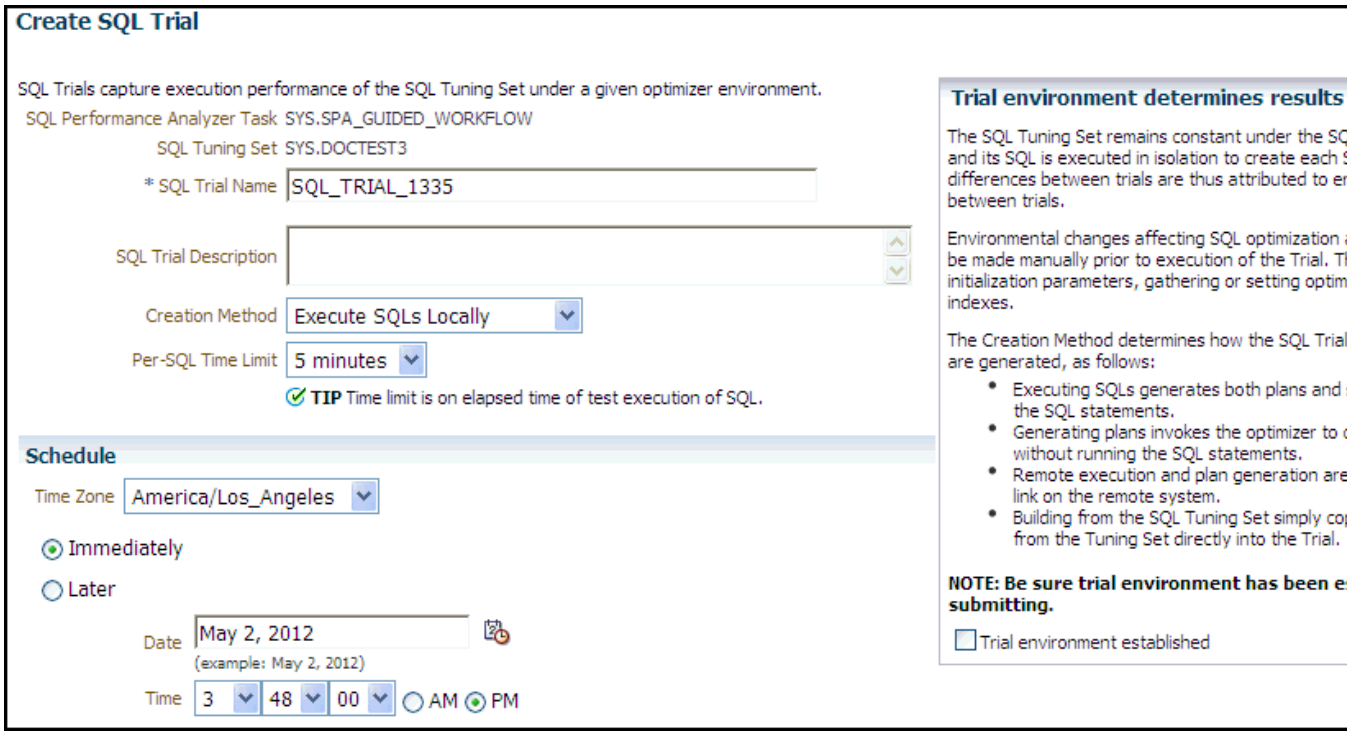
この項では、Oracle Enterprise Managerを使用して変更前のSQLパフォーマンス・データを収集する方法について説明します。

変更前のSQL試行を作成する前に、SQLパフォーマンス・アナライザのタスクを作成する必要があります。詳細は、「分析タスクの作成」を参照してください。

Enterprise Managerを使用して変更前のSQL試行を作成するには:

1. 「ガイド付きワークフロー」ページで、「初期環境へのSQL試行の作成」ステップの「実行」アイコンをクリックします。

   「SQL試行の作成」ページが表示されます。選択したSQLチューニング・セットのサマリー(SQLワークロードを含む)が表示されます。



2. 「SQL試行名」フィールドに、SQL試行の名前を入力します。
3. 「SQL試行の説明」フィールドに、SQL試行の説明を入力します。
4. 「作成方法」リストで、次のいずれかのアクションを実行して、SQL試行の作成方法および生成されるコンテンツを確認します。

   - 「SQLをローカルで実行」を選択します。

     このSQL試行では、実際にテスト・システムでSQL文をローカルに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。

   - 「SQLをリモートで実行」を選択します。

     このSQL試行では、実際にパブリック・データベース・リンクを介して別のテスト・システムでSQL文をリモートに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。

   - 「計画をローカルで実行」を選択します。

このSQL試行では、バインド値およびオプティマイザの構成が考慮された後に、オプティマイザが起動されて、テスト・システムにローカルに実行計画が作成されます。実際にSQL文が実行されることはありません。

- 「計画をリモートで実行」を選択します。

  このSQL試行では、バインド値およびオプティマイザの構成が考慮された後に、オプティマイザが起動されて、パブリック・データベース・リンクを介してリモートで別のテスト・システムに実行計画が作成されます。実際にSQL文が実行されることはありません。

- 「SQLチューニング・セットから作成」を選択します。

  このSQL試行では、直接SQLチューニング・セットから試行に実行計画および実行統計がコピーされます。

これらの各種の方法の詳細は、「変更前のSQLパフォーマンスの測定」を参照してください。

5. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。

   - 「5分」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。

   - 「無制限」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。

   - 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。

6. テスト・システムのデータベース環境が可能なかぎり本番環境と一致していることを確認して、「試行環境設定済み」を選択します。

7. 「スケジュール」セクションで、次の手順を実行します。

   - 「タイムゾーン」リストで、タイムゾーン・コードを選択します。

   - 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。

8. 「発行」をクリックします。

   実行が開始されると、「ガイド付きワークフロー」ページが表示されます。

   実行中、このステップのステータス・アイコンは時計に変わります。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。選択したオプションおよびSQLワークロードのサイズによっては、実行が完了するまで時間がかかる場合があります。実行が完了したら、ステータス・アイコンがチェック・マークに変わり、次のステップの「実行」アイコンが有効になります。

9. 変更前のパフォーマンス・データが作成されたらシステムに変更を行い、「変更後のSQL試行の作成」の説明に従って、変更後のテスト・システム上でSQLチューニング・セットのSQL文を再度実行し、変更後のパフォーマンス・データを作成します。

APIを使用した変更前のSQL試行の作成

この項では、DBMS_SQLPAパッケージを使用して変更前のパフォーマンス・データを構築する方法について説明します。

変更前のSQL試行を作成する前に、SQLパフォーマンス・アナライザのタスクを作成する必要があります。詳細は、「分析タスクの作成」を参照してください。

変更前のSQL試行を作成するには:

- 次のパラメータを使用して、EXECUTE_ANALYSIS_TASKプロシージャをコールします。
  - task_nameパラメータを、実行するSQLパフォーマンス・アナライザのタスクの名前に設定します。
  - 次のいずれかの方法でexecution_typeパラメータを設定します。
    - EXPLAIN PLANに設定して、SQL文を実行せずに、SQLチューニング・セット内のすべてのSQL文の実行計画を生成します。
    - TEST EXECUTE(推奨)に設定し、SQLチューニング・セット内のすべての文を実行し、それらの実行計画および実行統計を生成します。TEST EXECUTEを指定すると、プロシージャによって実行計画および実行統計が生成されます。実行統計によって、パフォーマンスが改善されたSQL文またはパフォーマンスが低下したSQL文をSQLパフォーマンス・アナライザで特定できるようになります。実行計画を生成するのみでなく、実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。
    - SQL試行の実行統計および計画に関してSQLチューニング・セットを参照するように、CONVERT SQLSETを設定します。実行パラメータSQLSET_NAMEおよびSQLSET_OWNERの値も指定する必要があります。
  - execution_nameパラメータを使用して、実行を識別するための名前を指定します。指定しなかった場合、SQLパフォーマンス・アナライザによってタスク実行の名前が自動的に生成されます。
  - execution_paramsパラメータを使用して、実行パラメータを指定します。execution_paramsパラメータは、指定した実行の名前/値ペアとして指定されます。たとえば、次の実行パラメータを設定できます。
    - time_limitパラメータは、グローバルな時間制限を指定して、タイムアウト前にSQLチューニング・セット内のすべてのSQL文を処理します。
    - local_time_limitパラメータは、時間制限を指定して、タイムアウト前にSQLチューニング・セット内の各SQL文を処理します。
    - リモートでテストを実行する場合は、DATABASE_LINKタスク・パラメータをDBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名に設定します。
    - 行ロックの取得と行の変更を含め、DML文を完全に実行するには、EXECUTE_FULLDMLパラメータをTRUEに設定します。SQLパフォーマンス・アナライザは、DML実行の後でロールバックを発行して、永続的な変更が行われないようにします。このパラメータのデフォルト値はFALSEで、データを変更せずにDML文の問合せ部分のみが実行されます。
    - 関連する取得済のinit.ora設定をテスト実行時にリストアするには、APPLY_CAPTURED_COMPILENVパラメータをTRUEに設定します。一般に、SQL試行を実行するのは、環境を変更するときの変更内容のテストが目的であるため、これはデフォルトの動作ではありません。

ただし、この方法は、init.ora設定が変更されていない場合(索引作成などの場合)に使用できます。この方法は、リモートSQL試行ではサポートされていません。

次の例は、システム変更前に行われたファンクション・コールを示しています。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_exec_BEFORE_change');
```

変更前のパフォーマンス・データが作成されたらシステムに変更を行い、「変更後のSQL試行の作成」の説明に従って、変更後のテスト・システム上でSQLチューニング・セットのSQL文を再度実行し、変更後のパフォーマンス・データを作成します。

関連項目:

- DBMS_SQLPA.EXECUTE_ANALYSIS_TASKファンクションについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

5 変更後のSQL試行の作成

変更前のSQLのパフォーマンス・データを計算したら、テスト・システムでシステム変更を実行できます。システム変更を実行する前に、変更前のパフォーマンス・データを生成するために初期環境でSQLワークロードを実行してあることを確認します。たとえば、データベース初期化パラメータを変更することによってSQLパフォーマンスが受ける影響をテストする場合は、データベース初期化パラメータを新しい値に変更する前にSQLワークロードを1回実行します。実行する変更の種類に応じて、SQLのパフォーマンス分析を実行する新しい環境と一致するように、環境をテスト・システムに再構成する必要がある場合があります。

ノート:

SQL 試行は、パブリック・データベース・リンクにアクセスすることで、リモート・システム上で実行することもできます。リモートの SQL 試行を実行する場合は、SQL 文が実行されるリモート・データベースのデータベース・バージョンが、その接続先のデータベースのデータベース・バージョン以下であることが必要です。Oracle Database リリース 11.2.0.2 以上では、Oracle Active Data Guard インスタンスなどの読取り専用データベースをリモート・データベースとして指定できます。

SQLパフォーマンス・アナライザを使用して分析できるシステム変更の例は、「SQLパフォーマンス・アナライザ」に示されています。たとえば、データベース初期化パラメータの変更またはデータベースのアップグレードによってSQLパフォーマンスが受ける影響を確認する場合があります。また、自動データベース診断モニター(ADDM)、SQLチューニング・アドバイザ、SQLアクセス・アドバイザなどのアドバイザから提示された推奨事項に基づいてシステム変更を決定する場合もあります。

システム変更を実行したら、SQLワークロードを再度実行して、変更後のバージョンのパフォーマンス・データを構築できます。SQLパフォーマンス・アナライザでは、SQL文の実行結果が変更後のSQL試行に格納されます。

この項では、変更後のSQL試行の作成方法について説明します。内容は次のとおりです。

- Oracle Enterprise Managerを使用した変更後のSQL試行の作成
- APIを使用した変更後のSQL試行の作成

ノート:

変更後の SQL 試行を作成するための主要なインタフェースは、Oracle Enterprise Manager です。なんらかの理由で Oracle Enterprise Manager を使用できない場合は、DBMS_SQLPA PL/SQL パッケージを使用して変更後の SQL 試行を作成できます。

関連項目:

- 「システム変更の実行」
- 「変更後のSQLパフォーマンスの測定」

Oracle Enterprise Managerを使用した変更後のSQL試行の作成

この項では、Oracle Enterprise Managerを使用して変更後のSQLパフォーマンス・データを収集する方法について説明します。

変更後のSQL試行を作成するためのシステム変更を実行する場合は、その前に、変更前のSQL試行を作成しておく必要があります。詳細は、「変更前のSQL試行の作成」を参照してください。

Enterprise Managerを使用して変更後のSQL試行を作成するには:

1. 「ガイド付きワークフロー」ページで、「変更された環境へのSQL試行の作成」ステップの「実行」アイコンをクリックします。

   「SQL試行の作成」ページが表示されます。
2. 「SQL試行名」フィールドに、SQL試行の名前を入力します。
3. 「SQL試行の説明」フィールドに、SQL試行の説明を入力します。
4. 「作成方法」リストで、次のいずれかのアクションを実行して、SQL試行の作成方法および生成されるコンテンツを確認します。

   - 「SQLをローカルで実行」を選択します。

     このSQL試行では、実際にテスト・システムでSQL文をローカルに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。
   - 「SQLをリモートで実行」を選択します。

     このSQL試行では、実際にパブリック・データベース・リンクを介して別のテスト・システムでSQL文をリモートに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と実行統計の両方が生成されます。
   - 「計画をローカルで実行」を選択します。

     このSQL試行では、オプティマイザが起動され、実際にSQL文が実行されることなく、テスト・システムで実行計画がローカルに作成されます。
   - 「計画をリモートで実行」を選択します。

     SQL試行では、オプティマイザが起動され、実際にSQL文が実行されることなく、パブリック・データベース・リンクを介して別のテスト・システムで実行計画がリモートに作成されます。

   これらの各作成方法に対して、アプリケーション・スキーマおよびデータがローカルまたはリモートのテスト・システムにすでに存在している必要があります。
5. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。

   - 「5分」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。
   - 「無制限」を選択します。

     この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる

時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。

- 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。

6. テストしているシステム変更がテスト・システムで実行されたことを確認して、「試行環境設定済み」を選択します。
7. 「スケジュール」セクションで、次の手順を実行します。
   - 「タイムゾーン」リストで、タイムゾーン・コードを選択します。
   - 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。
8. 「発行」をクリックします。

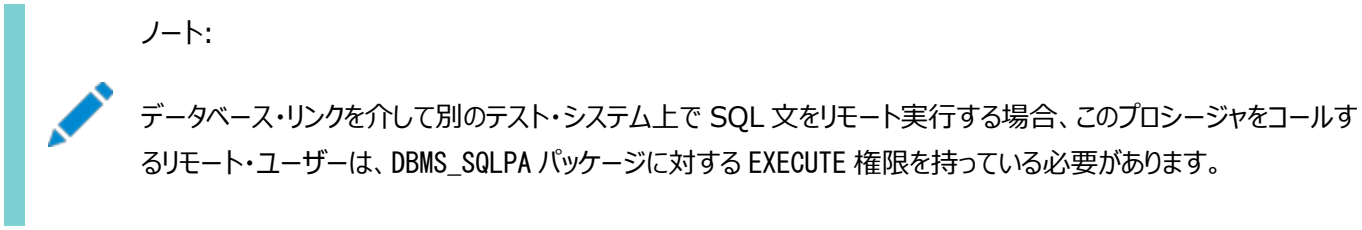
   実行が開始されると、「ガイド付きワークフロー」ページが表示されます。

   実行中、このステップのステータス・アイコンは時計に変わります。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。選択したオプションおよびSQLワークロードのサイズによっては、実行が完了するまで時間がかかる場合があります。実行が完了したら、ステータス・アイコンがチェック・マークに変わり、次のステップの「実行」アイコンが有効になります。
9. 変更後のパフォーマンス・データを構築したら、「SQL試行の比較」で説明する比較分析を実行して、変更前のSQL試行と変更後のSQL試行を比較します。

APIを使用した変更後のSQL試行の作成

この項では、DBMS_SQLPAパッケージを使用して変更後のSQLパフォーマンス・データを収集する方法について説明します。

変更後のSQL試行を作成するためのシステム変更を実行する場合は、その前に、変更前のSQL試行を作成しておく必要があります。詳細は、「変更前のSQL試行の作成」を参照してください。



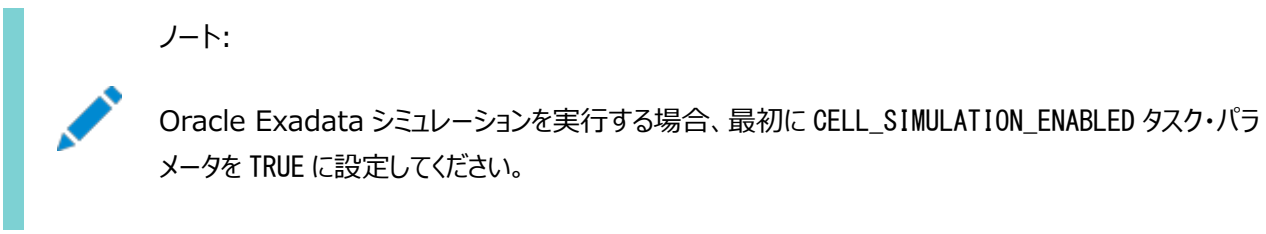
ノート:

データベース・リンクを介して別のテスト・システム上で SQL 文をリモート実行する場合、このプロシージャをコールするリモート・ユーザーは、DBMS_SQLPA パッケージに対する EXECUTE 権限を持っている必要があります。

変更後のSQL試行を作成するには:

- 「APIを使用した変更前のSQL試行の作成」で説明されているパラメータを使用して、EXECUTE_ANALYSIS_TASKプロシージャをコールします。

  execution_nameパラメータには必ず別の値を指定してください。また、execution_typeパラメータに同じ値を使用することで、変更前のSQL試行と同じ方法を使用して変更後のSQL試行を作成することを強くお薦めします。



ノート:

Oracle Exadata シミュレーションを実行する場合、最初に CELL_SIMULATION_ENABLED タスク・パラメータを TRUE に設定してください。

次の例は、システム変更後に行われたファンクション・コールを示しています。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_exec_AFTER_change');
```

変更後のパフォーマンス・データを構築したら、「SQL試行の比較」で説明する比較分析を実行して、変更前のSQL試行と変更後のSQL試行を比較します。

関連項目:

- 「APIを使用したExadataシミュレーションの分析タスクの構成」
- DBMS_SQLPA.EXECUTE_ANALYSIS_TASKファンクションについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

6 SQL試行の比較

変更後のSQLのパフォーマンス・データが構築されたら、SQLパフォーマンス・アナライザを使用した比較分析を実行して、変更前と変更後のSQL試行で収集されたパフォーマンス・データを比較できます。比較分析が完了したら、レポートを生成して、システム変更によりパフォーマンスが改善されたSQL文、パフォーマンスの変化がなかったSQL文またはパフォーマンスが低下したSQL文を特定できます。SQLパフォーマンス・アナライザ・レポートでは、各SQL文のパフォーマンスの変化に関して、2つの主要な影響の大きさを計算します。

- ワークロードに対する影響

  これは、SQL文に対するこの変更が、ワークロードの累積実行時間に与える影響の割合を、実行頻度を考慮して示しています。たとえば、SQL文の累積実行時間を101秒から1秒に改善する変更(変更前の残りのワークロードの実行時間の合計は99秒)の場合、この測定値は50パーセント(2x)になります。

- SQLに対する影響

  これは、SQL文に対するこの変更が、SQL文の応答時間に与える影響の割合を示しています。たとえば、SQL文の応答時間を10秒から1秒に改善する変更の場合、この測定値は90パーセント(10x)になります。

この章では、変更前と変更後のSQL試行のパフォーマンス・データを比較して分析する方法について説明します。内容は次のとおりです。

- Oracle Enterprise Managerを使用したSQL試行の比較
- APIを使用したSQL試行の比較

> ノート:
>
> SQL 試行を比較するための主要インタフェースは、Oracle Enterprise Manager です。なんらかの理由で Oracle Enterprise Manager を使用できない場合は、DBMS_SQLPA PL/SQL パッケージを使用して SQL 試行を比較できます。

関連項目:

「パフォーマンス測定値の比較」

Oracle Enterprise Managerを使用したSQL試行の比較

Oracle Enterprise Managerを使用してSQL試行を比較するには、次のステップを実行します。

- Oracle Enterprise Managerを使用したSQLパフォーマンスの分析
- Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認
- Oracle Enterprise Managerを使用した、パフォーマンスが低下したSQL文のチューニング

Oracle Enterprise Managerを使用したSQLパフォーマンスの分析

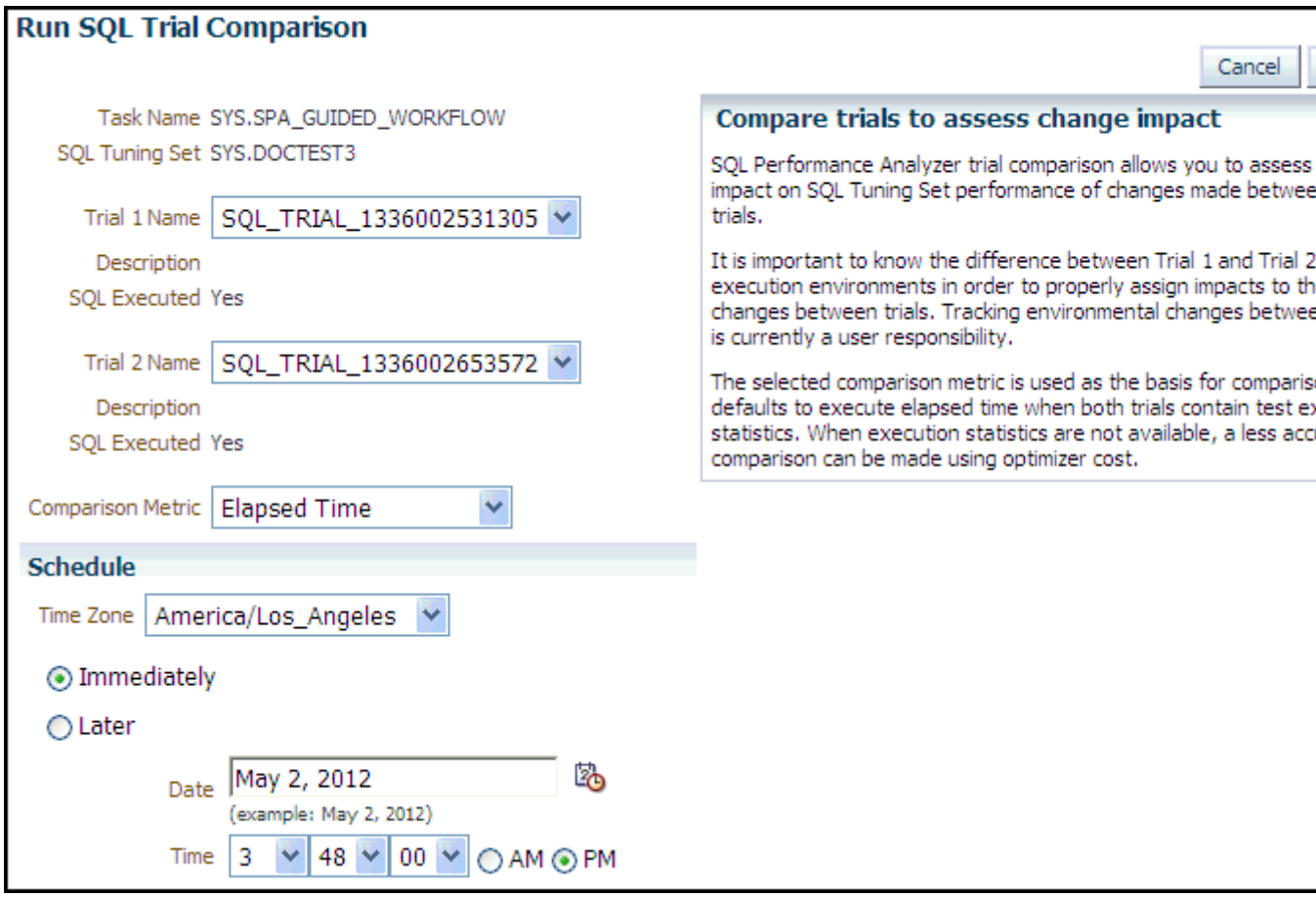
この項では、Oracle Enterprise Managerを使用してシステム変更の前後にSQLパフォーマンスを分析する方法について説明します。

SQL試行を比較する前に、変更後のSQL試行を作成する必要があります。詳細は、「変更後のSQL試行の作成」を参照してください。

Enterprise Managerを使用してSQLパフォーマンスを分析するには:

1. 「ガイド付きワークフロー」ページで、「ステップ2とステップ3を比較」の「実行」アイコンをクリックします。

   「SQL試行比較の実行」ページが表示されます。



   この例では、 SQL_TRIAL_1241213421833試行および SQL_TRIAL_1241213881923試行が比較用に選択されています。

2. デフォルトで示されている試行以外の試行を比較するには、「試行1の名前」リストおよび「試行2の名前」リストで目的の試行を選択します。

   統計試行は実行計画のみをテストする試行とは比較できないことに注意してください。

3. 「比較メトリック」リストで、比較分析に使用する比較メトリックを選択します。

   - 経過時間
   - CPU時間
   - ユーザーI/O時間

- バッファ読取り
- 物理I/O
- オプティマイザ・コスト
- I/Oインターコネクト・バイト

SQL試行で実行計画のみを生成した場合に選択可能な比較メトリックは「オプティマイザ・コスト」のみです。

複数の比較メトリックを使用して比較分析を実行するには、異なるメトリックを使用してこの手順を繰り返すことによって比較分析を別々に実行します。

4. 「スケジュール」セクションで、次の手順を実行します。
   - 「タイムゾーン」リストで、タイムゾーン・コードを選択します。
   - 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。
5. 「発行」をクリックします。

   比較分析が開始されると、「ガイド付きワークフロー」ページが表示されます。

   比較分析中、このステップのステータス・アイコンは矢印アイコンに変わります。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。変更前および変更後の実行から収集したパフォーマンス・データの量によっては、比較分析が完了するまで時間がかかる場合があります。比較分析が完了したら、「ステータス」アイコンがチェック・マークに変わり、次のステップの「実行」アイコンが有効になります。
6. SQLパフォーマンス・アナライザが変更前と変更後のパフォーマンス・データを分析した後、詳細な分析に使用できるSQLパフォーマンス・アナライザ・レポートを生成します。

   「ガイド付きワークフロー」ページで、「試行比較レポートの表示」の「実行」アイコンをクリックします。

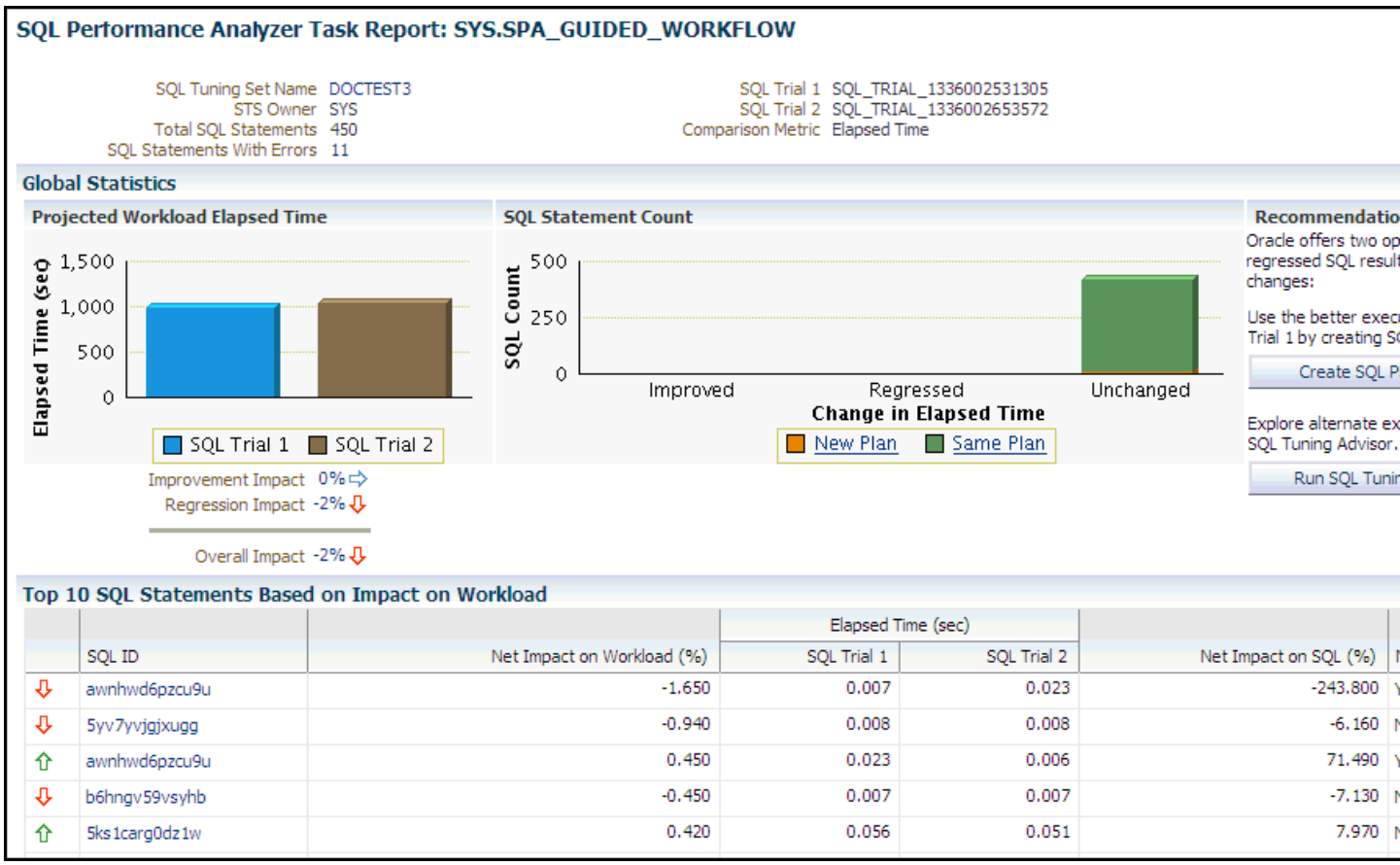
   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。レポートを確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認

SQLパフォーマンス・アナライザのタスクが完了すると、変更前と変更後のSQLパフォーマンスを比較するSQLパフォーマンス・アナライザ・レポートに結果のデータが生成されます。

図6-1に、SQLパフォーマンス・アナライザ・レポートのサンプルを示します。このサンプル・レポートでは、経過時間比較メトリックを使用して、変更前と変更後のSQLワークロードの実行を比較します。

図6-1 SQLパフォーマンス・アナライザ・レポート



SQLパフォーマンス・アナライザ・レポートを確認する前に、変更前と変更後のバージョンのパフォーマンス・データを比較します。詳細は、「Oracle Enterprise Managerを使用したSQL試行の比較」を参照してください

SQLパフォーマンス・アナライザ・レポートを生成および確認するには:

1. 「パフォーマンス」メニューから「SQL」を選択し、「SQLパフォーマンス・アナライザ」を選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。既存のSQLパフォーマンス・アナライザのリストが表示されます。

2. 「SQLパフォーマンス・アナライザのタスク」で、SQLパフォーマンス・アナライザ・レポートを表示するタスクを選択し、「最新レポートの表示」をクリックします。

   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。

3. パフォーマンス分析に関する一般情報を確認します。詳細は、「SQLパフォーマンス・アナライザ・レポートの確認: 一般情報」を参照してください。

4. 一般的な統計を確認します。詳細は、「SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計」を参照してください。

5. オプションで、詳細な統計を確認します。詳細は、「SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計の詳細」を参照してください。

6. アクティブ・レポートを生成するには、「保存」をクリックしてレポートを生成し保存するか、「メール」をクリックしてレポートを生成しHTML添付としてメールで送信します。

   アクティブ・レポートには、各カテゴリ(改善された計画、低下した計画および変更された計画)の上位SQL文と、変更

前と変更後の統計、実行計画およびタスク・サマリーに関する情報が含まれます。

詳細は、「SQLパフォーマンス・アナライザのアクティブ・レポートについて」を参照してください。

SQLパフォーマンス・アナライザ・レポートの確認: 一般情報

「一般情報」セクションには、SQLパフォーマンス・アナライザによって実行されたワークロードの比較に関する基本情報およびメタデータが含まれています。

一般情報を確認するには:

1. 「SQLパフォーマンス・アナライザのタスク・レポート: {0}」ページで、ページの上部にあるサマリーを確認します。

SQL Tuning Set Name	DOCTEST3	SQL Trial 1	INITIAL_SQL_TRIAL
STS Owner	SYS	SQL Trial 2	SECOND_SQL_TRIAL
Total SQL Statements	450	Comparison Metric	I/O Interconnect Bytes
SQL Statements With Errors	11		

   このサマリーには次の情報が含まれています。

   - SQLチューニング・セットの名前および所有者
   - チューニング・セットのSQL文の合計と、エラーが発生したSQL文の数、サポートされていないSQL文の数、またはタイムアウトになったSQL文の数
   - SQL試行および使用される比較メトリックの名前

2. オプションで、「SQLチューニング・セット名」の横にあるリンクをクリックします。

   「SQLチューニング・セット」ページが表示されます。

   このページには、SQLチューニング・セット内のすべてのSQL文に関する情報(SQL IDやSQLテキストなど)が含まれています。

3. エラーが検出された場合、「エラーのあるSQL文」の横にあるリンクをクリックします。

   エラー表には、特定のSQLワークロードの実行中に発生したすべてのエラーがレポートされます。エラーは、SQLチューニング・セット内のすべてのSQLの実行に共通する場合はSQLチューニング・セット・レベル、SQL文または実行計画に固有の場合は実行レベルでレポートされます。

4. グローバル統計を確認します。詳細は、「SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計」を参照してください。

SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計

「グローバル統計」セクションにより、SQLワークロードの全体的なパフォーマンスについての統計がレポートされます。このセクションによりSQLワークロードの全体的なパフォーマンスに対するシステム変更の影響をレポートされるため、SQLパフォーマンス・アナライザ分析の非常に重要な部分を占めます。このセクションの情報を使用して、ワークロード・パフォーマンスの傾向を理解し、システム変更によりワークロード・パフォーマンスがどのように影響を受けるかを判断します。

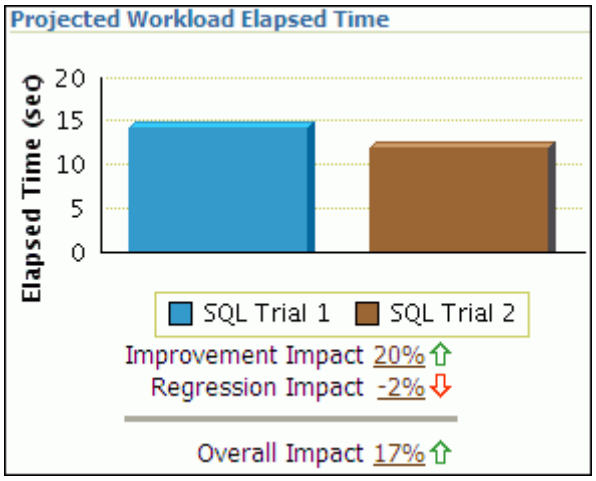
グローバル統計を確認するには:

1. 「予測されるワークロードの経過時間」サブセクション内のグラフを確認します。

ノート:

サブセクションの名前は、選択された比較メトリックによって異なります。

グラフでは、X軸に2つの試行、Y軸に経過時間(秒)が示されます。



最も重要な統計は全体的な影響で、パーセントで示されます。全体的な影響は、改善の影響と低下の影響とで異なります。「SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計の詳細」の説明に従って、あらゆる影響統計のリンクをクリックして詳細を確認できます。

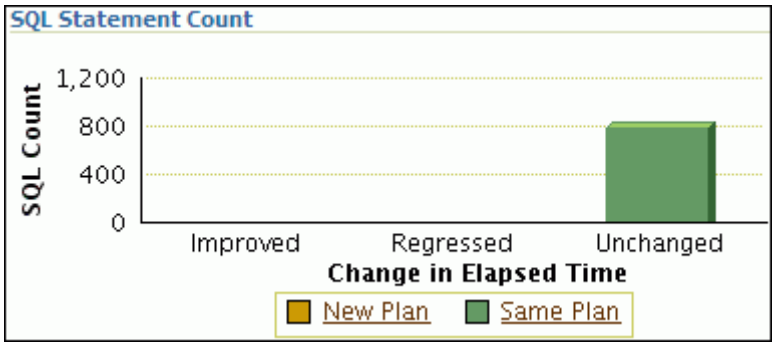
この例では、改善の影響が20%、低下の影響が-2%であるため、システム変更の全体的な影響は約18%の改善となります。つまり、この例のパフォーマンスの低下がすべて改善された場合、変更による全体の影響は、20%の改善ということになります。

ノート:

全体的な影響の割合は、改善の影響と低下の影響の合計と比較して 1%ほどずれることがあります。この相違は、数値の丸めによって起こるもの、あるいは SQL やワークロードの時間制限を推奨値である 1%に設定した場合に起こるものと考えられます。したがって、影響の少ない SQL 文を除外することにより、影響の大きい SQL 分を重点的に分析することができます。

2. 「SQL文の数」サブセクション内のグラフを確認します。

グラフのX軸には、システム変更後にパフォーマンスが改善されたSQL文、パフォーマンスが低下したSQL文またはパフォーマンスの変更がなかったSQL文の数が示されます。Y軸には、SQL文の数が示されます。また、グラフには、SQL文に対して実行計画が変更されたかどうかも示されます。

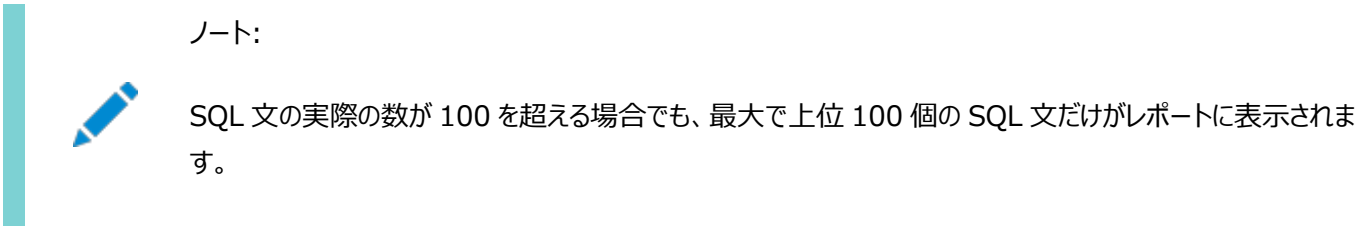


このグラフにより、SQL文の相対的なパフォーマンスを迅速に比較できます。「SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計の詳細」の説明に従って、グラフ内のいずれかのバーをクリックしてSQL文の詳細を確認できます。実際のSQL文の数が100を超えている場合でも、最大100個のSQL文のみ表示されます。

この例では、システム変更後にすべてのSQL文で変更がありません。

SQLパフォーマンス・アナライザ・レポートの確認: グローバル統計の詳細

SQLパフォーマンス・アナライザ・レポートを使用して、SQLワークロードの比較の詳細な統計を取得できます。詳細なグラフによって、レポートに表示されるSQL文のパフォーマンスにドリルダウンできます。このセクションの情報を使用して、特定のSQL文のパフォーマンスが低下した原因を調査します。



ノート:

SQL 文の実際の数が 100 を超える場合でも、最大で上位 100 個の SQL 文だけがレポートに表示されます。

グローバル統計の詳細を確認するには:

1. 「予測されるワークロードの経過時間」サブセクションで、詳細を確認するSQL文の影響のパーセントをクリックします。
   SQL文の詳細を確認するには、パフォーマンスの状態に応じて次の操作を実行します。
   - 改善された場合は、「改善の影響」のパーセントをクリックします。
   - 低下した場合は、「低下の影響」のパーセントをクリックします。
   - 改善されたか、または低下した場合は、「全体の影響」のパーセントをクリックします。

   詳細な統計を含む表が表示されます。選択したSQL文のタイプに応じて、次の列が含まれます。
   - SQL ID

     この列には、SQL文のIDが示されます。
   - 「ワークロードに対する最終的な影響(%)」

     この列には、SQLワークロードのパフォーマンスに対するシステム変更による影響が示されます。
   - 経過時間

     この列には、SQL文の実行にかかる合計時間(秒)が示されます。
   - SQLに対する最終的な影響(%)

     この列には、特定のSQL文のパフォーマンスに対する変更による局所的な影響が示されます。
   - 新規計画

     この列には、SQLの実行計画が変更されたかどうかが示されます。
2. 特定のSQL文の詳細を表示するには、対象のSQL文の「SQL ID」リンクをクリックします。

   「SQLの詳細」ページが表示されます。

   このページを使用して、SQLテキストにアクセスし、SQL文に関する低レベルの詳細(実行統計、実行計画など)を取得できます。

SQLパフォーマンス・アナライザのアクティブ・レポートについて

SQLパフォーマンス・アナライザのアクティブ・レポートは、ウェブホスト型のインタラクティブ・ユーザー・インタフェースを使用してレポート対象のすべてのデータを表示するHTMLファイルです。アクティブ・レポートには、Oracle Enterprise Managerで利用

可能なSQLパフォーマンス・アナライザ・レポートと同様に、各カテゴリ(改善された計画、低下した計画および変更された計画)の上位SQL文と、変更前と変更後の統計、実行計画およびタスク・サマリーに関する情報が含まれます。

SQLパフォーマンス・アナライザのアクティブ・レポートは、従来のHTMLレポートやテキスト・レポートよりも便利で、Oracle Enterprise Managerと似たユーザー・インタフェースを備えながら、データベースを使用できないときやデータベースを削除した後でも参照できます。そのため、アクティブ・レポートは、従来のレポートと動的なOracle Enterprise Manager分析の利点を生かすと同時に、両者の短所を解消しています。また、アクティブ・レポートでは比較分析に関する詳しい情報が提供され、より多くのインタラクティブなユーザー・オプションを使用できます。HTMLレポートやテキスト・レポートのかわりにアクティブ・レポートを使用することを強くお薦めします。

アクティブ・レポートのユーザー・インタフェース・コンポーネントは、Oracle Enterprise Managerに表示されるコンポーネントとよく似ています。ユーザー・インタフェース・コンポーネントの説明は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」の関連する項を参照してください。

Oracle Enterprise Managerを使用した、パフォーマンスが低下したSQL文のチューニング

SQLパフォーマンス・アナライザ・レポートを確認したら、SQLパフォーマンスの比較後に特定されるパフォーマンスが低下したSQL文をチューニングする必要があります。パフォーマンスが低下したSQL文が多数表示される場合は、根本原因を特定し、システム・レベルの変更を行って問題を修正する必要があります。パフォーマンスが低下したSQL文が少数の場合は、次のいずれかのチューニング方法を使用して、部分的な解決を行うことを検討してください。

- SQL計画ベースラインの作成
- SQLチューニング・アドバイザの実行

パフォーマンスが低下したSQL文をチューニングしたら、SQLパフォーマンス・アナライザを使用して、それらの変更をテストする必要があります。新しいSQL試行をテスト・システムで実行すると、2番目の比較(新しいSQL試行と最初の試行の比較)が実行され、結果が検証されます。SQLパフォーマンス・アナライザにパフォーマンスが安定していることが示されたら、テストは完了します。このステップでの修正を本番システムに実装してください。

Oracle Database 11g リリース1以上では、SQL文のチューニング時に、SQLチューニング・アドバイザによって代替計画が分析されます。SQLチューニング・アドバイザは、現在のシステムで以前の実行計画(最初のSQL試行の計画も含む)を検索します。最初のSQL試行の実行計画が2番目のSQL試行の実行計画と異なる場合、SQLチューニング・アドバイザは、最初のSQL試行の計画を推奨します。これらの実行計画のパフォーマンスがよい場合は、最初のSQL試行の計画を使用して計画ベースラインを作成できます。

> ノート:
>
> SQL パフォーマンス・アナライザでは、リモート SQL 試行の完了直後に、SQL 計画ベースラインを作成したり、SQL チューニング・アドバイザを実行するオプションを提供しません。このような場合、API を使用して、SQL チューニング・セットを手動で転送し、リモート・データベースで適切な手順を完了する必要があります。

関連項目:

- 代替計画の分析の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください

SQL計画ベースラインの作成

SQL計画ベースラインを作成すれば、既知の性能特性に基づいて実行計画を使用することによって、オプティマイザのパフォーマンス低下を避けることができます。計画を変更したためにパフォーマンスの低下をきたした場合は、SQL計画ベースラインを作成してこれを使用し、新たにパフォーマンスが低下した実行計画をオプティマイザが選択するのを防ぐことができます。

SQL計画ベースラインを作成するには:

1. 「SQLパフォーマンス・アナライザのタスク結果」ページの「推奨」で、「SQL計画ベースラインの作成」をクリックします。

   「SQL計画ベースラインの作成」ページが表示されます。「低下したSQL文」セクションに、新しいSQL計画ベースラインに関連付けるパフォーマンスが低下したSQL文が示されます。

2. 「ジョブ・パラメータ」で、次のようにジョブのパラメータを指定します。

   a. 「ジョブ名」フィールドに、ジョブの名前を入力します。

   b. 「説明」フィールドに、オプションでジョブの説明を入力します。

3. 「スケジュール」で、次の項目を選択します。

   a. 即時: ジョブを即時開始します。

   b. 後で: 「タイムゾーン」、「日付」および「時間」フィールドを使用して指定した時間に開始するようにジョブをスケジュールします。

4. 「OK」をクリックします。

   「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。ジョブが正常に実行されたことを示すメッセージが表示されます。

関連項目:

- SQL計画ベースラインの作成および管理の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

SQLチューニング・アドバイザの実行

SQLチューニング・アドバイザは、パフォーマンスが低下したSQL文を詳細に分析し、問題の根本的な原因を解決しようとします。

SQLチューニング・アドバイザを実行するには:

1. 「SQLパフォーマンス・アナライザのタスク結果」ページの「推奨」で、「SQLチューニング・アドバイザの実行」をクリックします。

   「SQLチューニング・タスクのスケジュール」ページが表示されます。

2. 「チューニング・タスク名」フィールドに、SQLチューニング・タスクの名前を入力します。
3. 「チューニング・タスクの説明」フィールドに、オプションでSQLチューニング・タスクの名前を入力します。
4. 「スケジュール」で、次の項目を選択します。
   - 即時: ジョブを即時開始します。

- 後で：「タイムゾーン」、「日付」および「時間」フィールドを使用して指定した時間に開始するようにジョブをスケジュールします。

5. 「OK」をクリックします。

   「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。SQLチューニング・レポートへのリンクが「推奨」の下に表示されます。

6. SQLチューニング・レポートを表示するには、「SQLチューニング・レポート」リンクをクリックします。

   「SQLチューニング結果」ページが表示されます。

関連項目:

- SQLチューニング・アドバイザの実行の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

APIを使用したSQL試行の比較

APIを使用してSQL試行を比較するには、次のステップを実行します。

- APIを使用したSQLパフォーマンスの分析
- コマンドラインを使用したSQLパフォーマンス・アナライザ・レポートの確認
- APIを使用したSQLチューニング・セットの比較
- APIを使用した、パフォーマンスが低下したSQL文のチューニング
- APIを使用した、リモートSQL試行からのパフォーマンスが低下したSQL文のチューニング
- APIを使用したSQL計画ベースラインの作成
- SQLパフォーマンス・アナライザのビューの使用

SQL試行を比較する前に、変更後のSQL試行を作成する必要があります。詳細は、「変更後のSQL試行の作成」を参照してください。

APIを使用したSQLパフォーマンスの分析

変更後のSQLのパフォーマンス・データが構築されたら、変更前バージョンと変更後バージョンのパフォーマンス・データを比較できます。DBMS_SQLPA.EXECUTE_ANALYSIS_TASKプロシージャまたはファンクションを使用して、比較分析を実行します。

変更前と変更後のSQLパフォーマンス・データを比較するには:

1. 次のパラメータを使用してEXECUTE_ANALYSIS_TASKプロシージャまたはファンクションをコールします。
   - task_nameパラメータをSQLパフォーマンス・アナライザのタスクの名前に設定します。
   - execution_typeパラメータをCOMPARE PERFORMANCEに設定します。この設定によって、2つのバージョンのSQLパフォーマンス・データが分析および比較されます。
   - execution_nameパラメータを使用して、実行を識別するための名前を指定します。指定しない場合は、SQLパフォーマンス・アナライザによって名前が生成され、ファンクションによって戻されます。

- execution_paramsパラメータを使用して、2つのバージョンのSQLパフォーマンス・データを指定します。execution_paramsパラメータは、指定した実行の名前/値ペアとして指定されます。SQLパフォーマンス・データの比較および分析に関連する実行パラメータを次のように設定します。
  - execution_name1パラメータを最初の実行(システム変更が実行される前)の名前に設定します。この値は、「APIを使用した変更前のSQL試行の作成」で指定されているexecution_nameパラメータの値に対応している必要があります。
  - execution_name2パラメータを2番目の実行(システム変更が実行された後)の名前に設定します。システム変更後にSQLワークロードを実行した場合、この値は、「APIを使用した変更後のSQL試行の作成」で指定されているexecution_nameパラメータの値に対応している必要があります。コール元が実行を指定しなかった場合、SQLパフォーマンス・アナライザではデフォルトで最後の2回のタスク実行が常に比較されます。
  - comparison_metricパラメータを設定して、パフォーマンスに対する影響の分析に使用する実行統計の式を指定します。指定可能な値は、elapsed_time(デフォルト)、cpu_time、buffer_gets、disk_reads、direct_writes、optimizer_costおよびio_interconnect_bytesの各メトリックまたはこれらの組合せです。

  比較用に設定可能なその他のパラメータについては、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_SQLPAパッケージに関する項を参照してください。

次の例は、ファンクション・コールを示しています。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'COMPARE PERFORMANCE', -
       execution_name => 'my_exec_compare', -
       execution_params => dbms_advisor.arglist(-
                           'comparison_metric', 'buffer_gets'));
```

2. 次のパラメータを使用してREPORT_ANALYSIS_TASKファンクションをコールします。
   - task_nameパラメータをSQLパフォーマンス・アナライザのタスクの名前に設定します。
   - execution_nameパラメータを、使用する実行の名前に設定します。この値は、レポートの生成対象となる実行のexecution_nameパラメータと一致している必要があります。

     レポートを生成して結果を表示するには:
     - SQLワークロードに対して生成された実行計画の場合は、目的のEXPLAIN PLAN実行のexecution_nameパラメータと一致するようにこの値を設定します。
     - SQLワークロードに対して生成された実行計画および実行統計の場合は、目的のTEST EXECUTE実行で使用されたexecution_nameパラメータの値と一致するようにこのパラメータを設定します。
     - 比較分析の場合は、目的のANALYZE PERFORMANCE実行のexecution_nameパラメータと一致するようにこの値を設定します。

     指定しなかった場合、SQLパフォーマンス・アナライザによって最後の実行に関するレポートが生成されます。
   - typeパラメータを設定して、生成するレポートのタイプを指定します。指定可能な値は、TEXT(デフォルト)、HTML、XMLおよびACTIVEです。

     アクティブ・レポートには、インタラクティブなユーザー・インタフェースを使用するきめ細かなレポート機能が用意されており、データベースやOracle Enterprise Managerに接続されていないときでも詳細な分析を実行

できます。可能なかぎり、HTMLレポートやテキスト・レポートのかわりにアクティブ・レポートを使用することをお薦めします。

アクティブ・レポートの詳細は、「SQLパフォーマンス・アナライザのアクティブ・レポートについて」を参照してください。

- levelパラメータを設定して、推奨事項の形式を指定します。指定可能な値は、TYPICAL(デフォルト)、ALL、BASIC、CHANGED、CHANGED_PLANS、ERRORS、IMPROVED、REGRESSED、TIMEOUT、UNCHANGED、UNCHANGED_PLANSおよびUNSUPPORTEDです。
- sectionパラメータを設定して、レポートに生成する特定のセクションを指定します。指定可能な値は、SUMMARY(デフォルト)およびALLです。
- top_sqlパラメータを設定して、レポートに生成するSQLチューニング・セット内のSQL文の数を指定します。デフォルトでは、システム変更によって影響を受ける上位100のSQL文がレポートに示されます。

アクティブ・レポートを生成するには、次のスクリプトを実行します。

```
set trimspool on
set trim on
set pages 0
set linesize 1000
set long 1000000
set longchunksize 1000000
spool spa_active.html
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(task_name => 'my_spa_task',
          type => 'active', section => 'all') FROM dual;
spool off
```

次に、比較サマリー・レポートをテキスト形式で作成し表示するためのSQLスクリプトの例の一部を示します。

```
VAR rep   CLOB;
EXEC :rep := DBMS_SQLPA.REPORT_ANALYSIS_TASK('my_spa_task', -
                'text', 'typical', 'summary');
SET LONG 100000 LONGCHUNKSIZE 100000 LINESIZE 130
PRINT :rep
```

3. 「コマンドラインを使用したSQLパフォーマンス・アナライザ・レポートの確認」で説明されているSQLパフォーマンス・アナライザ・レポートを確認してください。

関連項目:

- DBMS_SQLPA.EXECUTE_ANALYSIS_TASKおよびDBMS_SQLPA.REPORT_ANALYSIS_TASKファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

コマンドラインを使用したSQLパフォーマンス・アナライザ・レポートの確認

SQLパフォーマンス・アナライザ・レポートは、次のセクションに分類されます。

- 一般情報
- 結果のサマリー
- 結果の詳細

この項では、サンプル・レポートを使用して、SQLパフォーマンス・アナライザ・レポートを確認する方法について説明します。このサンプル・レポートでは、buffer_getsを比較メトリックとして使用して、変更前と変更後のSQLワークロードの実行を比較します。

一般情報

一般情報セクションには、SQLパフォーマンス・アナライザのタスク、使用されたSQLチューニング・セット、変更前と変更後の実行に関する基本情報およびメタデータが含まれています。例6-1に、サンプル・レポートの一般情報セクションを示します。

例6-1では、タスク名がmy_spa_taskであることがタスク情報セクションに示されています。ワークロード情報セクションには、101のSQL文を含むSQLチューニング・セットmy_stsの実行がタスクによって比較されたことが示されています。実行情報セクションには、比較実行名がmy_exec_compareであることが示されています。

分析情報セクションには、SQLパフォーマンス・アナライザによって、比較メトリックのbuffer_getsを使用してSQLチューニング・セットmy_stsの2つの実行(my_exec_BEFORE_changeとmy_exec_AFTER_change)が比較されたことが示されています。

例6-1 一般情報

```
--------------------------------------------------------------------------------------------------
General Information
--------------------------------------------------------------------------------------------------
 Task Information:                                Workload Information:
 -----------------------------------------------  -----------------------------------------------
  Task Name     : my_spa_task                      SQL Tuning Set Name         : my_sts
  Task Owner    : APPS                             SQL Tuning Set Owner        : APPS
  Description   :                                  Total SQL Statement Count   : 101
Execution Information:
--------------------------------------------------------------------------------------------------
  Execution Name   : my_exec_compare        Started              : 05/21/2007 11:30:09
  Execution Type   : ANALYZE PERFORMANCE    Last Updated         : 05/21/2007 11:30:10
  Description      :                        Global Time Limit    : UNLIMITED
  Scope            : COMPREHENSIVE          Per-SQL Time Limit   : UNUSED
  Status           : COMPLETED              Number of Errors     : 0
Analysis Information:
--------------------------------------------------------------------------------------------------
 Comparison Metric: BUFFER_GETS
 ------------------
 Workload Impact Threshold: 1%
 --------------------------
 SQL Impact Threshold: 1%
 ----------------------
 Before Change Execution:                         After Change Execution:
 -----------------------------------------------  -----------------------------------------------
  Execution Name       : my_exec_BEFORE_change     Execution Name       : my_exec_AFTER_change
  Execution Type       : TEST EXECUTE              Execution Type       : TEST EXECUTE
  Description          :                           Description          :
  Degree of Parallelism: 4                         Degree of Parallelism: 4
  Scope                : COMPREHENSIVE             Scope                : COMPREHENSIVE
  Status               : COMPLETED                 Status               : COMPLETED
  Started              : 05/21/2007 11:22:06       Started              : 05/21/2007 11:25:56
  Last Updated         : 05/21/2007 11:24:01       Last Updated         : 05/21/2007 11:28:30
  Global Time Limit    : 1800                      Global Time Limit    : 1800
  Per-SQL Time Limit   : UNUSED                    Per-SQL Time Limit   : UNUSED
  Number of Errors     : 0                         Number of Errors     : 0
--------------------------------------------------------------------------------------------------
```

結果のサマリー

結果のサマリー・セクションには、SQLパフォーマンス・アナライザのタスクの結果の概要が示されます。結果のサマリー・セクションは、次のサブセクションに分類されます。

- 全体的なパフォーマンス統計
- SQL文のパフォーマンス統計
- エラー

全体的なパフォーマンス統計

全体的なパフォーマンス統計のサブセクションには、SQLワークロードの全体的なパフォーマンスについての統計が表示されます。このセクションによりSQLワークロードの全体的なパフォーマンスに対するシステム変更の影響が示されるため、SQLパフォーマンス・アナライザ分析の非常に重要な部分を占めます。このセクションの情報を使用して、ワークロード・パフォーマンスの変化を理解し、システム変更後にワークロード・パフォーマンスが向上するか低下するかを判断します。

例6-2に、サンプル・レポートの全体的なパフォーマンス統計サブセクションを示します。

この例では、低下の影響が-10.08%であったにもかかわらず、SQLワークロードの全体的なパフォーマンスは47.94%改善されたことを示しています。つまり、この例のパフォーマンスの低下がすべて改善された場合、変更による全体の影響は、58.02%ということになります。システム変更後、101のSQL文のうち2つは実行速度が速くなりましたが、1つは遅くなりました。98の文のパフォーマンスには変化がありませんでした。

例6-2 全体的なパフォーマンス統計

```
Report Summary
------------------------------------------------------------------------------------------------

Projected Workload Change Impact:
-----------------------------------------
 Overall Impact      :   47.94%
 Improvement Impact  :   58.02%
 Regression Impact   :  -10.08%

SQL Statement Count
-----------------------------------------
 SQL Category  SQL Count  Plan Change Count
 Overall       101        6
 Improved      2          2
 Regressed     1          1
 Unchanged     98         3
.
.
.
------------------------------------------------------------------------------------------------
```

SQL文のパフォーマンス統計

パフォーマンス統計サブセクションでは、システム変更によって最も影響を受けるSQL文が強調表示されます。ワークロード内のSQL文ごとに、次の条件に基づいて変更前と変更後のパフォーマンス・データが比較されます。

- 各SQL文の実行頻度(重要性)
- SQLワークロード全体に対する各SQL文へのシステム変更の影響

- 各SQL文へのシステム変更の影響
- 各SQL文の実行計画の構造が変更されたかどうか

例6-3に、サンプル・レポートのSQL文のパフォーマンス統計サブセクションを示します。レポートは、ページに収まるように少し変更してあります。

これらのSQL文は、SQLワークロードに対する最終的な影響の絶対値で降順にソートされています。つまり、ソート順序は、影響の良し悪しとは関係がありません。

例6-3 SQL文のパフォーマンス統計

```
SQL Statements Sorted by their Absolute Value of Change Impact on the Workload
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
|           |               | Impact on | Execution |  Metric  |  Metric  | Impact | Plan   |
| object_id | sql_id        | Workload  | Frequency |  Before  |  After   | on SQL | Change |
-------------------------------------------------------------------------------------------------
|    205    | 73s2sgy2svfrw |   29.01%  |   100000  |  1681683 |  220590  | 86.88% |   y    |
|    206    | gq2a407mv2hsy |   29.01%  |   949141  |  1681683 |  220590  | 86.88% |   y    |
|    204    | 2wtgxbjz6u2by |  -10.08%  |   478254  |  1653012 |  2160529 | -30.7% |   y    |
-------------------------------------------------------------------------------------------------
```

エラー

エラー・サブセクションには、実行中に発生したすべてのエラーがレポートされます。エラーは、SQLチューニング・セット内のすべての実行に共通する場合はSQLチューニング・セット・レベル、SQL文または実行計画に固有の場合は実行レベルでレポートされます。

例6-4に、SQLパフォーマンス・アナライザ・レポートのエラー・サブセクションの例を示します。

例6-4 エラー

```
--------------------------------------------------------------------------------
                          SQL STATEMENTS WITH ERRORS
--------------------------------------------------------------------------------
SQL ID        Error
------------- ------------------------------------------------------------------
47bjmcdtw6htn ORA-00942: table or view does not exist
br61bjp4tnf7y ORA-00920: invalid relational operator
--------------------------------------------------------------------------------
```

結果の詳細

結果の詳細セクションでは、レポートの結果のサマリー・セクションに表示されるSQL文のパフォーマンスまでドリルダウンできます。このセクションの情報を使用して、特定のSQL文のパフォーマンスが低下した原因を調査します。

このセクションには、SQLパフォーマンスの影響分析で処理されたすべてのSQL文のエントリが示されます。各エントリは、次のサブセクションで構成されています。

- SQLの詳細
- 実行統計
- 実行計画

SQLの詳細

レポートのこのセクションには、SQL文の概要(SQL文の情報および実行の詳細)が表示されます。

例6-5に、サンプル・レポートのSQLの詳細サブセクションを示します。

例6-5には、パフォーマンスが低下したSQL文(IDは2wtgxbjz6u2by、対応するオブジェクトIDは204)の概要が示されています。

例6-5 SQLの詳細

```
SQL Details:
-----------------------------
 Object ID             : 204
 Schema Name           : APPS
 SQL ID                : 2wtgxbjz6u2by
 Execution Frequency   : 1
 SQL Text              : SELECT /* my_query_14_scott */ /*+ ORDERED INDEX(t1)
                         USE_HASH(t1) */ 'B' || t2.pg_featurevalue_05_id
                         pg_featurevalue_05_id, 'r' || t4.elementrange_id
                         pg_featurevalue_15_id, 'G' || t5.elementgroup_id
                         pg_featurevalue_01_id, 'r' || t6.elementrange_id . . .
.
.
.
-------------------------------------------------------------------------------------------------
```

実行統計

実行統計サブセクションでは、変更前と変更後の実行からのSQL文の実行統計が比較され、検出結果の概要が示されます。

例6-6に、サンプル・レポートの実行統計サブセクションを示します。

例6-6 実行統計

```
Execution Statistics:
-----------------------------
-------------------------------------------------------------------------------------------------
|              | Impact on | Value    | Value    | Impact      | % Workload | % Workload |
| Stat Name    | Workload  | Before   | After    | on SQL      | Before     | After      |
-------------------------------------------------------------------------------------------------
| elapsed_time |    -95.54%|  36.484  | 143.161  |   -292.39%  |     32.68% |     94.73% |
| parse_time   |    -12.37%|    .004  |    .062  |     -1450%  |       .85% |     11.79% |
| exec_elapsed |    -95.89%|   36.48  | 143.099  |   -292.27%  |     32.81% |     95.02% |
| exec_cpu     |    -19.73%|  36.467  |  58.345  |    -59.99%  |     32.89% |     88.58% |
| buffer_gets  |    -10.08%| 1653012  | 2160529  |     -30.7%  |     32.82% |     82.48% |
| cost         |     12.17%|   11224  |    2771  |     75.31%  |     16.16% |      4.66% |
| reads        |  -1825.72%|    4091  |  455280  | -11028.82%  |     16.55% |     96.66% |
| writes       |     -1500%|       0  |      15  |     -1500%  |         0% |       100% |
| rows         |           |     135  |     135  |             |            |            |
-------------------------------------------------------------------------------------------------
Notes:
-----------------------------
Before Change:
1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.
After Change:
1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.
```

```
Findings (2):
-----------------------------
1. The performance of this SQL has regressed.
2. The structure of the SQL execution plan has changed.
--------------------------------------------------------------------------------------------------
```

実行計画

実行計画サブセクションには、SQL文の変更前および変更後の実行計画が表示されます。パフォーマンスが低下した場合、このセクションには、根本原因および兆候についての検出結果も表示されます。

例6-7に、サンプル・レポートの実行計画サブセクションを示します。

例6-7 実行計画

```
Execution Plan Before Change:
-----------------------------
 Plan Id          : 1
 Plan Hash Value  : 3412943215

-------------------------------------------------------------------------------------------------------
----
| Id   | Operation                          | Name                | Rows   | Bytes    | Cost  | Time
|
-------------------------------------------------------------------------------------------------------
----
|    0 | SELECT STATEMENT                   |                     |      1 |      126 | 11224 |
00:02:15 |
|    1 |   HASH GROUP BY                    |                     |      1 |      126 | 11224 |
00:02:15 |
|    2 |    NESTED LOOPS                    |                     |      1 |      126 | 11223 |
00:02:15 |
|  * 3 |     HASH JOIN                      |                     |      1 |      111 | 11175 |
00:02:15 |
|  * 4 |      TABLE ACCESS FULL             | LU_ELEMENTGROUP_REL |      1 |       11 |   162 |
00:00:02 |
|  * 5 |      HASH JOIN                     |                     |    487 |    48700 | 11012 |
00:02:13 |
|    6 |       MERGE JOIN                   |                     |     14 |      924 |  1068 |
00:00:13 |
|    7 |        SORT JOIN                   |                     |   5391 |   274941 |  1033 |
00:00:13 |
|  * 8 |         HASH JOIN                  |                     |   5391 |   274941 |   904 |
00:00:11 |
|  * 9 |          TABLE ACCESS FULL         | LU_ELEMENTGROUP_REL |    123 |     1353 |   175 |
00:00:03 |
| * 10 |          HASH JOIN                 |                     |   5352 |   214080 |   729 |
00:00:09 |
| * 11 |           TABLE ACCESS FULL        | LU_ITEM_293         |   5355 |   128520 |    56 |
00:00:01 |
| * 12 |           TABLE ACCESS FULL        | ADM_PG_FEATUREVALUE |   1629 |    26064 |   649 |
00:00:08 |
| * 13 |        FILTER                      |                     |        |          |       |
|
| * 14 |         SORT JOIN                  |                     |      1 |       15 |    36 |
00:00:01 |
```

```
| * 15 |          TABLE ACCESS FULL          | LU_ELEMENTRANGE_REL |      1 |      15 |    35 |
00:00:01 |
|   16 |       INLIST ITERATOR               |                     |        |         |       |
|
| * 17 |        TABLE ACCESS BY INDEX ROWID  | FACT_PD_OUT_ITM_293 | 191837 | 6522458 |  9927 |
00:02:00 |
|   18 |         BITMAP CONVERSION TO ROWIDS |                     |        |         |       |
|
| * 19 |          BITMAP INDEX SINGLE VALUE  | FACT_274_PER_IDX    |        |         |       |
|
| * 20 |     TABLE ACCESS FULL               | LU_ELEMENTRANGE_REL |      1 |      15 |    49 |
00:00:01 |
-------------------------------------------------------------------------------------------------------
----

.
.
.
Execution Plan After Change:
----------------------------
 Plan Id          : 102
 Plan Hash Value  : 1923145679


-------------------------------------------------------------------------------------------------------
| Id   | Operation                           | Name                | Rows | Bytes  | Cost | Time     |
-------------------------------------------------------------------------------------------------------
|    0 | SELECT STATEMENT                    |                     |    1 |    126 | 2771 | 00:00:34 |
|    1 |   HASH GROUP BY                     |                     |    1 |    126 | 2771 | 00:00:34 |
|    2 |    NESTED LOOPS                     |                     |    1 |    126 | 2770 | 00:00:34 |
|  * 3 |     HASH JOIN                       |                     |    1 |    111 | 2722 | 00:00:33 |
|  * 4 |      HASH JOIN                      |                     |    1 |    100 | 2547 | 00:00:31 |
|  * 5 |       TABLE ACCESS FULL             | LU_ELEMENTGROUP_REL |    1 |     11 |  162 | 00:00:02 |
|    6 |       NESTED LOOPS                  |                     |      |        |      |          |
|    7 |        NESTED LOOPS                 |                     |  484 |  43076 | 2384 | 00:00:29 |
|  * 8 |         HASH JOIN                   |                     |   14 |    770 |  741 | 00:00:09 |
|    9 |          NESTED LOOPS               |                     |    4 |    124 |  683 | 00:00:09 |
| * 10 |           TABLE ACCESS FULL         | LU_ELEMENTRANGE_REL |    1 |     15 |   35 | 00:00:01 |
| * 11 |           TABLE ACCESS FULL         | ADM_PG_FEATUREVALUE |    4 |     64 |  649 | 00:00:08 |
| * 12 |          TABLE ACCESS FULL          | LU_ITEM_293         | 5355 | 128520 |   56 | 00:00:01 |
|   13 |         BITMAP CONVERSION TO ROWIDS |                     |      |        |      |          |
| * 14 |          BITMAP INDEX SINGLE VALUE  | FACT_274_ITEM_IDX   |      |        |      |          |
| * 15 |        TABLE ACCESS BY INDEX ROWID  | FACT_PD_OUT_ITM_293 |   36 |   1224 | 2384 | 00:00:29 |
| * 16 |      TABLE ACCESS FULL              | LU_ELEMENTGROUP_REL |  123 |   1353 |  175 | 00:00:03 |
| * 17 |     TABLE ACCESS FULL               | LU_ELEMENTRANGE_REL |    1 |     15 |   49 | 00:00:01 |
-------------------------------------------------------------------------------------------------------
```

APIを使用したSQLチューニング・セットの比較

DBMS_SQLPAパッケージを使用すると、2つのSQLチューニング・セットを比較できます。たとえば、データベース・リプレイの使用時に、本番システムでワークロードの取得中に1つのSQLチューニング・セットを取得し、テスト・システムでワークロードのリプレイ中にもう1つ別のSQLチューニング・セットを取得することがあります。その後、SQL文を再実行しなくても、SQLパフォーマンス・アナライザを使用してそれらのSQLチューニング・セットを比較できます。これは、システム変更の前後にワークロードを実行する別のユーティリティ(カスタム・スクリプトなど)がすでに存在する場合に便利です。

SQLチューニング・セットを比較する場合、SQLパフォーマンス・アナライザはSQLチューニング・セットで取得した実行時統計を

使用して比較分析を行い、一方のSQLチューニング・セットに存在し、もう一方のチューニング・セットには存在しない新しいSQL文や欠落しているSQL文をレポートします。2つのSQLチューニング・セット間の実行計画における変更もレポートされます。両方のSQLチューニング・セットのSQL文ごとに、検出されたパフォーマンスの改善と低下が、SQL文単位(実行ごとの平均統計値に基づいて計算)とワークロード全体(累積統計値に基づいて計算)でレポートされます。

APIを使用してSQLチューニング・セットを比較するには:

1. SQLパフォーマンス・アナライザ・タスクを作成するには、次の手順を実行します。

```
VAR aname varchar2(30);
EXEC :aname := 'compare_s2s';
EXEC :aname := DBMS_SQLPA.CREATE_ANALYSIS_TASK(task_name => :aname);
```

   作成時にSQLチューニング・セットをタスクに関連付ける必要はありません。

2. 最初のSQL試行を作成し、最初のSQLチューニング・セットを変換するには、次の手順を実行します。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'convert sqlset', -
          execution_name => 'first trial', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'sqlset_name', 'my_first_sts', -
                              'sqlset_owner', 'APPS'));
```

   SQLSET_NAMEおよびSQLSET_OWNERタスク・パラメータを使用して、SQLチューニング・セットの名前と所有者を指定します。SQLパフォーマンス・アナライザ・タスクでSQLチューニング・セットの内容が複製されることはありません。かわりに、SQLチューニング・セットへの参照が、新しいSQL試行(この例では「first trial」)との関連付けに記録されます。

3. 2番目のSQL試行を作成し、比較する2番目のSQLチューニング・セットと関連付けます。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'convert sqlset', -
          execution_name => 'second trial', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'sqlset_name', 'my_second_sts', -
                              'sqlset_owner', 'APPS'));
```

4. 比較分析を実行して、2つのSQL試行(SQLチューニング・セット)のパフォーマンス・データを比較します。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'compare', -
          execution_name => 'comparison', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'workload_impact_threshold', 0, -
                              'sql_impact_threshold', 0));
```

   この例では、ワークロードとSQLごとの影響しきい値は、比較のために0%に設定されています(デフォルト値は1%)。

5. 比較分析が完了した後、DBMS_SQLPA.REPORT_ANALYSIS_TASKファンクションを使用して、SQLパフォーマンス・アナライザ・レポートを生成します。

   APIを使用したSQLパフォーマンス・アナライザ・レポートの生成については、「APIを使用したSQLパフォーマンスの分析」を参照してください。

レポートを作成した後、そのレポートを確認して、2つのSQLチューニング・セットの内容の違いを識別します。例6-8に、2つのSQLチューニング・セットを比較して生成されたサンプル・レポートの分析情報セクションとレポート・サマリー・セクションを示します。

例6-8 分析情報とレポート・サマリー

```
Analysis Information:
-------------------------------------------------------------------------------------------------
 Before Change Execution:                          After Change Execution:
 ------------------------                          -----------------------
  Execution Name      : first trial                 Execution Name      : second trial
  Execution Type      : CONVERT SQLSET              Execution Type      : CONVERT SQLSET
  Status              : COMPLETED                   Status              : COMPLETED
  Started             : ...
  Last Updated        : ...

 Before Change Workload:                           After Change Workload:
 -----------------------                           ----------------------
  SQL Tuning Set Name        : my_first_sts         SQL Tuning Set Name        : my_second_sts
  SQL Tuning Set Owner       : APPS                 SQL Tuning Set Owner       : APPS
  Total SQL Statement Count  : 5                    Total SQL Statement Count  : 6

-------------------------------------------------------------------------------------------------
Report Summary
-------------------------------------------------------------------------------------------------

Projected Workload Change Impact:
---------------------------------------
 Overall Impact      :  72.32%
 Improvement Impact  :  47.72%
 Regression Impact   :   -.02%
 Missing-SQL Impact  :   33.1%
 New-SQL Impact      :  -8.48%

SQL Statement Count
---------------------------------------
 SQL Category   SQL Count  Plan Change Count
 Overall                7                  1
 Common                 4                  1
  Improved              3                  1
  Regressed             1                  0
 Different              3                  0
  Missing SQL           1                  0
  New SQL               2                  0
```

例6-8に示すように、このレポートには、標準のSQLパフォーマンス・アナライザ・レポートにはない、2つの追加カテゴリがあります。この2つのカテゴリは「Different」という見出しの下にまとめられています。

- 欠落したSQL

  このカテゴリは、最初のSQLチューニング・セットに存在し、2番目のSQLチューニング・セットに存在しすべてのSQL文を示します。この例では、SQL文が1つだけ欠落しています。例6-9に示すように、このSQL文は次のような内容になっています。

  - sql_id値: gv7xb8tyd1v91
  - 変更ごとのワークロードのパフォーマンスへの影響: 33.1%
  - 変更ごとのSQL文のパフォーマンスへの影響: なし(「Total Metric After」の変更値が欠落している)

- 新規SQL

このカテゴリは、2番目のSQLチューニング・セットに存在し、最初のSQLチューニング・セットに存在しないすべてのSQL文を示します。この例では、2番目のSQLチューニング・セットで新しく追加されたSQL文は2つだけです。例6-9に示すように、これらSQL文は次のような内容になっています。

- sql_id値: 4c8nrqxhtb2sfおよび9utadgu5udmh4
- ワークロードに対するパフォーマンス全体への影響: -8.48%
- 「Total Metric Before」の変更値: なし

例6-9に、欠落しているSQL文と新しいSQL文を示すサンプル・レポートの表を示します。ワークロードへの影響ごとに特定されたその他の上位SQL文も示されています。

例6-9 ワークロードに対する変更の影響の絶対値ごとに分類された上位7つのSQL

```
Top 7 SQL Sorted by Absolute Value of Change Impact on the Workload
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
|           |               | Impact on | Total Metric | Total Metric | Impact   | Plan   |
| object_id | sql_id        | Workload  |    Before    |    After     | on SQL   | Change |
-------------------------------------------------------------------------------------------------
|         4 | 7gj3w9ya4d9sj |    41.04% |       812791 |        36974 |      95% | y      |
|         7 | gv7xb8tyd1v91 |     33.1% |       625582 |              |          | n      |
|         2 | 4c8nrqxhtb2sf |    -8.35% |              |       157782 |          | n      |
|         1 | 22u3tvrt0yr6g |     4.58% |       302190 |       215681 |  28.63%  | n      |
|         6 | fgdd0fd56qmt0 |      2.1% |       146128 |       106369 |  27.21%  | n      |
|         5 | 9utadgu5udmh4 |     -.13% |              |         2452 |          | n      |
|         3 | 4dtv43awxnmv3 |     -.02% |         3520 |         3890 | -47.35%  | n      |
-------------------------------------------------------------------------------------------------
```

対象のSQL文を識別した後、そのSQL文のレポートを作成して、さらに詳しく調べることができます。たとえば、ワークロードに対して最も大きな影響のあった、sql_idの値が7gj3w9ya4d9sjで、object_idの値が4のSQL文を調べることもできます。

```
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(task_name => :aname, object_id => 4) rep
FROM dual;
```

例6-10に、このSQL文に対して生成されたサンプル・レポートを示します。

例6-10 SQL文のサンプル・レポート

```
SQL Details:
-----------------------------
 Object ID  : 4
 SQL ID     : 7gj3w9ya4d9sj
 SQL Text   : /* my_csts_query1 */ select * FROM emp where empno=2

SQL Execution Statistics (average):
---------------------------------------
-----------------------------------------------------
|              | Impact on | Value   | Value   | Impact |
| Stat Name    | Workload  | Before  | After   | on SQL |
-----------------------------------------------------
| elapsed_time |    41.04% | .036945 | .001849 |    95% |
| cpu_time     |    13.74% | .004772 |  .00185 | 61.24% |
| buffer_gets  |     9.59% |       8 |       2 | 69.01% |
| cost         |    11.76% |       1 |       1 |    10% |
| reads        |     4.08% |       0 |       0 | 63.33% |
```

```
| writes       |        0% |       0 |       0 |     0% |
| rows         |           |       0 |       0 |        |
| executions   |           |      22 |      20 |        |
| plan_count   |           |       3 |       2 |        |
-------------------------------------------------------

Findings (2):
-----------------------------
 1. The performance of this SQL has improved.
 2. The structure of the SQL execution plan has changed.

Plan Execution Statistics (average):
--------------------------------------
-------------------------------------------------------------------------------------
| Statistic Name  | Plans Before Change              | Plans After Change        |
-------------------------------------------------------------------------------------
| plan hash value | 440231712  571903972  3634526668 | 571903972  3634526668     |
| --------------- | ---------  ---------  ---------- | ---------  ----------     |
|  schema name    | APPS1      APPS2      APPS2      | APPS2      APPS2          |
|  executions     | 7          5          10         | 10         10             |
|  cost           | 2          1          2          | 1          2              |
|  elapsed_time   | .108429    .000937    .00491     | .000503    .003195        |
|  cpu_time       | .00957     .0012      .0032      | .0005      .0032          |
|  buffer_gets    | 18         0          5          | 0          5              |
|  reads          | 0          0          0          | 0          0              |
|  writes         | 0          0          0          | 0          0              |
|  rows           | 0          0          0          | 0          0              |
-------------------------------------------------------------------------------------

Execution Plans Before Change:
------------------------------
Plan Hash Value  : 440231712
-----------------------------------------------------------------------------
| Id | Operation              | Name     | Rows | Bytes | Cost | Time     |
-----------------------------------------------------------------------------
|  0 | SELECT STATEMENT       |          |      |       |    2 |          |
|  1 |   PX COORDINATOR       |          |      |       |      |          |
|  2 |    PX SEND QC (RANDOM) | :TQ10000 |    1 |    87 |    2 | 00:00:01 |
|  3 |     PX BLOCK ITERATOR  |          |    1 |    87 |    2 | 00:00:01 |
|  4 |      TABLE ACCESS FULL | EMP      |    1 |    87 |    2 | 00:00:01 |
-----------------------------------------------------------------------------

Note
-----
- dynamic sampling used for this statement
Plan Hash Value  : 571903972
-------------------------------------------------------------------------------------
| Id | Operation                     | Name       | Rows | Bytes | Cost | Time   |
-------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |            |      |       |    1 |        |
|  1 |   TABLE ACCESS BY INDEX ROWID | EMP        |    1 |    87 |    1 |00:00:01|
|  2 |    INDEX UNIQUE SCAN          | MY_EMP_IDX |    1 |       |    0 |        |
-------------------------------------------------------------------------------------

Plan Hash Value  : 3634526668
----------------------------------------------------------------------
| Id | Operation           | Name | Rows | Bytes | Cost | Time     |
----------------------------------------------------------------------
|  0 | SELECT STATEMENT    |      |      |       |    2 |          |
|  1 |   TABLE ACCESS FULL | EMP  |    1 |    87 |    2 | 00:00:01 |
----------------------------------------------------------------------

Note
```

```
-----
- dynamic sampling used for this statement

Executions Plan After Change:
-----------------------------
Plan Hash Value  : 571903972
-------------------------------------------------------------------------------------
| Id | Operation                     | Name       | Rows | Bytes | Cost | Time    |
-------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |            |      |       |    1 |         |
|  1 |   TABLE ACCESS BY INDEX ROWID | EMP        |    1 |    87 |    1 |00:00:01|
|  2 |    INDEX UNIQUE SCAN          | MY_EMP_IDX |    1 |       |    0 |         |
-------------------------------------------------------------------------------------
Plan Hash Value  : 3634526668
-----------------------------------------------------------------
| Id | Operation           | Name | Rows | Bytes | Cost | Time     |
-----------------------------------------------------------------
|  0 | SELECT STATEMENT    |      |      |       |    2 |          |
|  1 |   TABLE ACCESS FULL | EMP  |    1 |    87 |    2 | 00:00:01 |
-----------------------------------------------------------------
Note
-----
- dynamic sampling used for this statement
-------------------------------------------------------------------------------------
```

「SQL Execution Statistics」セクションには、SQL文の(実行ごとの)平均実行時統計が表示されます。この表のデータから、このSQL文は両方のSQLチューニング・セットに存在していますが、最初のSQLチューニング・セットの実行計画は3つ、2番目のSQLチューニング・セットの実行計画は2だということがわかります。さらに、SQL文は最初のSQLチューニング・セットでは22回実行されたのに対し、2番目のSQLチューニング・セットでは20回しか実行されていません。

「Plan Execution Statistics」セクションには、実行計画(または計画ハッシュ値)ごとの実行時統計が表示されます。「Plans Before Change」列には、最初のSQLチューニング・セットについての計画とその計画に関連付けられた実行統計が表示され、「Plans After Change」列には、2番目のSQLチューニング・セットについてのそれらの値が表示されています。レポートの最後に、両方のSQLチューニング・セットの実行計画の構造が示されています。

レポートのこれらのセクションを使用して、2つのSQLチューニング・セット間の実行計画の違いを識別できます。これは、実行計画の違いによって、テスト結果が変わり、パフォーマンスに直接影響が出る可能性があるため重要です。2つのSQLチューニング・セットを比較した場合、SQL文が次のように設定されていると、SQLパフォーマンス・アナライザは実行計画の違いをレポートします。

- 両方のSQLチューニング・セットに計画が1つであるにもかかわらず、計画の構造が異なっている。
- 計画が複数で、両方のSQLチューニング・セットの計画の数が次のようになっている。
    - 計画の数は同じでも、2番目のSQLチューニング・セットの1つ以上の計画が、最初のSQLチューニング・セットのすべての計画と異なっている。
    - 計画数が異なっている

SQL文と計画の違いを評価した後、さらに処置が必要かどうかを決定します。SQL文のパフォーマンスが低下している場合は、次のいずれかの処置を実行します。

- パフォーマンスが低下したSQL文をチューニングします。詳細は、「APIを使用した、パフォーマンスが低下したSQL文の

チューニング」を参照してください

- SQL計画ベースラインを作成します。詳細は、「APIを使用したSQL計画ベースラインの作成」を参照してください

APIを使用した、パフォーマンスが低下したSQL文のチューニング

SQLパフォーマンス・アナライザ・レポートを確認したら、SQLパフォーマンスの比較後に特定されるパフォーマンスが低下したSQL文をチューニングする必要があります。パフォーマンスが低下したSQL文が多数表示される場合は、根本原因を特定し、システム・レベルの変更を行って問題を修正する必要があります。パフォーマンスが低下したSQL文が少数の場合は、SQLチューニング・アドバイザを使用して、部分的な解決を検討するか、SQL計画ベースラインを作成して、今後は変更前の実行計画を選択するようにオプティマイザに指示します。

APIを使用して、パフォーマンスが低下したSQL文をチューニングするには:

- DBMS_SQLTUNEパッケージのCREATE_TUNING_TASKファンクションを使用して、SQLパフォーマンス・アナライザを実行するためのSQLチューニング・タスクを作成します。

```
BEGIN
  DBMS_SQLTUNE.CREATE_TUNING_TASK(
    spa_task_name => 'my_spa_task',
    spa_task_owner => 'immchan',
    spa_compare_exec => 'my_exec_compare');
  DBMS_SQLTUNE.EXECUTE_TUNING_TASK(spa_task_name => 'my_spa_task');
END;
/
```

この例では、my_spa_taskという名前のSQLパフォーマンス・アナライザ・タスクについて、my_exec_compareという名前でパフォーマンスの比較を実行したときにパフォーマンスが低下したSQL文をチューニングするSQLチューニング・タスクを作成し、実行します。この場合、このバージョンのCREATE_TUNING_TASKファンクション・コールを使用することが重要です。そうしないと、SQL文が取得された本番システムの環境でチューニングが行われる可能性があるため、システムの変更が反映されません。

ノート:

別のデータベースで SQL ワークロードをリモートで実行することを選択した場合は、パフォーマンスが低下した SQL 文のチューニングに、このバージョンの CREATE_TUNING_TASK ファンクション・コールを使用しないでください。かわりに、リモート・データベース上の SQL 試行によって識別されたパフォーマンスの低下をチューニングしてください(これは、アプリケーション・スキーマが SQL パフォーマンス・アナライザを実行しているデータベース上に存在していないためです)。そのため、スキーマが存在し、変更が行われたデータベースで、SQL チューニング・アドバイザを実行する必要があります。

表6-1に、DBMS_SQLTUNE.CREATE_TUNING_TASKファンクションで使用できるSQLパフォーマンス・アナライザのパラメータを示します。

表6-1 CREATE_TUNING_TASKファンクションのSQLパフォーマンス・アナライザのパラメータ

パラメータ	説明

パラメータ	説明
SPA_TASK_NAME	SQL パフォーマンス・アナライザ・タスクの名前。
SPA_TASK_OWNER	指定した SQL パフォーマンス・アナライザ・タスクの所有者。指定しないと、デフォル値の現行ユーザーになります。
SPA_COMPARE_EXEC	指定した SQL パフォーマンス・アナライザ・タスクのパフォーマンスの比較試行の実行名。指定しないと、デフォルト値として、指定した SQL パフォーマンス・アナライザ・タスクで最後に実行した COMPARE PERFORMANCE タイプの名前になります。

パフォーマンスが低下したSQL文をチューニングしたら、SQLパフォーマンス・アナライザを使用して、それらの変更をテストする必要があります。新しいSQL試行をテスト・システムで実行すると、2番目の比較(新しいSQL試行と最初の試行の比較)が実行され、結果が検証されます。SQLパフォーマンス・アナライザによってパフォーマンスが安定していることが表示されたら、このステップで行った修正を本番システムに対して実行します。

Oracle Database 11g リリース2以上では、SQL文のチューニング時に、SQLチューニング・アドバイザによって代替計画が分析されます。SQLチューニング・アドバイザは、自動ワークロード・リポジトリに保存されている履歴計画も含め、SQL文の実行履歴を確認します。SQLチューニング・アドバイザによって代替計画が検出された場合は、そのSQL文で適切な実行計画が使用されるように、特定の計画を選択して計画ベースラインを作成できます。

関連項目:

- 「APIを使用した、リモートSQL試行からのパフォーマンスが低下したSQL文のチューニング」
- SQLチューニング・アドバイザの使用方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- 代替計画の分析の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- DBMS_SQLTUNEパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照

APIを使用した、リモートSQL試行からのパフォーマンスが低下したSQL文のチューニング

別のデータベースでSQLワークロードをリモートで実行することを選択した場合は、SQLパフォーマンス・アナライザ・タスクが存在するシステムではなく、リモート・データベースでSQL試行によって識別されたパフォーマンスの低下をチューニングする必要があります。

APIを使用して、リモートSQL試行からのパフォーマンスが低下したSQL文をチューニングするには:

1. SQLパフォーマンス・アナライザが実行されているシステムで、パフォーマンスが低下しているSQL文のサブセットをSQLチューニング・セットとして作成します。

```
DECLARE
  sqlset_cur   DBMS_SQLTUNE.SQLSET_CURSOR;
```

```
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('SUB_STS1', 'test purpose');

  OPEN sqlset_cur FOR
    SELECT value(p)
    FROM table(
      DBMS_SQLTUNE.SELECT_SQLPA_TASK(
        task_name  => 'SPA_TASK1',
        execution_name => 'COMP',
        level_filter => 'REGRESSED')) p;

  DBMS_SQLTUNE.LOAD_SQLSET('SUB_STS1', sqlset_cur);

  CLOSE sqlset_cur;
END;
/
```

'REGRESSED'以外にも、'CHANGED'、'ERRORS'、'CHANGED_PLANS'などのフィルタを使用して、SQLチューニング・セットのSQL文を選択できます。詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

2. SQLチューニング・セットをエクスポートするステージング表を作成します。

```
BEGIN
  DBMS_SQLTUNE.CREATE_STGTAB_SQLSET(
    table_name  => 'STG_TAB1',
    schema_name => 'JOHNDOE',
    tablespace_name => 'TBS_1',
    db_version => DBMS_SQLTUNE.STS_STGTAB_11_1_VERSION);
END;
/
```

db_versionパラメータを使用して、SQLチューニング・セットのエクスポートとチューニングを行う適切なバージョンのデータベースを指定します。この例では、Oracle Database 11g リリース1を実行しているシステムにエクスポートできる形式でステージング表を作成し、後でSQLチューニング・アドバイザを使用してこの表をチューニングします。その他のデータベース・バージョンの詳細は、そのリリースの『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

3. SQLチューニング・セットをステージング表にエクスポートするには、次の手順を実行します。

```
BEGIN
  DBMS_SQLTUNE.PACK_STGTAB_SQLSET(
    sqlset_name => 'SUB_STS1',
    sqlset_owner => 'JOHNDOE',
    staging_table_name => 'STG_TAB1',
    staging_schema_owner => 'JOHNDOE',
    db_version => DBMS_SQLTUNE.STS_STGTAB_11_1_VERSION);
END;
/
```

4. 任意の方法(Oracle Data Pumpやデータベース・リンクなど)でステージング表を(SQLワークロードが実行された)リモート・データベースに移動します。

5. リモート・データベースで、ステージング表からSQLチューニング・セットをインポートします。

```
BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET(
```

```
    sqlset_name => 'SUB_STS1',
    staging_table_name => 'STG_TAB1',
    replace => TRUE);
END;
/
```

6. SQLチューニング・アドバイザを実行し、SQLチューニング・セット内のパフォーマンスが低下したSQL文をチューニングします。

```
BEGIN
  sts_name  := 'SUB_STS1';
  sts_owner := 'JOHNDOE';
  tune_task_name := 'TUNE_TASK1';
  tname := DBMS_SQLTUNE.CREATE_TUNING_TASK(sqlset_name  => sts_name,
                                           sqlset_owner => sts_owner,
                                           task_name    => tune_task_name);
  EXEC DBMS_SQLTUNE.SET_TUNING_TASK_PARAMETER(:tname,
                                              'APPLY_CAPTURED_COMPILENV',
                                              'FALSE');
  exec_name := DBMS_SQLTUNE.EXECUTE_TUNING_TASK(tname);
END;
/
```

ノート:

この例で使用する APPLY_CAPTURED_COMPILENV パラメータは、Oracle Database 11g リリース 1 以上のリリースでのみサポートされています。これより前のバージョンの Oracle Database からデータベース・アップグレードをテストする場合は、かわりに SQL チューニング・セットに保存されている環境変数が使用されます。

パフォーマンスが低下したSQL文をチューニングしたら、SQLパフォーマンス・アナライザを使用して、それらの変更をテストする必要があります。新しいSQL試行をテスト・システムで実行すると、2番目の比較(新しいSQL試行と最初の試行の比較)が実行され、結果が検証されます。SQLパフォーマンス・アナライザによってパフォーマンスが安定していることが表示されたら、このステップで行った修正を本番システムに対して実行します。

関連項目:

- SQLチューニング・アドバイザの使用方法およびSQLチューニング・セットの転送の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- DBMS_SQLTUNEパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照

APIを使用したSQL計画ベースラインの作成

SQLチューニング・アドバイザを実行するためのもう1つの方法として、計画の変更によってパフォーマンスが低下したSQL文のSQL計画ベースラインを作成することがあります。これにより、このようなSQL文に対しては、今後変更前の実行計画を使用するように、オプティマイザに指示されます。

元の計画用のSQL計画ベースラインを作成するには:

1. パフォーマンスが低下したSQL文のSQLチューニング・セットのサブセットのみを作成します。
2. DBMS_SPMパッケージのLOAD_PLANS_FROM_SQLSETファンクションを使用して、計画をロードして、このSQL文のサブセットのSQL計画ベースラインを作成します(次の例を参照してください)。

```
DECLARE
  my_plans PLS_INTEGER;
BEGIN
  my_plans := DBMS_SPM.LOAD_PLANS_FROM_SQLSET(
    sqlset_name => 'regressed_sql');
END;
/
```

関連項目:

- SQL計画ベースラインの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- DBMS_SPMパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

SQLパフォーマンス・アナライザのビューの使用

次のビューを問い合せて、SQLパフォーマンス・アナライザを監視し、分析結果を表示することができます。

ノート:

これらのビューで参照可能な情報は、SQL パフォーマンス・アナライザ・レポートにも含まれています。かわりに、SQL パフォーマンス・アナライザ・レポートを使用して分析結果を表示することをお薦めします。結果のより詳細な分析を実行する場合にのみ、これらのビューを使用することを検討してください。

- DBA_ADVISOR_TASKSおよびUSER_ADVISOR_TASKSビューには、作成されたSQLパフォーマンス・アナライザのタスクに関する記述情報が表示されます。
- DBA_ADVISOR_EXECUTIONSおよびUSER_ADVISOR_EXECUTIONSビューには、タスク実行に関する情報が表示されます。SQLパフォーマンス・アナライザでは、3つ以上の実行が作成され、SQLワークロードに対するデータベース変更によって発生するSQLパフォーマンスへの影響が分析されます。最初の実行では、変更前バージョンのパフォーマンス・データが収集されます。2番目の実行では、変更後バージョンのパフォーマンス・データが収集されます。3番目の実行では、比較分析が実行されます。
- DBA_ADVISOR_FINDINGSおよびUSER_ADVISOR_FINDINGSビューには、SQLパフォーマンス・アナライザの結果が表示されます。SQLパフォーマンス・アナライザでは、次のタイプの結果が生成されます。
  - 問題(パフォーマンスの低下など)
  - 兆候(実行計画の構造が変化した場合など)
  - エラー(オブジェクトまたはビューが存在しない場合など)
  - 情報メッセージ(変更前バージョンの実行計画の構造がSQLチューニング・セットに格納されているものと異なる場合など)

- DBA_ADVISOR_SQLPLANSおよびUSER_ADVISOR_SQLPLANSビューには、すべての実行計画のリストが表示されます。
- DBA_ADVISOR_SQLSTATSおよびUSER_ADVISOR_SQLSTATSビューには、すべてのSQLコンパイルおよび実行統計のリストが表示されます。
- V$ADVISOR_PROGRESSビューには、SQLパフォーマンス・アナライザの操作の進行状況が表示されます。このビューを使用して、完了しているSQL文の数およびSQL試行で実行待機中のSQL文の数を監視します。SOFAR列は、これまでに処理されたSQL文の数を示し、TOTAL WORK列は、タスクの実行によって処理されるSQL文の合計を示します。

DBAビューにアクセスするには、SELECT_CATALOG_ROLEロールが必要です。

関連項目:

- DBA_ADVISOR_TASKS、DBA_ADVISOR_EXECUTIONSおよびDBA_ADVISOR_SQLPLANSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

7 SPAクイック・チェックの使用

Oracle Enterprise Manager Cloud Control (Cloud Control)には、SQLパフォーマンス・アナライザ・クイック・チェック(SPAクイック・チェック)機能が含まれています。一部のCloud Controlデータベース管理ページで、SPAクイック・チェックは、変更を加える前にデータベース・ワークロードへのシステム変更の影響を検証できます。

SPAクイック・チェックを使用して、次の変更でのデータベース・ワークロードへの影響を検証できます。

- 初期化パラメータの値の変更
- 保留中のオプティマイザ統計の収集
- キーSQLプロファイルの実装

ノート:

SPA クイック・チェックは、Cloud Control リリース 12.1.0.4 バンドル・パッチ 8 以降で使用可能です。

SPA クイック・チェックは、Oracle Database 10g リリース 2 (10.2)以降を実行中のデータベースでサポートされています。ただし、すべての SPA クイック・チェック機能が Oracle Database 10g リリース 2 (10.2)でサポートされているわけではありません。

たとえば、保留中のオプティマイザ統計および自動 SQL チューニング・アドバイザの機能は、Oracle Database 11g リリース 1 (11.1)以降で使用可能であるため、これらの機能の SPA クイック・チェック・ワークフローは、Oracle Database 11g リリース 1 (11.1)以降を実行中のデータベースでのみサポートされています。

この章では、次のトピックで、SPAクイック・チェックの使用方法について説明します。

- 「SPAクイック・チェックの構成について」
- 「SPAクイック・チェックのデフォルト値の指定」
- 「初期化パラメータの変更の影響の検証」
- 「保留中のオプティマイザ統計の影響の検証」
- 「キーSQLプロファイルの実装の影響の検証」
- 「自動SQLチューニング・アドバイザの統計結果の検証」

SPAクイック・チェックの構成について

SPAクイック・チェックを使用して初期化パラメータの変更または保留中のオプティマイザ統計の収集の影響を検証する前に、SPAクイック・チェックのデフォルト設定を指定する必要があります。

設定の1つとして使用するSPAクイック・チェックのデフォルトのSQLチューニング・セットを指定します。このSQLチューニング・セットには、チューニングを試行するデータベース・アプリケーションで使用されるSQL文を含める必要があります。

ノート:

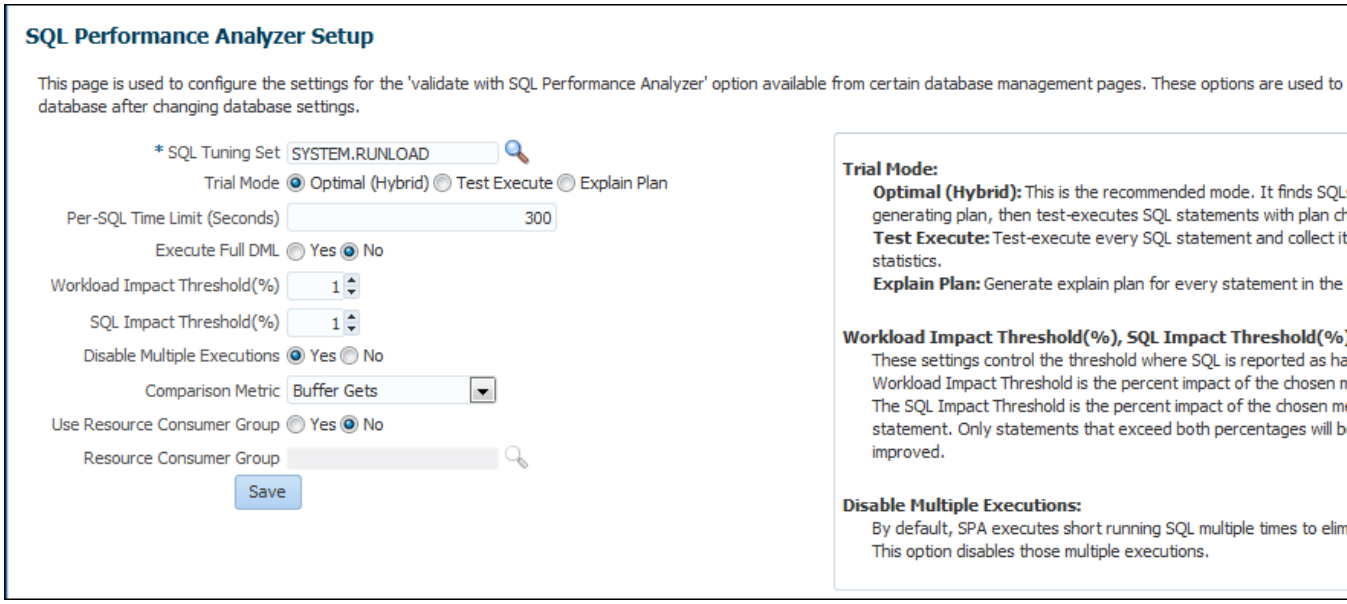
SPA クイック・チェックを使用して 1 つ以上のキーSQL プロファイルの実装の影響を検証する前に、SPA クイック・チェックのデフォルト値を設定する必要はありません。

SPAクイック・チェックのデフォルト値の指定

Cloud Controlの「SQLパフォーマンス・アナライザの設定」ページで、SPAクイック・チェックのデフォルト設定を指定します。

SPAクイック・チェックのデフォルト設定を指定するには:

1. Cloud Controlの「データベース・ホーム」ページで、「パフォーマンス」メニューから、「SQL」、「SQLパフォーマンス・アナライザの設定」の順に選択します。「データベース・ログイン」ページが表示されたら、データベースの管理者権限を入力し、「ログイン」をクリックします。
   「SQLパフォーマンス・アナライザの設定」ページが表示されます。



2. 一部のCloud Controlデータベース管理ページで使用可能なSPAクイック・チェック機能の設定を構成します。指定するSQLチューニング・セットは、チューニングするアプリケーションのワークロードを代表するものである必要があります。
3. 「保存」をクリックして、指定したデフォルトのSPAクイック・チェック設定を保存します。

初期化パラメータの変更の影響の検証

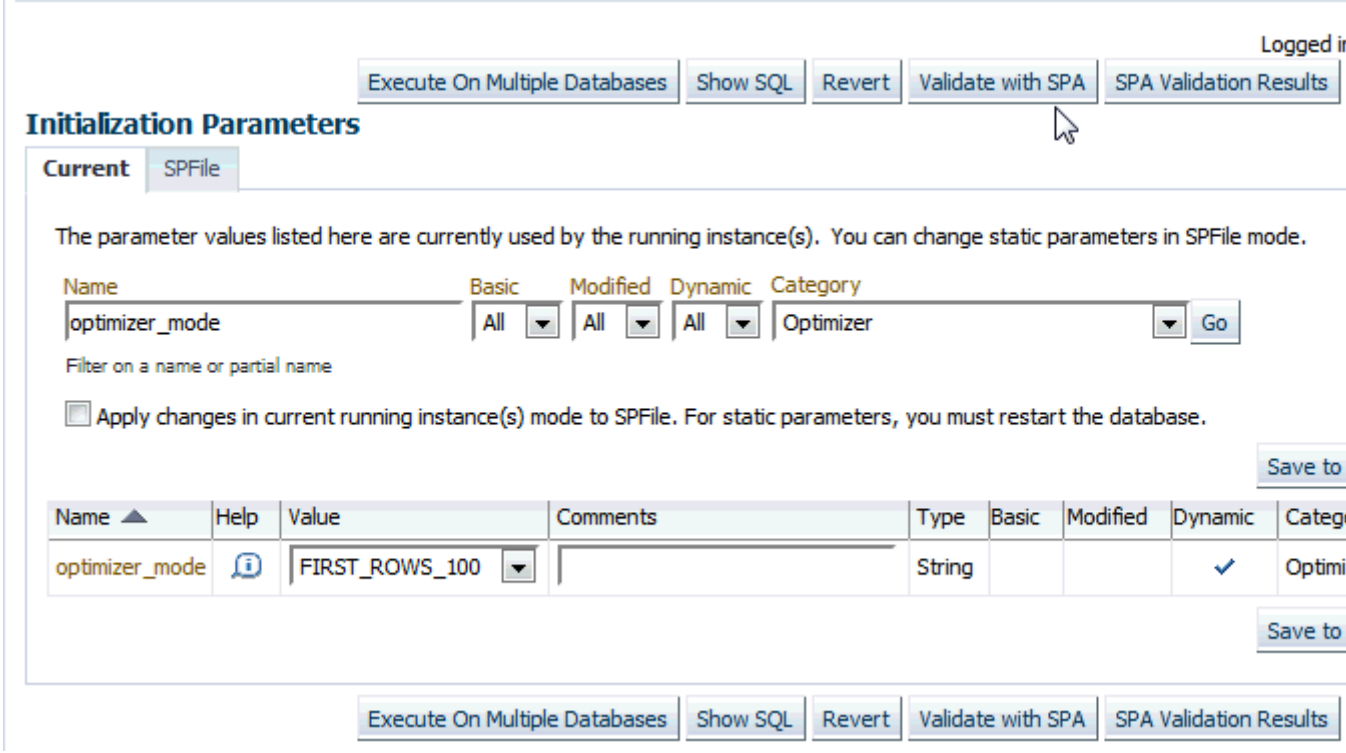
セッションで変更可能な初期化パラメータの値を変更する前に、SPAクイック・チェックを使用してデータベース・ワークロードのその変更の影響を検証できます。セッションで変更可能なパラメータは、ALTER SESSION文を使用して値を変更できる初期化パラメータです。

ノート:

SPA クイック・チェックを使用して、Oracle Database 10g リリース 2 (10.2)以降を実行中のデータベースで、初期化パラメータの変更の影響を検証できます。

初期化パラメータの影響を検証するには:

1. Cloud Controlの「データベース・ホーム」ページで、「管理」メニューから、「初期化パラメータ」を選択します。
   「初期化パラメータ」ページが表示されます。
2. 「初期化パラメータ」ページのフィルタを使用して、値を変更する、セッションで変更可能な初期化パラメータを特定し、「実行」をクリックして、ページの下部にある表にそのパラメータを表示します。「オプティマイザ」カテゴリのほとんどのパラメータは、セッションで変更可能です。
3. 表で、現在のパラメータの値を、SPAクイック・チェックを使用して影響を検証する新しい値に変更します。
4. 「SPAを使用した検証」をクリックします。



   情報メッセージがページ上部に表示され、初期化パラメータの変化の影響を検証するSPAタスクが発行されたことが通知されます。
5. 情報メッセージ内のSPAタスクのリンクをクリックします。
   「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。
6. ページの下部にある「SQLパフォーマンス・アナライザのタスク」セクションで、初期化パラメータ・ジョブのタスクを選択し、「最新レポートの表示」をクリックします。
   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。
7. ページの下部にある表を表示して、初期化パラメータの値の変更の結果がワークロードで最も影響力を持つSQL文にあることを確認します。

保留中のオプティマイザ統計の影響の検証

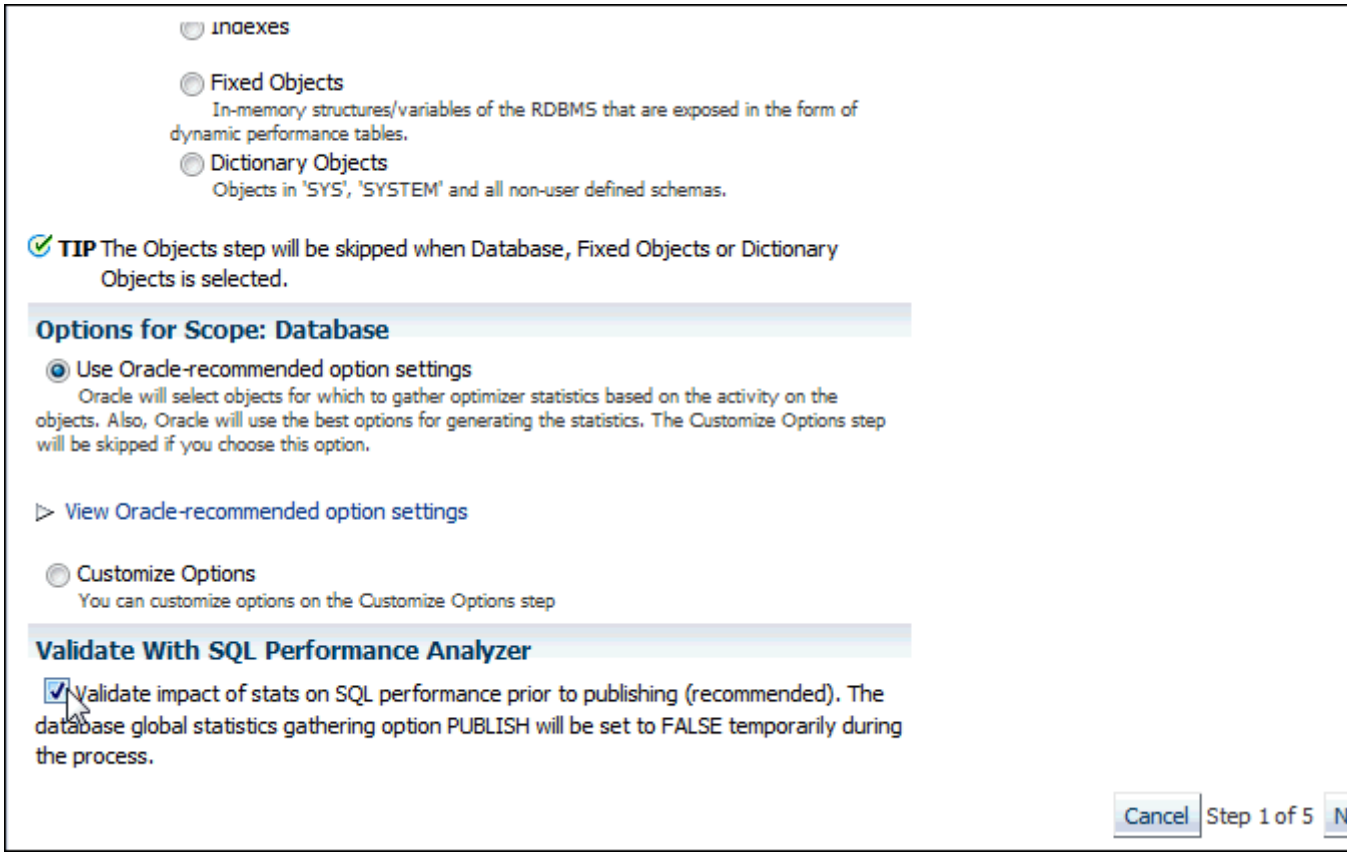
保留中のオプティマイザ統計を収集する前に、SPAクイック・チェックを使用してデータベース・ワークロードでのその統計の収集の影響を検証できます。

ノート:

SPA クイック・チェックを使用して、Oracle Database 11g リリース 1 (11.1)以降を実行中のデータベースで、保留中のオプティマイザ統計の収集の影響を検証できます。

保留中のオプティマイザ統計の収集の影響を検証するには:

1. Cloud Controlの「データベース・ホーム」ページで、「パフォーマンス」メニューから、「SQL」、「オプティマイザ統計」の順に選択します。
   「オプティマイザ統計コンソール」ページが表示されます。
2. 「操作」セクションで、「採取」をクリックします。
   オプティマイザ統計の採取ウィザードが表示されます。
3. 「オプティマイザ統計の採取: 有効範囲」ページの下部にある「SQLパフォーマンス・アナライザで検証」セクションで、公開前にSQLパフォーマンスで統計の影響を検証します(推奨)オプションを有効にします。データベース・グローバル統計の採取オプションPUBLISHは、プロセス中に一時的にFALSEに設定されます。その後、「次」をクリックします。



4. ウィザードを進み、「オプティマイザ統計の採取: 有効範囲」ページで、「発行」をクリックします。
   一時停止統計の収集に加え、これにより、データベースの最適化統計の収集の影響を検証するSQLパフォーマンス・アナライザ・タスクを作成するジョブが開始されます。
5. ジョブが開始すると、オプティマイザ統計の収集ジョブが正常に発行されたことを示す確認メッセージが、「オプティマイザ統計の管理」ページに表示されます。メッセージ内のリンクをクリックします。
   「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。
6. ページの下部にある「SQLパフォーマンス・アナライザのタスク」表で、統計収集ジョブが完了したことを確認します。ジョブが完了するまでに数分かかる場合があります。次に、オプティマイザ統計の採取ジョブの行を選択し、「最新レポートの表示」をクリックします。
   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。
7. ページの下部にある表を表示して、保留中のオプティマイザ統計の発行の結果がワークロードで最も影響力を持つSQL文にあることを確認します。

キーSQLプロファイルの実装の影響の検証

SQL文のキーSQLプロファイルを実装する前に、SPAクイック・チェックを使用してそのプロファイルの使用の影響を検証できます。

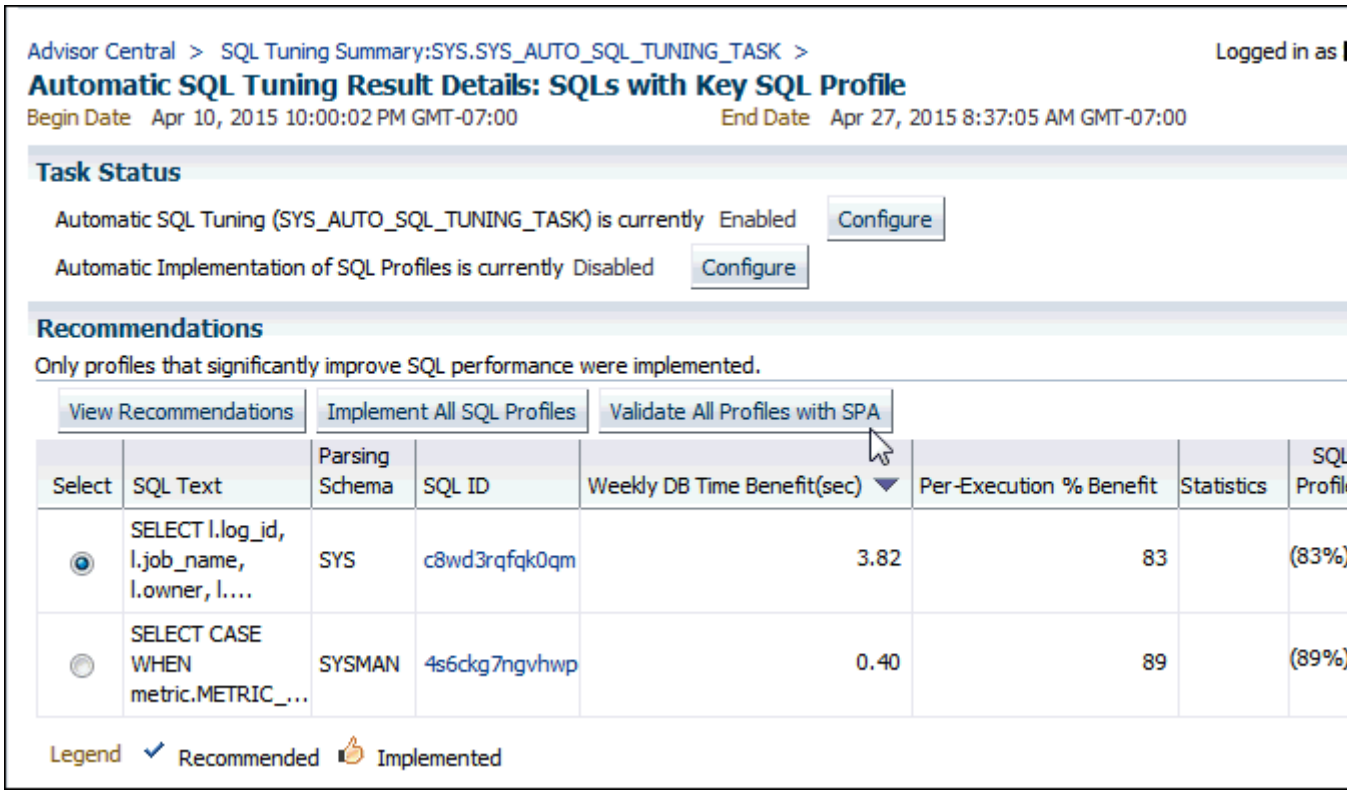
「自動SQLチューニング結果のサマリー」ページで、キーSQLプロファイルの影響を検証できます。キーSQLプロファイルは、パフォーマンスが少なくとも3倍向上することが検証され、自動SQLチューニング・アドバイザで自動実装が有効になっていると自動的に実装されるプロファイルです。

ノート:

SPA クイック・チェックを使用して、Oracle Database 11g リリース 1 (11.1)以降を実行中のデータベースで、キーSQL プロファイルの実装の影響を検証できます。

キーSQLプロファイルの影響を検証するには:

1. Cloud Controlの「データベース・ホーム」ページで、「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。
   「アドバイザ・セントラル」ページが表示されます。
2. 「アドバイザ」セクションで、「SQLアドバイザ」をクリックします。
   「SQLアドバイザ」ページが表示されます。
3. 「SQLチューニング・アドバイザ」セクションで、「自動SQLチューニングの結果」をクリックします。
   「自動SQLチューニング結果のサマリー」ページが表示されます。
4. 「タスク・ステータス」セクションの「キーSQLプロファイル」フィールドに、現在の自動SQLチューニング・タスクのキーSQLプロファイルの数がリストされます。フィールドに値0が表示される場合、使用(または検証)するキーSQLプロファイルはありません。「キーSQLプロファイル」フィールドに0より大きい値が表示された場合、値をクリックして、1つ以上のキーSQLプロファイルを使用する場合の影響を検証します。
   「自動SQLチューニング結果の詳細: キーSQLプロファイルのあるSQL」ページが表示されます。
5. キーSQLプロファイルが「推奨」セクションに表示されます。「すべてのプロファイルをSPAを使用して検証してください」をクリックします。



   確認文がページ上部に表示され、SQLプロファイルを検証するSPAタスクが発行されたことが通知されます。
6. 確認文内のSPAタスクのリンクをクリックします。

「SQLパフォーマンス・アナライザ・ホーム」ページが表示され、キーSQLプロファイルを検証するSPAタスクが、ページの下部にある「SQLパフォーマンス・アナライザのタスク」表に表示されます。

7. タスクをクリックし、「最新レポートの表示」をクリックします。
   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。
8. ページの下部にある表を表示して、結果がワークロードで最も影響力を持つSQL文の「自動SQLチューニング結果のサマリー」ページで推奨されているキーSQLプロファイルの実装であることを確認します。

自動SQLチューニング・アドバイザの統計結果の検証

SPAクイック・チェックを使用して、自動SQLチューニング・アドバイザの統計結果の影響を検証できます。

ノート:

SPA クイック・チェックを使用して、Oracle Database 11g リリース 1 (11.1)以降を実行中のデータベースで、自動 SQL チューニング・アドバイザの統計結果の検証の影響を検証できます。

自動SQLチューニング・アドバイザの統計結果の影響を検証するには:

1. Cloud Controlの「データベース・ホーム」ページで、「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。
   「アドバイザ・セントラル」ページが表示されます。
2. 「アドバイザ」セクションで、「SQLアドバイザ」をクリックします。
   「SQLアドバイザ」ページが表示されます。
3. 「SQLチューニング・アドバイザ」セクションで、「自動SQLチューニングの結果」をクリックします。
   「自動SQLチューニング結果のサマリー」ページが表示されます。
4. 統計結果は、使用可能な場合、ページの下部の近くにある「統計結果サマリー」セクションに表示されます。ユーザー・スキーマの統計結果の影響を検証するには、「SPAを使用した検証」をクリックします。
   確認文がページ上部に表示され、統計結果を検証するSPAタスクが発行されたことが通知されます。
5. 確認文内のSPAタスクのリンクをクリックします。
   「SQLパフォーマンス・アナライザ・ホーム」ページが表示され、統計結果を検証するSPAタスクが、ページの下部にある「SQLパフォーマンス・アナライザのタスク」表に表示されます。
6. タスクのすべてのステップが正常に完了し、表の「最終実行ステータス」列に「完了」と表示されたら、タスクを選択し、「最新レポートの表示」をクリックします。タスクのステップがすべて完了するまでに数分かかる場合があります。
   「SQLパフォーマンス・アナライザのタスク・レポート」ページが表示されます。
7. ページの下部にある表を表示して、結果がワークロードで最も影響力を持つSQL文の「自動SQLチューニング結果のサマリー」ページの統計の実装であることを確認します。

8 データベースのアップグレードのテスト

SQLパフォーマンス・アラナイザは、Oracle9i以降のリリースからOracle Database 10gリリース2以降へのデータベース・アップグレードのテストをサポートしています。Oracle9i DatabaseおよびOracle Database 10gリリース1からのデータベース・アップグレードのテストに使用する方法は、Oracle Database 10gリリース2以降からのデータベース・アップグレードのテストに使用する方法とわずかに異なるので、ここでは両方の方法を説明します。

この章では、データベースのアップグレードにSQLパフォーマンス・アナライザを使用する方法について説明します。内容は次のとおりです。

- Oracle9i DatabaseおよびOracle Database 10gリリース1からのアップグレード
- Oracle Database 10gリリース2以上のリリースからのアップグレード
- データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング

関連項目:

- その他の事例でのSQLパフォーマンス・アナライザの使用の詳細は、「SQLパフォーマンス・アナライザ」を参照してください
- Oracle Database 12cのアップグレード・パスの詳細は、『Oracle Databaseアップグレード・ガイド』を参照してください。

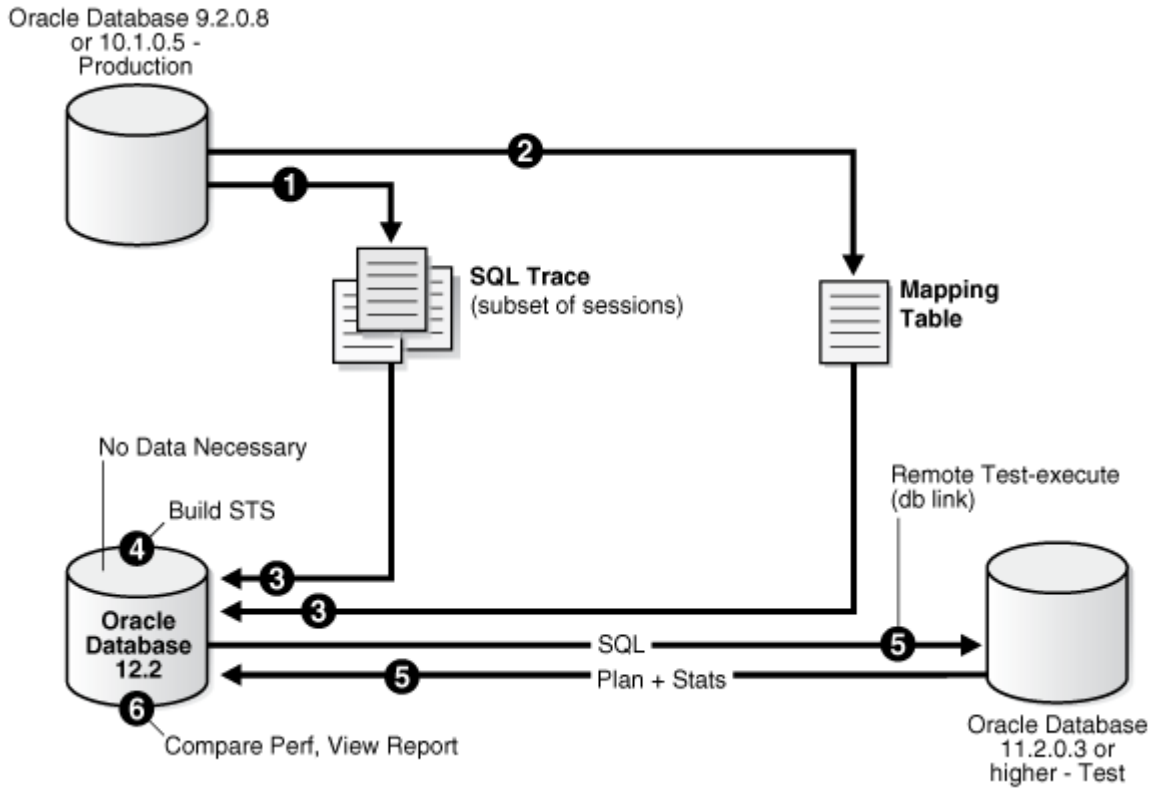
Oracle9i DatabaseおよびOracle Database 10g リリース1からのアップグレード

SQLパフォーマンス・アナライザでは、Oracle9i DatabaseとOracle Database 10gリリース1からOracle Database 11gリリース2以上のリリースへのデータベース・アップグレードのテストをサポートしています。次のステップを使用し、図8-1を確認してください。

- 本番システムで取得したSQLトレース・ファイルからのSQLチューニング・セットの構築
- データベース・リンクを介したのアップグレードされたデータベースに対するリモートからSQLチューニング・セットの実行
- 本番システムで取得した結果との比較

SQLパフォーマンス・アナライザでは、入力ソースとしてSQLチューニング・セットに格納されているSQL文のセットしか受け取らないため、またSQLチューニング・セットはOracle9iデータベースではサポートされていないため、Oracle9iデータベースからアップグレードする場合は、SQLパフォーマンス・アナライザの入力ソースとして使用できるようSQLチューニング・セットを構築する必要があります。

図8-1 Oracle9iまたは10gリリース1からOracle Database 11gリリース2以上にデータベースをアップグレードする場合のSQLパフォーマンス・アナライザのワークフロー



データベースのアップグレードをテストする前に、次の条件が満たされていることを確認します。

- アップグレード元となる本番システムでは、Oracle9i (9.2.0.8)またはOracle Database 10gリリース1 (10.1.0.5)を実行しています。
- アップグレードするテスト・システムでは、Oracle Database 11gリリース2以上のリリースを実行しています。

  データベースのバージョンは、リリース11.2.0.3以上にできます。
- 両システムのパフォーマンスが相互に比較されるため、テスト・システムは本番システムとできるかぎり同じにする必要があります。
- 両方のシステムのハードウェア構成は可能なかぎり同じにする必要があります。

また、Oracle Database 12cリリース2を実行する別のSQLパフォーマンス・アナライザ・システムを設定する必要があります。このシステムを使用して、SQLチューニング・セットを作成し、SQLパフォーマンス・アナライザを実行します。このシステムでは、本番システムのSQLトレース・ファイルに格納されている統計を使用してSQLチューニング・セットが作成されるため、本番データおよびスキーマを使用できる必要はありません。SQLパフォーマンス・アナライザのタスクがテスト・システム上でリモートに実行され、指定したデータベース・リンクを介してSQL試行に対する実行計画および実行統計が生成されます。データベース・リンクは、DBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続するパブリック・データベース・リンクである必要があります。また、テスト・システム上のユーザーのスキーマから既存のPLAN_TABLEを削除する必要もあります。

前述のとおりアップグレード環境を構成したら、Oracle9iまたはOracle Database 10gリリース1からそれ以上のリリースへのデータベースのアップグレードでSQLパフォーマンス・アナライザを使用するために、次に説明するステップを実行します。

1. 本番システムでSQLトレース機能を有効にします。詳細は、「本番システムでのSQLトレースの有効化」を参照してください。

   本番システムへのパフォーマンスの影響を最小限に抑えながら、SQL文の典型的なセットを完全に取得する場合は、必要な期間セッションのサブセットに対してのみSQLトレースを有効にして、すべての重要なSQL文を1回以上取得す

ることを検討してください。

2. 本番システムでマッピング表を作成します。詳細は、「マッピング表の作成」を参照してください。

   このマッピング表は、SQLトレース・ファイルのユーザーおよびオブジェクトの識別子番号を、同等の文字列に変換するために使用します。

3. SQLトレース・ファイルおよびマッピング表を本番システムからSQLパフォーマンス・アナライザのシステムに移動します。詳細は、「マッピング表の作成」を参照してください。

4. SQLパフォーマンス・アナライザのシステムで、SQLトレース・ファイルを使用してSQLチューニング・セットを作成します。詳細は、「SQLチューニング・セットの作成」を参照してください。

   SQLチューニング・セットには、SQLトレース・ファイルで取得されたSQL文がそれらのSQL文に関連する実行コンテキストおよび実行統計とともに含まれます。

5. SQLパフォーマンス・アナライザ・システムでは、SQLパフォーマンス・アナライザを使用して、SQLパフォーマンス・アナライザのタスクを作成し、SQLチューニング・セットの内容を、比較のベースラインとして使用するアップグレード前のSQL試行に変換します。その後、データベース・リンクを介したテスト・システムでSQL文のテスト実行をリモートで行い、アップグレード後のSQL試行を作成します。詳細は、「Oracle9i DatabaseおよびOracle Database 10gリリース1からのデータベースのアップグレードのテスト」を参照してください。

6. SQLパフォーマンスを比較し、パフォーマンスが低下したSQLを修正します。

   SQLパフォーマンス・アナライザでは、アップグレード前のSQL試行時にSQLチューニング・セットから読み取ったSQL文のパフォーマンスと、アップグレード後のSQL試行時にリモートのテスト実行によって取得したSQL文のパフォーマンスが比較されます。SQL文の実行計画またはパフォーマンスの変更内容を確認するためのレポートが生成されます。

   パフォーマンスが低下したSQL文がレポートに示された場合は、パフォーマンスが低下したSQLを修正するためにさらに変更を行うことができます。詳細は、「データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング」を参照してください。

   SQLチューニング・セットの実行およびそのパフォーマンスと以前の実行のパフォーマンスとの比較を繰り返して、分析結果に満足するまで、行った変更をテストします。

本番システムでのSQLトレースの有効化

Oracle9iでは、SQLトレース機能を使用して個々のSQL文のパフォーマンス・データが収集されます。SQLトレースによって生成された情報は、SQLトレース・ファイルに格納されます。SQLパフォーマンス・アナライザでは、これらのファイルに格納されている次の情報が使用されます。

- 解析が行われたSQLテキストおよびユーザー名
- 各実行のバインド値
- CPU時間および経過時間
- 物理読取りおよび論理読取り
- 処理された行数
- 各SQL文の実行計画(SQL文のカーソルがクローズしている場合にのみ取得される)

インスタンスに対してSQLトレースを有効にすることは可能ですが、セッションのサブセットに対してSQLトレースを有効にすることをお薦めします。インスタンスに対してSQLトレース機能を有効にすると、そのインスタンスで実行されたすべてのSQL文のパ

フォーマンス統計がSQLトレース・ファイルに格納されます。このようにSQLトレースを使用すると、パフォーマンスに重大な影響を与えたり、システムのオーバーヘッドの増加、過度なCPU使用率、ディスク領域の不足などを引き起こす可能性があります。トレース・レベルを4に設定して、バインド値および実行計画を取得する必要があります。

Oracle Database 10gリリース1が実行されている本番システムで、DBMS_MONITOR.SESSION_TRACE_ENABLEプロシージャを使用して、SQLトレースを別のセッションで透過的に有効にします。bindsプロシージャ・パラメータをTRUEに設定して(デフォルト値はFALSE)、バインドを明示的に有効にする必要もあります。

SQLトレースを有効にしたら、SQLパフォーマンス・アナライザで使用するSQL文の典型的なセットの統計が含まれているSQLトレース・ファイルを特定します。その後、SQLトレース・ファイルをSQLパフォーマンス・アナライザのシステムにコピーします。SQLワークロードをSQLトレース・ファイルに取得したら、本番システムでSQL試行を無効にします。

関連項目:

- SQLトレース・ファイルを管理するための初期化パラメータの設定など、SQLトレースを使用する際のその他の考慮事項は、『Oracle Database SQLチューニング・ガイド』を参照してください
- DBMS_MONITORパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

マッピング表の作成

SQLトレース・ファイルに格納されているユーザーおよびオブジェクトの識別子番号を個別の名前に変換するには、それぞれのマッピングを指定する表を用意する必要があります。SQLパフォーマンス・アナライザのシステムでは、トレース・ファイルがSQLチューニング・セットに変換されるときにこのマッピング表が読み取られます。

マッピング表を作成するには:

- 本番データベースで次のSQL文を実行します。

```
CREATE TABLE mapping AS
    SELECT object_id id, owner, SUBSTR(object_name, 1, 30) name FROM dba_objects
    WHERE object_type NOT IN ('CONSUMER GROUP', 'EVALUATION CONTEXT', 'FUNCTION',
                              'INDEXTYPE', 'JAVA CLASS', 'JAVA DATA',
                              'JAVA RESOURCE', 'LIBRARY', 'LOB', 'OPERATOR',
                              'PACKAGE', 'PACKAGE BODY', 'PROCEDURE', 'QUEUE',
                              'RESOURCE PLAN', 'SYNONYM', 'TRIGGER', 'TYPE',
                              'TYPE BODY')
    UNION ALL
    SELECT user_id id, username owner, null name FROM dba_users;
```

マッピング表が作成されたら、データ・ポンプを使用してSQLパフォーマンス・アナライザのシステムに転送できます。

関連項目:

- データ・ポンプの使用の詳細は、Oracle Databaseユーティリティを参照

SQLチューニング・セットの作成

SQLパフォーマンス・アナライザのシステムにSQLトレース・ファイルおよびマッピング表が移動されたら、DBMS_SQLTUNEパッケージを使用してSQLチューニング・セットを作成できます。

SQLチューニング・セットを作成するには:

1. SQLパフォーマンス・アナライザのシステム上のディレクトリにSQLトレース・ファイルをコピーします。
2. このディレクトリのディレクトリ・オブジェクトを作成します。
3. DBMS_SQLTUNE.SELECT_SQL_TRACEファンクションを使用して、SQLトレース・ファイルからSQL文を読み取ります。

   各SQL文に対して、単一実行の情報のみが収集されます。各SQL文の実行頻度は取得されません。そのため、Oracle Database 10gリリース1以前のリリースが実行されている本番システムの比較分析を実行する場合は、SQLパフォーマンス・アナライザ・レポートのワークロード・レベルの統計を無視して、実行レベルでのパフォーマンスの変更のみを評価する必要があります。

   次の例では、sql_trace_prodディレクトリ・オブジェクトに格納されているSQLトレース・ファイルのコンテンツを読み取ってSQLチューニング・セットにロードします。

```
DECLARE
  cur sys_refcursor;
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('my_sts_9i');
  OPEN cur FOR
    SELECT VALUE (P)
    FROM table(DBMS_SQLTUNE.SELECT_SQL_TRACE('sql_trace_prod', '%ora%')) P;
  DBMS_SQLTUNE.LOAD_SQLSET('my_sts_9i', cur);
  CLOSE cur;
END;
/
```

SELECT_SQL_TRACEファンクションの構文は、次のようになります。

```
DBMS_SQLTUNE.SELECT_SQL_TRACE (
  directory               IN VARCHAR2,
  file_name               IN VARCHAR2 := NULL,
  mapping_table_name      IN VARCHAR2 := NULL,
  mapping_table_owner     IN VARCHAR2 := NULL,
  select_mode             IN POSITIVE := SINGLE_EXECUTION,
  options                 IN BINARY_INTEGER := LIMITED_COMMAND_TYPE,
  pattern_start           IN VARCHAR2 := NULL,
  parttern_end            IN VARCHAR2 := NULL,
  result_limit            IN POSITIVE := NULL)
RETURN sys.sqlset PIPELINED;
```

表8-1に、SELECT_SQL_TRACEファンクションで使用できるパラメータを示します。

表8-1 DBMS_SQLTUNE.SELECT_SQL_TRACEファンクションのパラメータ

パラメータ	説明
directory	SQL トレース・ファイルが格納されるディレクトリを指すディレクトリ・オブジェクトを指定し

パラメータ	説明
	ます。
file_name	処理する SQL トレース・ファイルの名前の全体または一部を指定します。指定しなかった場合、特定のディレクトリにある現在または最近のトレース・ファイルが使用されます。トレース・ファイル名の照合には、%ワイルドカードを使用できます。
mapping_table_name	マッピング表の名前を指定します。デフォルト値の NULL に設定されている場合、現在のデータベースからのマッピングが使用されます。マッピング表の名前は、大/小文字が区別されないことに注意してください。
mapping_table_owner	マッピング表が存在するスキーマを指定します。NULL に設定すると、現在のスキーマが使用されます。
select_mode	トレース・ファイルから SQL 文を選択するためのモードを指定します。デフォルト値は SINGLE_EXECUTION です。このモードでは、SQL 文ごとに 1 回の実行の統計のみが SQL チューニング・セットにロードされます。統計は、SQL チューニング・セットの他のデータソースのテーブル・ファンクションの場合と同様に累積されません。
options	操作のオプションを指定します。デフォルト値は LIMITED_COMMAND_TYPE で、SQL トレース・ファイルから戻されるのは、SQL パフォーマンス・アナライザにとって意味のある SQL タイプ(SELECT、INSERT、UPDATE、DELETE など)のみです。
pattern_start	対象とするトレース・ファイル・セクションの開始区切りパターンを指定します。このパラメータは、現在使用されていません。
pattern_end	処理するトレース・ファイル・セクションの終了区切りパターンを指定します。このパラメータは、現在使用されていません。
result_limit	(除外された)ソースの上位 SQL を指定します。デフォルト値は 2^{31} で、これは実質的に無制限であることを表します。

関連項目:

- DBMS_SQLTUNEパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照

Oracle9i DatabaseおよびOracle Database 10gリリース1からのデータベース・アップグレードのテスト

SQLチューニング・セットが作成されたら、SQLパフォーマンス・アナライザを使用して、SQLチューニング・セット内の実行計画および実行時の統計からアップグレード前のSQL試行を作成できます。変更前のSQL試行が作成されたら、テスト・システムでSQLチューニング・セット内のSQL文のテスト実行または計画の生成を行い、アップグレード後のSQL試行を作成する必要があります。SQLパフォーマンス・アナライザでは、テスト・システムにリモート接続し、SQL試行の実行計画と統計を生成することによって、指定したパブリック・データベース・リンクを使用してSQL文をテスト実行します。データベース・リンクは、SQLパフォーマンス・アナライザ・システムに存在し、テスト・システムでSQLチューニング・セットを実行する権限を持つリモート・ユーザーに接続されている必要があります。

SQLパフォーマンス・アナライザを実行し、Oracle Enterprise ManagerまたはAPIを使用して、Oracle9i DatabaseまたはOracle Database 10gリリース1からのデータベースのアップグレードをテストすることができます。詳細は、次の項を参照してください。

- Cloud Controlを使用したリリース9.xおよび10.1からのデータベース・アップグレードのテスト
- APIを使用したリリース9.xおよび10.1からのデータベースのアップグレードのテスト

Cloud Controlを使用したリリース9.xおよび10.1からのデータベース・アップグレードのテスト

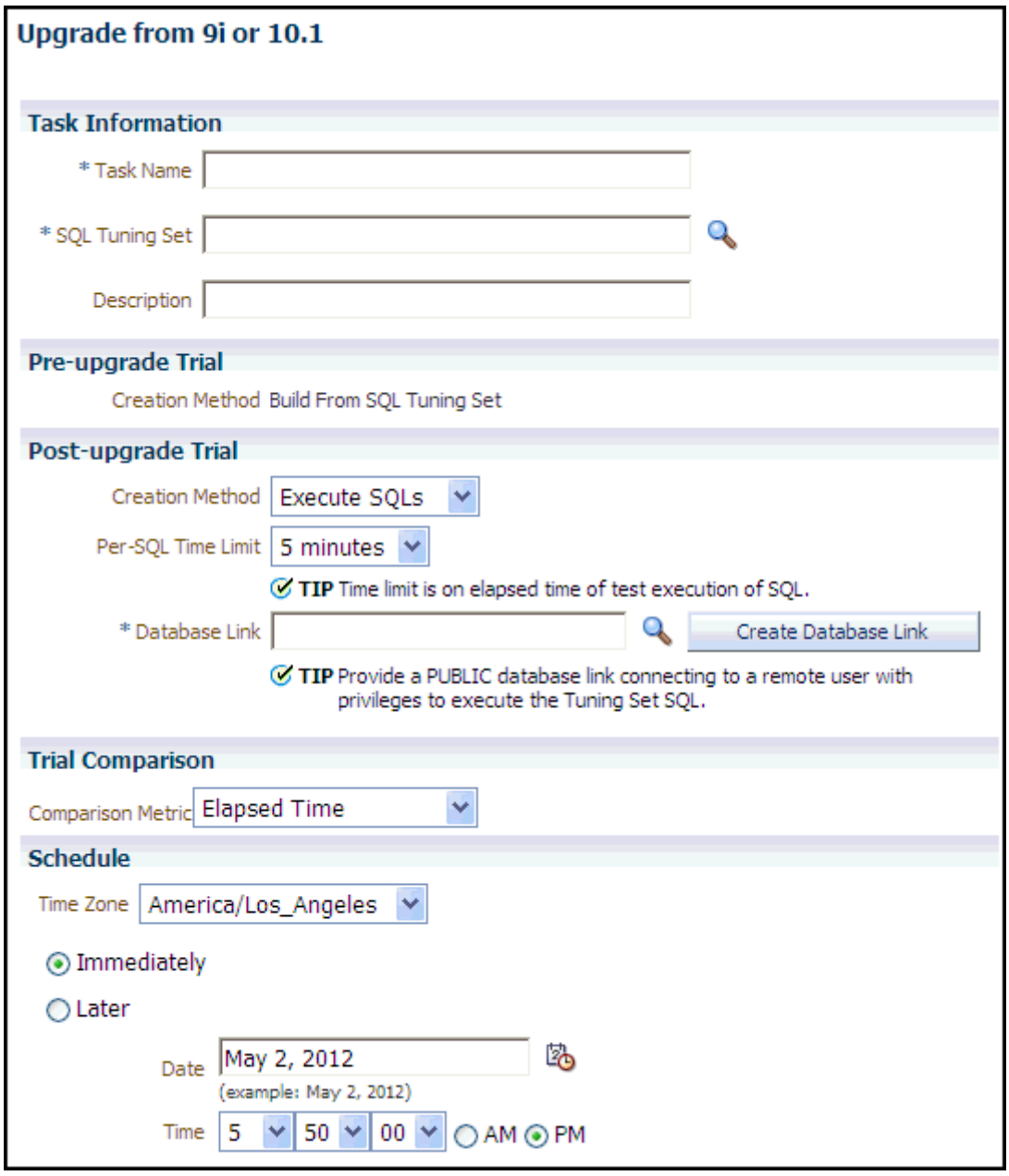
SQLパフォーマンス・アナライザを使用して、Oracle9i DatabaseまたはOracle Database 10g リリース1からデータベースのアップグレードをテストするには:

1. 「パフォーマンス」メニューから「SQL」を選択し、「SQLパフォーマンス・アナライザ」を選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。
2. SQLパフォーマンス・アナライザのワークフローで、「9iまたは10.1からのアップグレード」をクリックします。

   9i以上からのアップグレードページが表示されます。



3. 「タスク情報」で、次のように指定します。

   a. 「タスク名」フィールドに、タスクの名前を入力します。

   b. 「SQLチューニング・セット」フィールドに、作成されたSQLチューニング・セットの名前を入力します。

      または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを検索します。

      選択したSQLチューニング・セットが「SQLチューニング・セット」フィールドに表示されます。

   c. 「説明」フィールドに、オプションでタスクの説明を入力します。

4. 「作成方法」フォールドで、次のように選択します。

   a. SQLの実行: 実際にパブリック・データベース・リンクを介してテスト・システムでSQL文をリモートに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と統計の両方を生成します。

   b. 計画の生成: 実際にSQL文が実行されることなく、パブリック・データベース・リンクを介してテスト・システムで実行計画がリモートに作成されます。

5. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定しま

す。

a. 「5分」を選択します。

この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。

b. 「無制限」を選択します。

この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。

c. 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。

6. 「データベース・リンク」フィールドに、DBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名を入力します。

あるいは、検索アイコンをクリックしてデータベース・リンクを検索して選択するか、または「データベース・リンクの作成」ページで「データベース・リンクの作成」をクリックしてデータベース・リンクを作成します。

7. 「比較メトリック」リストで、比較分析に使用する比較メトリックを選択します。

a. 経過時間

b. CPU時間

c. ユーザーI/O時間

d. バッファ読取り

e. 物理I/O

f. オプティマイザ・コスト

g. I/Oインターコネクト・バイト

SQL試行で実行計画のみを生成した場合に選択可能な比較メトリックは「オプティマイザ・コスト」のみです。

複数の比較メトリックを使用して比較分析を実行するには、異なるメトリックを使用してこの手順を繰り返すことによって比較分析を別々に実行します。

8. 「スケジュール」で、次の項目を選択します。

a. 「タイムゾーン」リストで、タイムゾーン・コードを選択します。

b. 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。

9. 「発行」をクリックします。

「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。

「SQLパフォーマンス・アナライザのタスク」セクションに、このタスクのステータスが表示されます。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。タスクが完了すると、「ステータス」フィールドが「完了」に変更されます。

10. 「SQLパフォーマンス・アナライザのタスク」で、タスクを選択して「名前」列のリンクをクリックします。

「SQLパフォーマンス・アナライザのタスク」ページが表示されます。

このページには、次のセクションが含まれています。

a. SQLチューニング・セット

このセクションには、SQLチューニング・セットに関する情報(名前、所有者、説明、SQLチューニング・セットに含まれているSQL文の数など)の概要が表示されます。

b. SQL試行

このセクションには、SQLパフォーマンス・アナライザのタスクで使用されるSQL試行を示す表が含まれています。

c. SQL試行比較

このセクションには、SQL試行比較の結果を示す表が含まれています。

11. 「比較レポート」列のアイコンをクリックします。

「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。

12. パフォーマンス分析の結果を確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

データベースのアップグレード後にパフォーマンスが低下したSQL文が検出された場合は、「データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング」に従って、それらのSQL文をチューニングします。

APIを使用したリリース9.xおよび10.1からのデータベースのアップグレードのテスト

この項では、APIを使用してOracle Databaseリリース9.xおよび10.1からのデータベースのアップグレードをテストする方法を説明します。

リリース9.xおよび10.1からのデータベースのアップグレードをテストするには:

1. SQLパフォーマンス・アナライザを実行しているシステムで、分析タスクを作成します。
2. 次のパラメータを使用して、EXECUTE_ANALYSIS_TASKプロシージャを呼び出すことにより、SQLチューニング・セット内の実行計画および実行時の統計からアップグレード前のSQL試行を構築します。
   - task_nameパラメータを、実行するSQLパフォーマンス・アナライザのタスクの名前に設定します。
   - execution_typeパラメータをCONVERT SQLSETに設定し、SQLチューニング・セット内の統計を試行実行として処理するようにSQLパフォーマンス・アナライザに指示します。
   - execution_nameパラメータを使用して、実行を識別するための名前を指定します。指定しなかった場合、SQLパフォーマンス・アナライザによってタスク実行の名前が自動的に生成されます。

   次の例では、my_spa_taskというSQLパフォーマンス・アナライザのタスクを試行実行として実行します。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'CONVERT SQLSET', -
       execution_name => 'my_trial_9i');
```

3. EXECUTE_ANALYSIS_TASKプロシージャを使用して、実行計画またはテスト実行を行い、アップグレード後のSQL試行を構築します。
   - execution_typeパラメータをEXPLAIN PLANまたはTEST EXECUTEに設定します。
     - EXPLAIN PLANを使用することを選択した場合、実行計画のみが生成されます。その後の比較では、パフォーマンスの変更に関して結論を出さずに、変更済の計画のリストを生成することのみが可能に

なります。

- TEST EXECUTEを使用することを選択した場合、SQLワークロードは完了するまで実行されます。これにより、テスト・システムから生成された統計と実行計画を使用して、アップグレード後のSQL試行が効果的に作成されます。ソースでSQL実行計画およびパフォーマンス・データを取得する場合は、より正確な分析を行えるように、TEST EXECUTEを使用することをお薦めします。

- DATABASE_LINKタスク・パラメータをDBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名に設定します。

次の例では、データベース・リンクを介してSQL文のテスト実行をリモートで行います。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_10g', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

関連項目:

- 「APIを使用した分析タスクの作成」
- 「APIを使用した変更前のSQL試行の作成」
- 「APIを使用した変更後のSQL試行の作成」
- DBMS_SQLPA.EXECUTE_ANALYSIS_TASKファンクションについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

Oracle Database 10gリリース2以上のリリースからのアップグレード

SQLパフォーマンス・アナライザを使用すると、本番システムでSQLチューニング・セットを取得し、データベース・リンクを介してSQLチューニング・セットをリモートで2回実行する(まず変更前のSQL試行を作成し、次に変更後のSQL試行を再度作成する)ことによって、Oracle Database 10gリリース2以上のリリースからそれ以上のリリースへのデータベースのアップグレードのSQLレスポンス時間に対する影響をテストできます。

データベースのアップグレードをテストする前に、次の条件が満たされていることを確認します。

- アップグレードする本番システムで、Oracle Database 10gリリース2以上のリリースを実行しています。
- 最初、テスト・システムも同じリリースのOracle Databaseを実行している必要があります。
- テスト・システムには、本番システムに存在するものと完全に同一な本番データのコピーがある必要があります。
- 可能なかぎり本番システムと同じハードウェア構成にする必要があります。

また、Oracle Database 11gリリース2を実行する別のSQLパフォーマンス・アナライザ・システムを設定する必要があります。このシステムを使用して、SQLパフォーマンス・アナライザを実行します。このシステムでは、本番システムのSQLトレース・ファイルに格納されている統計を使用してSQLチューニング・セットが作成されるため、本番データおよびスキーマを使用できる必要はありません。SQLパフォーマンス・アナライザのタスクがテスト・システム上でリモートに実行され、指定したデータベース・リンクを介してSQL試行に対する実行計画および実行統計が生成されます。データベース・リンクは、DBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続するパブリック・データベース・リンクである必要があります。また、

テスト・システム上のユーザーのスキーマから既存のPLAN_TABLEを削除する必要もあります。

前述のとおりアップグレード環境を構成したら、Oracle Database 10g リリース2以上のリリースから以降のリリースへのデータベースのアップグレードでSQLパフォーマンス・アナライザを使用するために、次に説明するステップを実行します。

1. 本番システムで、分析対象のSQLワークロードを取得して、SQLチューニング・セットに格納します。詳細は、「SQLワークロードの取得」を参照してください。
2. 可能なかぎり本番環境と一致するようにテスト・システムを設定します。詳細は、「テスト・システムの設定」を参照してください。
3. SQLチューニングセットをSQLパフォーマンス・アナライザのシステムに転送します。

   SQLチューニング・セットの転送については、使用するツールに応じて次のマニュアルを参照してください。

   - Oracle Enterprise Managerを使用する場合は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
   - APIを使用する場合は、『Oracle Database SQLチューニング・ガイド』を参照してください
4. SQLパフォーマンス・アナライザで、SQLチューニング・セットを入力ソースとして使用してSQLパフォーマンス・アナライザのタスクを作成します。

   テスト・システムで、データベース・リンクを介してSQLチューニング・セットのSQL文をリモートでテスト実行し、比較用のベースラインとして使用するアップグレード前のSQL試行を作成します。詳細は、「Oracle Database 10gリリース2以上のリリースからのデータベースのアップグレードのテスト」を参照してください。
5. テスト・システムをアップグレードします。
6. アップグレードしたテスト・システムで、データベース・リンクを介してSQL文をリモートで2回テスト実行し、アップグレード後のSQL試行を作成します。詳細は、「Oracle Database 10gリリース2以上のリリースからのデータベースのアップグレードのテスト」を参照してください。
7. SQLパフォーマンスを比較し、パフォーマンスが低下したSQLを修正します。

   SQLパフォーマンス・アナライザでは、アップグレード前のSQL試行時にSQLチューニング・セットから読み取ったSQL文のパフォーマンスと、アップグレード後のSQL試行時にリモートのテスト実行によって取得したSQL文のパフォーマンスが比較されます。SQL文の実行計画またはパフォーマンスの変更内容を確認するためのレポートが生成されます。

   パフォーマンスが低下したSQL文がレポートに示された場合は、パフォーマンスが低下したSQLを修正するためにさらに変更を行うことができます。詳細は、「データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング」を参照してください。

   SQLチューニング・セットの実行およびそのパフォーマンスと以前の実行のパフォーマンスとの比較を繰り返して、分析結果に満足するまで、行った変更をテストします。

Oracle Database 10gリリース2以上のリリースからのデータベース・アップグレードのテスト

SQLチューニング・セットをSQLパフォーマンス・アナライザ・システムに転送すると、SQLパフォーマンス・アナライザを使用して、テスト・システムでSQLチューニング・セット内のSQL文の計画を実行または生成し、アップグレード前の SQL試行を作成できます。SQLパフォーマンス・アナライザでは、テスト・システムにリモート接続し、SQL試行の実行計画と統計を生成することによって、指定したデータベース・リンクを使用してSQL文をテスト実行します。データベース・リンクは、SQLパフォーマンス・アナライザ・システムに存在し、テスト・システムでSQLチューニング・セットを実行する権限を持つリモート・ユーザーに接続されている必要があります。

SQL試行が作成されたら、テスト・システムをアップグレードする必要があります。データベースがアップグレードされたら、アップグレードしたテスト・システムで、SQLチューニング・セットに含まれるSQL文の2回目の実行または計画の生成を行い、アップグレード後の SQL試行を作成します。ハードウェア・リソースが使用可能な場合は、別の方法として、アップグレードした別のテスト・システムを使用して、2回目のリモートSQL試行を実行することもできます。この方法は、SQLパフォーマンス・アナライザによって特定された問題を調査する場合に役に立つことがあります。

SQLパフォーマンス・アナライザを実行し、Oracle Enterprise ManagerまたはAPIを使用し、Oracle Database 10g リリース2以上のリリースからのデータベースのアップグレードをテストすることができます。詳細は、次の項を参照してください。

- Cloud Controlを使用したリリース10.2以上からのデータベース・アップグレードのテスト
- 「APIを使用したリリース10.2以上からのデータベースのアップグレードのテスト」

Cloud Controlを使用したリリース10.2以上からのデータベース・アップグレードのテスト

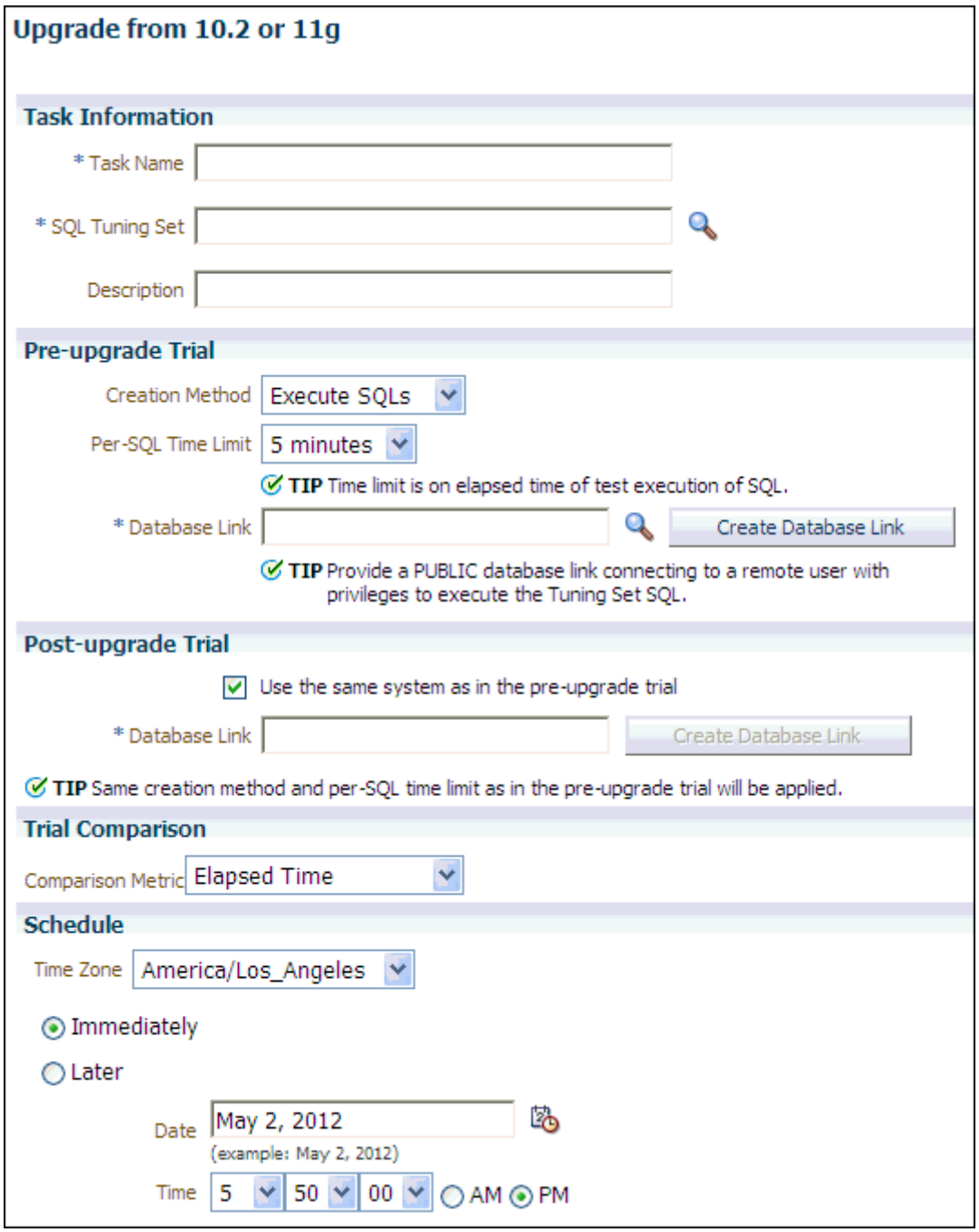
SQLパフォーマンス・アナライザを使用して、Oracle Database 10gリリース以上のリリースからのデータベースのアップグレードをテストするには:

1. 「パフォーマンス」メニューから「SQL」を選択し、「SQLパフォーマンス・アナライザ」を選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。
2. SQLパフォーマンス・アナライザのワークフローで、「10.2または11gからのアップグレード」をクリックします。

   「10.2以上のリリースからのアップグレード」ページが表示されます。



3. 「タスク情報」で、次のように指定します。

   a. 「タスク名」フィールドに、タスクの名前を入力します。

   b. 「SQLチューニング・セット」フィールドに、作成されたSQLチューニング・セットの名前を入力します。

      または、検索アイコンをクリックして、「検索と選択: SQLチューニング・セット」ウィンドウでSQLチューニング・セットを検索します。

      選択したSQLチューニング・セットが「SQLチューニング・セット」フィールドに表示されます。

   c. 「説明」フィールドに、オプションでタスクの説明を入力します。

4. 「作成方法」フォールドで、次のように選択します。

   a. SQLの実行: 実際にパブリック・データベース・リンクを介してテスト・システムでSQL文をリモートに実行することによって、SQLチューニング・セット内の各SQL文に対して実行計画と統計の両方を生成します。

   b. 計画の生成: 実際にSQL文が実行されることなく、パブリック・データベース・リンクを介してテスト・システムで

実行計画がリモートに作成されます。

5. 「SQL当たりの時間制限」リストで、次のいずれかのアクションを実行して、試行時のSQL実行の時間制限を決定します。

   a. 「5分」を選択します。

      この実行では、SQLチューニング・セット内の各SQL文が最大5分間実行され、パフォーマンス・データが収集されます。

   b. 「無制限」を選択します。

      この実行では、SQLチューニング・セット内の各SQL文が完了するまで実行され、パフォーマンス・データが収集されます。実行統計を収集することによってパフォーマンス分析の精度は大幅に向上しますが、分析にかかる時間は長くなります。1つのSQL文によってタスクが長時間停止状態になる場合があるため、この設定は使用しないことをお薦めします。

   c. 「カスタマイズ」を選択して、指定する秒数、分数、時間数を入力します。

6. 「データベース・リンク」フィールドに、DBMS_SQLPAパッケージのEXECUTE権限およびアップグレード前のシステムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名を入力します。

   あるいは、検索アイコンをクリックしてデータベース・リンクを検索して選択するか、または「データベース・リンクの作成」ページで「データベース・リンクの作成」をクリックしてデータベース・リンクを作成します。

7. 「アップグレード後の試行」で、次のように指定します。

   a. アップグレード前の試行とアップグレード後の試行の実行に同じシステムを使用するように、「アップグレード前の試行時と同じシステムを使用」を選択します。

      システム構成が異なることによって発生する可能性のあるエラーを回避するため、このオプションを使用することをお薦めします。このオプションを使用する場合は、テスト・データベースを上位データベースにアップグレードしてから、アップグレード後の試行を実行する必要があります。

   b. 「データベース・リンク」フィールドに、DBMS_SQLPAパッケージのEXECUTE権限およびアップグレード後のシステムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名を入力します。

8. 「比較メトリック」リストで、比較分析に使用する比較メトリックを選択します。

   a. 経過時間

   b. CPU時間

   c. ユーザーI/O時間

   d. バッファ読取り

   e. 物理I/O

   f. オプティマイザ・コスト

   g. I/Oインターコネクト・バイト

   SQL試行で実行計画のみを生成した場合に選択可能な比較メトリックは「オプティマイザ・コスト」のみです。

   複数の比較メトリックを使用して比較分析を実行するには、異なるメトリックを使用してこの手順を繰り返すことによって比較分析を別々に実行します。

9. 「スケジュール」で、次の項目を選択します。

    a. 「タイムゾーン」リストで、タイムゾーン・コードを選択します。

    b. 「即時」(即時にタスクを開始する場合)または、「後で」(「日付」および「時間」フィールドで指定した時間にタスクを開始するようスケジュールする場合)を選択します。

10. 「発行」をクリックします。

「SQLパフォーマンス・アナライザ・ホーム」ページが表示されます。

「SQLパフォーマンス・アナライザのタスク」セクションに、このタスクのステータスが表示されます。ステータス・アイコンをリフレッシュするには、「リフレッシュ」をクリックします。

アップグレード前の試行とアップグレード後の試行の実行に同じシステムを使用している場合は、アップグレード前の試行のステップを完了した後で、データベースをアップグレードする必要があります。データベースをアップグレードした後、アップグレード後の試行を実行できます。タスクが完了すると、「ステータス」フィールドが「完了」に変更されます。

11. 「SQLパフォーマンス・アナライザのタスク」で、タスクを選択して「名前」列のリンクをクリックします。

「SQLパフォーマンス・アナライザのタスク」ページが表示されます。

このページには、次のセクションが含まれています。

    a. SQLチューニング・セット

        このセクションには、SQLチューニング・セットに関する情報(名前、所有者、説明、SQLチューニング・セットに含まれているSQL文の数など)の概要が表示されます。

    b. SQL試行

        このセクションには、SQLパフォーマンス・アナライザのタスクで使用されるSQL試行を示す表が含まれています。

    c. SQL試行比較

        このセクションには、SQL試行比較の結果を示す表が含まれています。

12. 「比較レポート」列のアイコンをクリックします。

「SQLパフォーマンス・アナライザのタスク結果」ページが表示されます。

13. パフォーマンス分析の結果を確認します。詳細は、「Oracle Enterprise Managerを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください。

データベースのアップグレード後にパフォーマンスが低下したSQL文が検出された場合は、「データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング」に従って、それらのSQL文をチューニングします。

「APIを使用したリリース10.2以上からのデータベースのアップグレードのテスト」

この項では、APIを使用してOracle Databaseリリース10.2以上のデータベース・アップグレードをテストする方法について説明します。

リリース10.2以上のデータベースのアップグレードをテストするには:

1. SQLパフォーマンス・アナライザを実行しているシステムで、分析タスクを作成します。
2. SQLチューニング・セット内のSQL文の実行計画またはテスト実行を行い、アップグレード前のSQL試行を構築します。

    次のパラメータを使用して、EXECUTE_ANALYSIS_TASKプロシージャをコールします。

- task_nameパラメータを、実行するSQLパフォーマンス・アナライザのタスクの名前に設定します。
- execution_typeパラメータをEXPLAIN PLANまたはTEST EXECUTEに設定します。
  - EXPLAIN PLANを使用することを選択した場合、実行計画のみが生成されます。その後の比較では、パフォーマンスの変更に関して結論を出さずに、変更済の計画のリストを生成することのみが可能になります。
  - TEST EXECUTEを使用することを選択した場合、SQLワークロードは完了するまで実行されます。これにより、テスト・システムから生成された統計と実行計画を使用して、アップグレード前のSQL試行が効果的に作成されます。ソースでSQL実行計画およびパフォーマンス・データを取得する場合は、より正確な分析を行えるように、TEST EXECUTEを使用することをお薦めします。
- execution_nameパラメータを使用して、実行を識別するための名前を指定します。指定しなかった場合、SQLパフォーマンス・アナライザによってタスク実行の名前が自動的に生成されます。
- DATABASE_LINKタスク・パラメータをDBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名に設定します。

次の例では、my_spa_taskというSQLパフォーマンス・アナライザのタスクを実行し、データベース・リンクを介してSQL文のテスト実行をリモートで行います。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_10g', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

3. EXECUTE_ANALYSIS_TASKプロシージャを使用して、実行計画またはテスト実行を行い、アップグレード後のSQL試行を構築します。
   - execution_typeパラメータをEXPLAIN PLANまたはTEST EXECUTEに設定します。
     - EXPLAIN PLANを使用することを選択した場合、実行計画のみが生成されます。その後の比較では、パフォーマンスの変更に関して結論を出さずに、変更済の計画のリストを生成することのみが可能になります。
     - TEST EXECUTEを使用することを選択した場合、SQLワークロードは完了するまで実行されます。これにより、テスト・システムから生成された統計と実行計画を使用して、アップグレード後のSQL試行が効果的に作成されます。ソースでSQL実行計画およびパフォーマンス・データを取得する場合は、より正確な分析を行えるように、TEST EXECUTEを使用することをお薦めします。
   - DATABASE_LINKタスク・パラメータをDBMS_SQLPAパッケージのEXECUTE権限およびテスト・システムのADVISOR権限を持つユーザーに接続しているパブリック・データベース・リンクのグローバル名に設定します。

次の例では、データベース・リンクを介してSQL文のテスト実行をリモートで行います。

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_12c', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

関連項目:

- 「APIを使用した分析タスクの作成」
- 「APIを使用した変更前のSQL試行の作成」
- 「APIを使用した変更後のSQL試行の作成」
- DBMS_SQLPA.EXECUTE_ANALYSIS_TASKファンクションについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

データベースのアップグレードをテストした後のパフォーマンスが低下したSQL文のチューニング

テスト・システムのデータベースをアップグレードした後に、パフォーマンスが低下したSQL文をSQLパフォーマンス・アナライザで特定する場合があります。

SQLチューニング・アドバイザまたはSQL計画ベースラインを使用して(詳細は、「SQL試行の比較」を参照)、パフォーマンスが低下したSQL文をチューニングできます。これには、パフォーマンスが低下したSQL文だけを含むSQLチューニング・セットのサブセットを作成し、このパフォーマンスが低下したSQL文のサブセットをリモート・データベースに転送するためのAPIの使用、およびリモート・データベースでのSQLチューニング・アドバイザの実行が含まれます。

Oracle Enterprise Managerでは、1つ以上のSQL試行を含むSQLパフォーマンス・アナライザを実行した後、パフォーマンスの低下を修正することはできません。

Oracle Database 10gリリース2以上のリリースからアップグレードする場合は、将来、既存の実行計画を選択するようにオプティマイザに指示するためのSQL計画ベースラインを作成することもできます。

関連項目:

- 「APIを使用した、リモートSQL試行からのパフォーマンスが低下したSQL文のチューニング」
- 「APIを使用したSQL計画ベースラインの作成」

第II部 データベース・リプレイ

データベース・リプレイでは、本番環境と同じワークロードをテスト・システムでリプレイし、システムの変更による全体的な影響を評価できます。

第II部は、データベース・リプレイについて説明し、次の章で構成されています。

- データベース・リプレイの概要
- データベース・ワークロードの取得
- データベース・ワークロードの事前処理
- データベース・ワークロードのリプレイ
- 取得およびリプレイ済ワークロードの分析
- ワークロード・インテリジェンスの使用
- データベース統合リプレイの使用
- ワークロード・スケールアップの使用

9 データベース・リプレイの概要

データベース・リプレイを使用すると、本番システムのワークロードを取得して、それを元のワークロードとまったく同じタイミング、同時実行性およびトランザクション特性に従ってテスト・システムでリプレイできます。これによって、本番システムに影響を与えずに、システム変更の影響をテストできます。

データベース・リプレイでは、Oracle Database 10gリリース2以上のリリースが実行されているシステムでのワークロードの取得がサポートされています。Oracle Database 10g リリース2が実行されているシステムでワークロードを取得するには、データベースのバージョンは10.2.0.4以上である必要があります。ワークロードのリプレイは、Oracle Database 11g リリース1以上のリリースが実行されているシステムでのみサポートされています。

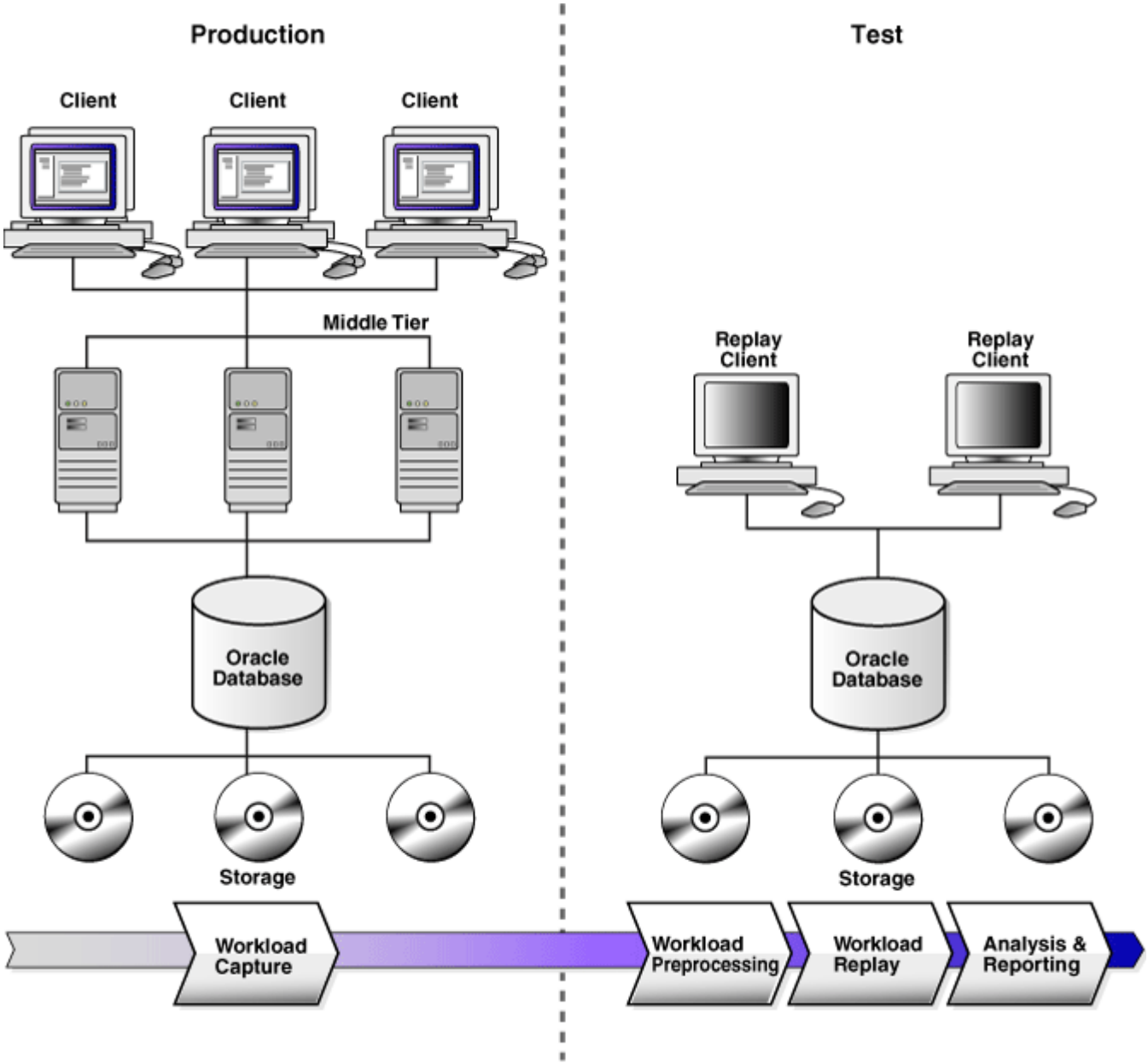
ノート:

Oracle9i Database、または旧バージョンの Oracle Database 10g を実行しているシステムでワークロード取得機能を使用する場合は、次の URL に記載されている My Oracle Support の Note ID 560977.1 で必要なパッチを確認するか、Oracle サポート・サービスに詳細をお問い合せください。

https://support.oracle.com/rs?type=doc&id=560977.1

データベース・リプレイを使用してシステム変更の影響を分析する場合は、図9-1に示すステップを実行します。

図9-1 データベース・リプレイのワークフロー



1. 本番システムで、ワークロードを取得ファイルに取得します。詳細は、「ワークロードの取得」を参照してください。
2. 取得ファイルをテスト・システムにコピーして事前処理します。詳細は、「ワークロードの事前処理」を参照してください。
3. テスト・システムで、事前処理済のファイルをリプレイします。詳細は、「ワークロードのリプレイ」を参照してください。
4. データベース・リプレイで生成されたレポートを使用して、ワークロードの取得およびワークロードのリプレイの両方を詳細に分析します。詳細は、「分析およびレポート」を参照してください。

ワークロードの取得

データベース・リプレイを使用するための最初のステップは、本番ワークロードの取得です。ワークロードの取得には、外部クライアントからOracle Databaseに対するすべてのリクエストの記録が含まれます。

ワークロードの取得を有効にすると、Oracle Databaseに対するすべての外部クライアントのリクエストが追跡され、ファイル・システム上のバイナリ・ファイル(取得ファイル)に格納されます。取得ファイルの格納場所は指定することができます。ワークロードの取得が開始されると、外部データベースのすべてのコールが取得ファイルに書き込まれます。取得ファイルには、クライアント・リクエ

ストに関連するすべての情報(SQLテキスト、バインド値、トランザクション情報など)が含まれます。バックグラウンド・アクティビティおよびデータベース・スケジューラ・ジョブは取得されません。これらの取得ファイルは、プラットフォームに依存しないため、別のシステムに転送できます。

関連項目:

- 本番システムでのワークロードの取得方法の詳細は、「データベース・ワークロードの取得」を参照

ワークロードの事前処理

ワークロードを取得したら、取得ファイルの情報を前処理する必要があります。事前処理により、ワークロードのリプレイに必要なすべてのメタデータが作成されます。事前処理は、取得したワークロードをリプレイする前に、ワークロードごとに1回行う必要があります。事前処理された取得済のワークロードは、同じバージョンのOracle Databaseが稼動するリプレイ・システムで繰り返しリプレイできます。通常、取得ファイルは、前処理のためにテスト・システムにコピーする必要があります。ワークロードの事前処理は、時間がかかり、リソースを大量に消費するため、このステップはワークロードをリプレイするテスト・システムで実行することをお薦めします。

関連項目:

- 取得されたワークロードの事前処理の方法の詳細は、「データベース・ワークロードの事前処理」を参照

ワークロードのリプレイ

事前処理された取得済ワークロードは、テスト・システムでリプレイできます。ワークロードのリプレイ・フェーズにおいて、Oracle Databaseは、取得したすべての外部クライアント・リクエストを本番システムと同じタイミング、同時実行性およびトランザクション依存性に従って再現し、ワークロードの取得フェーズ中に記録されたアクションをテスト・システム上で実行します。

データベース・リプレイでは、リプレイ・クライアントというクライアント・プログラムを使用して、ワークロードの取得時に記録されたすべての外部クライアント・リクエストを再現します。取得したワークロードによっては、そのワークロードを適切にリプレイするために1つ以上のリプレイ・クライアントが必要です。特定のワークロードに必要なリプレイ・クライアントの数を判定するための測定ツールが提供されています。DMLやSQL問合せを含め、ワークロード全体がリプレイされるため、リプレイ・システムのデータは取得システムのデータとできるかぎり論理的に同じである必要があります。これによって、リプレイの相違が最小限に抑えられ、リプレイの分析の信頼性が向上します。

関連項目:

- テスト・システムでの前処理されたワークロードのリプレイ方法の詳細は、「データベース・ワークロードのリプレイ」を参照

分析およびレポート

ワークロードのリプレイ後、ワークロードの取得とリプレイを詳しく分析するために、詳細なレポートが提供されます。

ワークロードの取得レポートとワークロードのリプレイ・レポートには、リプレイ中に発生したエラーや、DMLまたはSQL問合せによっ

て戻された行におけるデータの相違など、ワークロードの取得およびリプレイに関する基本情報が含まれます。ワークロードの取得とワークロードのリプレイを対象とする複数の統計(データベース時間、平均アクティブ・セッション、ユーザー・コールなど)の比較データも提供されます。

リプレイの期間比較レポートは、あるワークロード・リプレイと、その取得、または同じ取得の別のリプレイとの高度な比較を実行するために使用できます。相違の概要として、データの相違が発生しているか、および大幅なパフォーマンスの変化があったかに関する分析も示されます。さらに、これらのレポートには自動データベース診断モニター(ADDM)の結果も取り込まれています。

高度な分析には、自動ワークロード・リポジトリ(AWR)・レポートを利用して、ワークロードの取得とワークロードのリプレイを対象とするパフォーマンス統計を詳細に比較することができます。これらのレポートで参照可能な情報は非常に詳細であり、ワークロードの取得とリプレイにおける相違点を検出できます。ワークロード・インテリジェンスで、ワークロードの取得時に記録されたデータを操作してワークロードを示すモデルを作成することもできます。このモデルを使用すると、ワークロードの一部として実行されるテンプレートの有意なパターンを特定できます。特定のパターンの実行回数やそのパターンの実行に消費されたデータベース時間など重要な統計を、パターンごとに確認できます。

SQLパフォーマンス・アナライザ・レポートは、ワークロード取得のSQLチューニング・セットを、ワークロード・リプレイの別のSQLチューニング・セットと、または2つのワークロード・リプレイの2つのSQLチューニング・セットと比較する場合に使用できます。SQLパフォーマンス・アナライザのテスト実行では、各SQL試行のSQL文ごとに1つの実行計画のみを生成するのに対し、SQLチューニング・セットをデータベース・リプレイと比較すると、各SQL文のすべての実行計画が考慮されて示されるため、SQLパフォーマンス・アナライザのテスト実行よりも詳しい情報が得られます。また、データベース・リプレイではすべてのバインド値が取得され、PL/SQLパッケージの状態などの動的なセッション状態がより正確に再現されるため、SQL文がより信頼度の高い環境で実行されます。データベース・リプレイを使用してロード・テストおよび現行性テストを実行する前に、健全性テストとしてSQLパフォーマンス・アナライザのテスト実行を行い、SQL文が低下しておらず、テスト・システムが適切に設定されていることを確認しておくことをお薦めします。

リプレイの相違情報を使用して、与えられたシステム変更のリプレイ特性を分析することに加えて、アプリケーション・レベルの検証手順を使用して、システム変更を評価する必要もあります。リプレイの全体的な成功を評価するためのスクリプトの作成を検討してください。たとえば、ワークロードの取得時に10,000のオーダーが処理される場合、同じ数のオーダーがリプレイ時にも処理されることを検証する必要があります。

リプレイの分析が完了したら、データベースをワークロード取得時点の元の状態にリストアし、ワークロードのリプレイを繰り返して、システムに対する他の変更をテストすることができます。

関連項目:

- データベース・リプレイ・レポートを使用してデータとパフォーマンスの相違を分析する方法の詳細は、「取得およびリプレイ済ワークロードの分析」を参照してください

PDBでのワークロードの取得とリプレイ

ワークロード取得の有効化とワークロード・リプレイの開始は、プラガブル・データベース(PDB)レベルで実行できます。

Oracle Databaseリリー19cより前のリリースでは、ワークロードの取得とリプレイは、取得とリプレイを開始したコンテナ・データベース(CDB)ルートのCDB管理者専用のものでした。Oracle Databaseリリース19c以降では、現在のプラガブル・データベース(PDB)に対してワークロード取得を有効にできます。現在のPDBに対するワークロード・リプレイも開始できます。

ノート:

ワークロードの同時取得と同時リプレイは、PDB レベルではサポートされていません。

10 データベース・ワークロードの取得

この章では、本番システムでデータベース・ワークロードを取得する方法について説明します。データベース・リプレイを使用するための最初のステップは、本番ワークロードの取得です。

この章の構成は、次のとおりです。

関連項目:

データベース・リプレイのアーキテクチャに適したデータベース・ワークロードの取得方法の詳細は、「ワークロードの取得」を参照してください

データベース・ワークロードの取得の前提条件

ワークロードの取得を開始する前に、テスト・システムでデータベースをリストアするための方針を決定する必要があります。ワークロードをリプレイするには、リプレイ・システムのアプリケーション・データの論理状態を、リプレイの開始時における取得システムのデータの状態と同じにする必要があります。これを行うには、次のいずれかの方法を使用します。

- Recovery Manager(RMAN)のDUPLICATEコマンド
- スナップショット・スタンバイ
- データ・ポンプのインポートおよびエクスポート

これにより、リプレイ・システムのデータベースを、ワークロードの取得の開始時と同じ運用の状態までリストアできます。

データベースがDatabase Vaultにより保護されている場合、データベース・リプレイを使用するには、Database Vault環境においてDBMS_WORKLOAD_CAPTUREおよびDBMS_WORKLOAD_REPLAYパッケージの使用が承認されている必要があります。

関連項目:

- Database Vault環境でのデータベース・リプレイの使用の詳細は、『Oracle Database Vault管理者ガイド』を参照してください。
- RMANを使用したデータベースの複製については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- スナップショット・スタンバイ・データベースの管理の詳細は、Oracle Data Guard概要および管理を参照
- データ・ポンプの使用の詳細は、Oracle Databaseユーティリティを参照

取得ディレクトリの設定

取得したワークロードを格納する場所を決定し、ディレクトリを設定します。ワークロードの取得を開始する前に、ディレクトリが空であり、ワークロードを格納するために十分なディスク領域があることを確認します。ワークロードの取得中にディレクトリのディスク領域が不足すると、取得が停止します。必要なディスク容量を見積もるため、短期間(数分間など)、ワークロードのテスト取得を実行し、完全取得に必要な容量がどれくらいかを推定することができます。潜在的なパフォーマンスの問題を回避するため、ターゲットのリプレイ・ディレクトリが別のファイル・システムにマウントされていることも確認する必要があります。

Oracle RACの場合、共有ファイル・システムの使用を検討してください。また、インスタンスごとに別々の物理ディレクトリを指定する1つの取得ディレクトリ・パスを設定することもできますが、これらの各ディレクトリに作成されたファイルは、単一のディレクトリに統合する必要があります。Enterprise Managerでは、Oracle RACデータベースに対する取得については、共有ファイル・システムで構成されたOracle RACのみをサポートしています。事前処理に使用できるようにするには、(取得ファイルのみでなく)各インスタンスのローカル取得ディレクトリのコンテンツ全体を共有ディレクトリにコピーする必要があります。たとえば、次のように想定します。

- host1およびhost2という名前の2つのデータベース・インスタンスをLinuxのOracle RAC環境で実行します。
- 両方のインスタンスで、/$ORACLE_HOME/rdbms/captureに解決される、CAPDIRという取得ディレクトリ・オブジェクトを使用しています。
- /nfs/rac_captureにある共有ディレクトリを使用しています。

各ホストにログインして次のコマンドを実行する必要があります。

```
cp -r /$ORACLE_HOME/rdbms/capture/* /nfs/rac_capture
```

両方のインスタンスでこの処理を実行すると、/nfs/rac_capture共有ディレクトリで前処理またはマスキングが可能になります。

ワークロードの取得のオプション

正確な取得を行い、別の環境でのリプレイ時に役立つように、ワークロードを取得する前に適切な計画を行う必要があります。

データベース・ワークロードを取得する前に、次のオプションを慎重に検討してください。

- データベースの再起動
- ワークロードの取得時のフィルタの使用

データベースの再起動

このステップは必須ではありませんが、ワークロードの取得の開始前に進行中のトランザクションおよび依存トランザクションを確実に完了またはロールバックするために、その取得前にデータベースを再起動することをお薦めします。取得の開始前にデータベースを再起動しないと、進行中のトランザクションまたはコミット前のトランザクションは、ワークロードに完全に取得されません。進行中のトランザクションは、コールが取得された部分のトランザクションしかリプレイされないため、適切にリプレイされません。これにより、ワークロードのリプレイ時に望ましくないリプレイの相違が発生する可能性があります。不完全なトランザクションに依存する後続のトランザクションでも、リプレイ時にエラーが発生する場合があります。処理量が多いシステムでは、リプレイの相違がある程度あるのは一般的ですが、その場合でも、違いがあるコールがDB時間などの主要な属性でリプレイの大部分を占めていなければ、リプレイを使用して、システム変更に関して意味のある分析を実行できます。

そのため、データベースを再起動する前に、本番データベースを停止する適切なタイミング(システムへの影響が最も少ない時間)について検討するようにしましょう。たとえば、ワークロードの取得を午前8:00から開始しようと思った場合でも、営業時間中にサービスが停止することを避けるためには、その時間帯にデータベースが再起動されないようにする必要があります。そのような場合は、ワークロードの取得開始をもっと早い時間にずらすなどして、なるべく不都合のない時間にデータベースが再起動されるようにしましょう。

データベースを再起動したら、ユーザー・セッションが再接続してワークロードを発行し始める前に、ワークロードの取得を開始することが重要です。そうしないと、そのユーザー・セッションによって実行されたトランザクションは、その後のデータベース・リプレイで適切にリプレイされません(これは、トランザクションのうち、ワークロードの取得を開始した後に実行されたコールの部分しかリプレイされないためです)。この問題を避けるには、SYSユーザーのみにログインとワークロードの取得開始を許可するように、STARTUP RESTRICTを使用してRESTRICTEDモードでデータベースを再起動します。デフォルトでは、ワークロードの取得が開始されると、RESTRICTEDモードのデータベース・インスタンスは自動的にUNRESTRICTEDモードに切り替わり、ワークロードの取得中は通常の操作を続行できます。

ワークロードの取得は、常に1回しか実行できません。Oracle Real Application Clusters(Oracle RAC)構成の場合、ワークロードの取得はデータベース全体を対象に実行されます。いずれかのOracle RACノードで取得を有効にすると、すべてのデータベース・インスタンスのワークロードの取得が開始されます(ワークロードを取得するプロセスは、Oracle RACを認識します)。必須ではありませんが、ワークロードの取得が開始される前にOracle RAC構成のすべてのインスタンスを再起動し、進行中のトランザクションは取得されないようにすることをお薦めします。

ワークロードの取得前にOracle RAC構成のすべてのインスタンスを再起動するには:

1. すべてのインスタンスを停止します。
2. すべてのインスタンスを再起動します。
3. ワークロードの取得を開始します。
4. アプリケーションを接続し、ユーザー・ワークロードを再起動します。

関連項目:

- 起動時のインスタンスへのアクセス制限の詳細は、Oracle Database管理者ガイドを参照

ワークロードの取得時のフィルタの使用

デフォルトでは、すべてのユーザー・セッションがワークロードの取得時に記録されます。ワークロード・フィルタを使用すると、ワークロードの取得時にワークロードに含めるユーザー・セッションと除外するユーザー・セッションを指定できます。包含フィルタおよび除外フィルタという2種類のワークロード・フィルタがあります。ワークロードの取得では、包含フィルタまたは除外フィルタのいずれか一方を使用でき、両方は使用できません。

包含フィルタでは、ワークロードに取得するユーザー・セッションを指定できます。これは、データベース・ワークロードの一部のみを取得する場合に便利です。

除外フィルタでは、ワークロードに取得しないユーザー・セッションを指定できます。これは、インフラストラクチャ(Oracle Enterprise Manager(EM)、Statspackなど)を監視するセッションやテスト・システムですでに実行中のプロセスなど、ワークロードで取得する必要がないセッション・タイプを除外する場合に便利です。たとえば、ワークロードをリプレイするシステムでEMを実行している場合、取得したEMセッションをそのシステムでリプレイすると、ワークロードが重複します。この場合、除外フィルタを使用してEMセッションを除外できます。

ワークロードの取得の制限事項

特定のタイプのユーザー・セッションおよびクライアント・リクエストは、ワークロードに取得されてもデータベース・リプレイによってサポートされない場合があります。これらのセッションおよびリクエストのタイプをワークロードで取得すると、ワークロード・リプレイ中にエラーが発生する場合があります。

次のタイプのユーザー・セッションおよびクライアント・リクエストはデータベース・リプレイによってサポートされません。

- SQL*Loaderなどのユーティリティを使用する、外部ファイルからのデータのダイレクト・パス・ロード
- PL/SQL以外のアドバンスト・キューイング(AQ)
- フラッシュバック問合せ
- Oracle Call Interface(OCI)ベースのオブジェクト・ナビゲーション
- SQL以外のオブジェクト・アクセス
- 分散トランザクション

  取得された分散トランザクションはすべてローカル・トランザクションとしてリプレイされます。
- XAトランザクション

  XAトランザクションは取得もリプレイもされません。すべてのローカル・トランザクションは取得されます。
- JAVA_XAトランザクション

  ワークロードでJAVA_XAパッケージが使用される場合、JAVA_XAファンクション・コールおよびプロシージャ・コールは通常のPL/SQLワークロードとして取得されます。ワークロード・リプレイ中の問題を回避するには、リプレイを正常に完了できるように、リプレイ・システム上のJAVA_XAパッケージの削除を検討してください。
- データベース常駐接続プーリング(DRCP)
- OUTバインドを使用したワークロード
- 同期モードをOBJECT_IDに設定したマルチスレッド・サーバー(MTS)および共有サーバーのセッション

- 移行されたセッション

  移行されたセッションのワークロードは取得されます。ただし、ユーザーのログインやセッションの移行操作は取得されません。ユーザー・ログインやセッションの移行が有効でない場合、ワークロードが不正なユーザーによってリプレイされる可能性があるため、リプレイでエラーが発生する可能性があります。

通常、データベース・リプレイでは、これらのタイプのサポートされていないユーザー・セッションおよびクライアント・リクエストの取得は行われません。取得された場合でも、データベース・リプレイによってリプレイされることはありません。そのため、サポートされていないユーザー・セッションおよびクライアント・リクエストを手動で除外する必要は、通常はありません。これらが取得され、リプレイ中にエラーが発生することがわかった場合は、ワークロードからこれらを除外するワークロード取得フィルタの使用を検討してください。

関連項目:

- ワークロード取得フィルタの使用の詳細は、「ワークロード取得でのフィルタの使用」を参照してください
- ワークロード・リプレイ・フィルタの使用の詳細は、「ワークロード・リプレイでのフィルタの使用」を参照してください

ワークロードの取得機能の有効化および無効化

データベース・リプレイは、Oracle Database 11g 以上のリリースへのデータベースのアップグレードをテストするために使用できる、Oracle Database 10g リリース2を実行中のシステムにおけるデータベース・ワークロードの取得をサポートします。Oracle Database 10gリリース2(10.2)の場合、ワークロードの取得機能はデフォルトでは有効になっていません。PRE_11G_ENABLE_CAPTURE初期化パラメータを指定することで、この機能を有効または無効にすることができます。

ノート:

Oracle Database 10g リリース 2 が実行されているシステムでデータベース・ワークロードを取得する場合は、ワークロードの取得機能の有効化のみが必要です。

Oracle Database 11g リリース 1 以上が実行されているシステムでデータベース・ワークロードを取得する場合は、ワークロードの取得機能はデフォルトで有効になっているため、この機能を有効にする必要はありません。また、PRE_11G_ENABLE_CAPTURE 初期化パラメータは Oracle Database 10g リリース 2(10.2)のみで有効で、その後のリリースでは使用できません。

Oracle Database 10gリリース2が実行されているシステムでワークロードの取得機能を有効にするには、SQLプロンプトでwrrenbl.sqlスクリプトを実行します。

```
@$ORACLE_HOME/rdbms/admin/wrrenbl.sql
```

wrrenbl.sqlスクリプトでは、ALTER SYSTEM SQL文がコールされ、PRE_11G_ENABLE_CAPTURE初期化パラメータがTRUEに設定されます。サーバー・パラメータ・ファイル(spfile)が使用されている場合、PRE_11G_ENABLE_CAPTURE初期化パラメータは、現在実行しているインスタンス用に変更された後、spfileに記録されるため、データベースを再起動しても、新しい設定が保持されるようになります。spfileが使用されていない場合、PRE_11G_ENABLE_CAPTURE初期化パラメータは、現在実行しているインスタンス用に変更されるだけであるため、データベースを再起動すると、新しい設定は保持されません。spfileを使用せずに設定を保持するには、初期化パラメータ・ファイル(init.ora)にパラメータを手動で指定する必要があります。

ワークロードの取得を無効にするには、SQLプロンプトでwrrdsbl.sqlスクリプトを実行します。

```
@$ORACLE_HOME/rdbms/admin/wrrdsbl.sql
```

wrrdsbl.sqlスクリプトでは、ALTER SYSTEM SQL文がコールされ、PRE_11G_ENABLE_CAPTURE初期化パラメータがFALSEに設定されます。サーバー・パラメータ・ファイル(spfile)が使用されている場合、PRE_11G_ENABLE_CAPTURE初期化パラメータは、現在実行しているインスタンス用に変更された後、spfileに記録されるため、データベースを再起動しても、新しい設定が保持されるようになります。spfileが使用されていない場合、PRE_11G_ENABLE_CAPTURE初期化パラメータは、現在実行しているインスタンス用に変更されるだけであるため、データベースを再起動すると、新しい設定は保持されません。spfileを使用せずに設定を保持するには、初期化パラメータ・ファイル(init.ora)にパラメータを手動で指定する必要があります。

ノート:

PRE_11G_ENABLE_CAPTURE 初期化パラメータは、Oracle Database 10g リリース 2(10.2)のみで使用できます。その後のリリースでは、このパラメータは無効です。データベースのアップグレード後、サーバー・パラメータ・ファイル(spfile)または初期化パラメータ・ファイル(init.ora)からパラメータを削除する必要があり、削除しないと、データベースは起動に失敗します。

Enterprise Managerの権限およびロール

任意のデータベース・リプレイ・エンティティを参照または操作するには、データベース・リプレイのリソース・タイプの権限が必要です。また、ワークロードに関連付けられたエンティティにアクセスするには、ワークロードを取得したターゲットのターゲット・オペレータ権限が必要です。ターゲットが現在存在しない場合、そのエンティティにアクセスするには、そのエンティティを所有するEnterprise ManagerまたはEnterprise Managerスーパー・ユーザーである必要があります。

次の項で説明する2つのセキュリティ・ロールを使用すると、データベース・リプレイ・エンティティに関連する権限の付与および取消しが簡単になります。

データベース・リプレイ・ビューア・ロール

データベース・リプレイ・ビューア・ロールを持つユーザーは、すべてのデータベース・リプレイ・エンティティを表示できます。デフォルトでは、このロールを付与されるEnterprise Managerはいません。ただし、EM_ALL_VIEWERロールにはデフォルトでこのロールが含まれています。

データベース・リプレイ・ビューア・ロールは、データベース・リプレイ・ビューア(リソース・タイプ)権限で構成されています。

データベース・リプレイ・オペレータ・ロール

データベース・リプレイ・オペレータ・ロールには、データベース・リプレイ・ビューア・ロールが含まれるので、その権限も含まれます。データベース・リプレイ・オペレータ・ロールを持つユーザーは、すべてのデータベース・リプレイ・エンティティを編集および削除できます。デフォルトでは、このロールを付与されるEnterprise Managerはいません。ただし、EM_ALL_OPERATORロールにはデフォルトでこのロールが含まれています。

データベース・リプレイ・オペレータ・ロールには、次の権限があります。

- データベース・リプレイ・オペレータ(リソース・タイプ権限)

- 名前付き資格証明の新規作成(リソース・タイプ権限)
- 新規ジョブの作成(リソース・タイプ権限)
- 表示可能な任意のターゲットに接続(ターゲット・タイプ権限)
- 任意の場所でのコマンドの実行(ターゲット・タイプ権限)

データベース・ターゲットでワークロードを取得またはリプレイする場合、Enterprise Managerユーザーにはデータベース・リプレイ・オペレータ・ロールのすべての権限に加え、ターゲット・データベースのターゲット・オペレータ権限が必要です。

Enterprise Managerを使用したデータベース・ワークロードの取得

この項では、Enterprise Managerを使用してデータベース・ワークロードを取得する方法について説明します。データベース・ワークロードを取得するための主要ツールは、Oracle Enterprise Managerです。

前提条件の詳細は、「データベース・ワークロードの取得の前提条件」を参照してください。

ヒント:

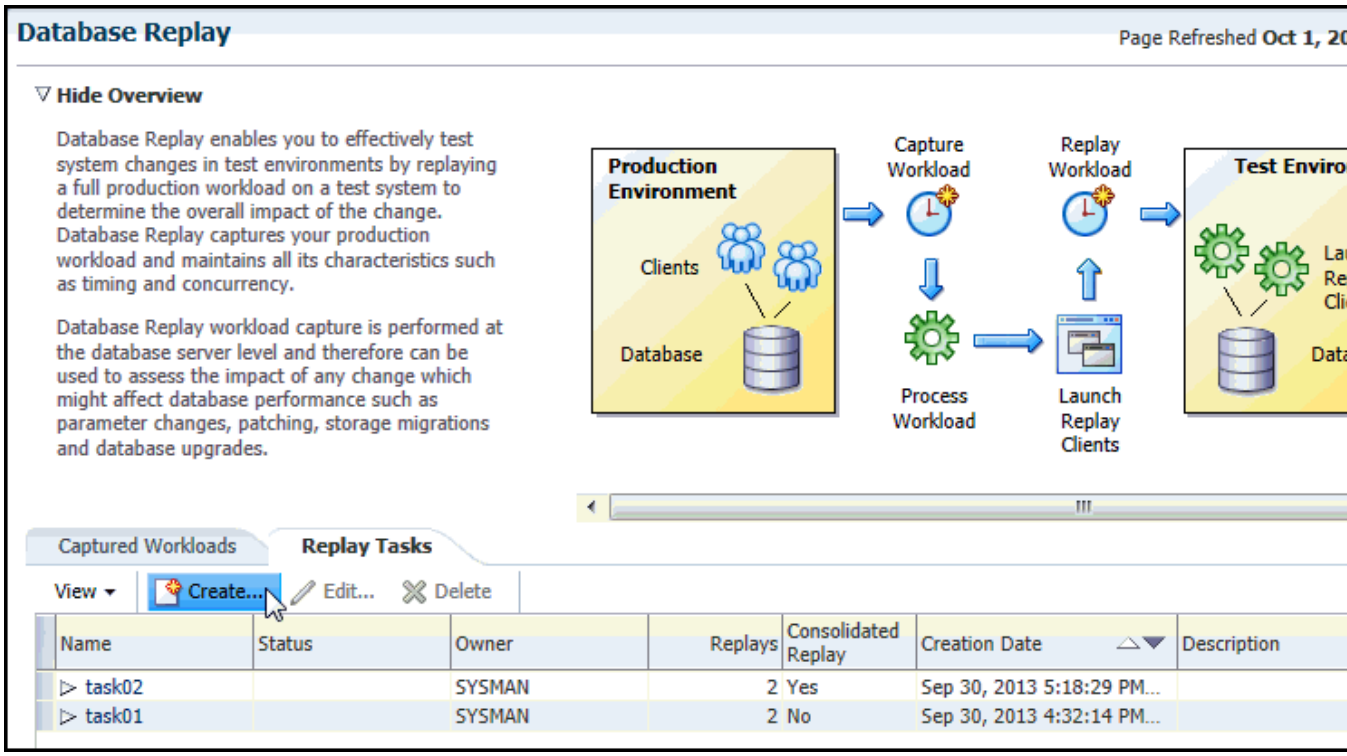
Oracle Enterprise Manager を使用できない場合は、「API を使用したデータベース・ワークロードの取得」で説明されているように、API を使用してデータベース・ワークロードを取得できます。

Enterprise Managerを使用してデータベース・ワークロードを取得するには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

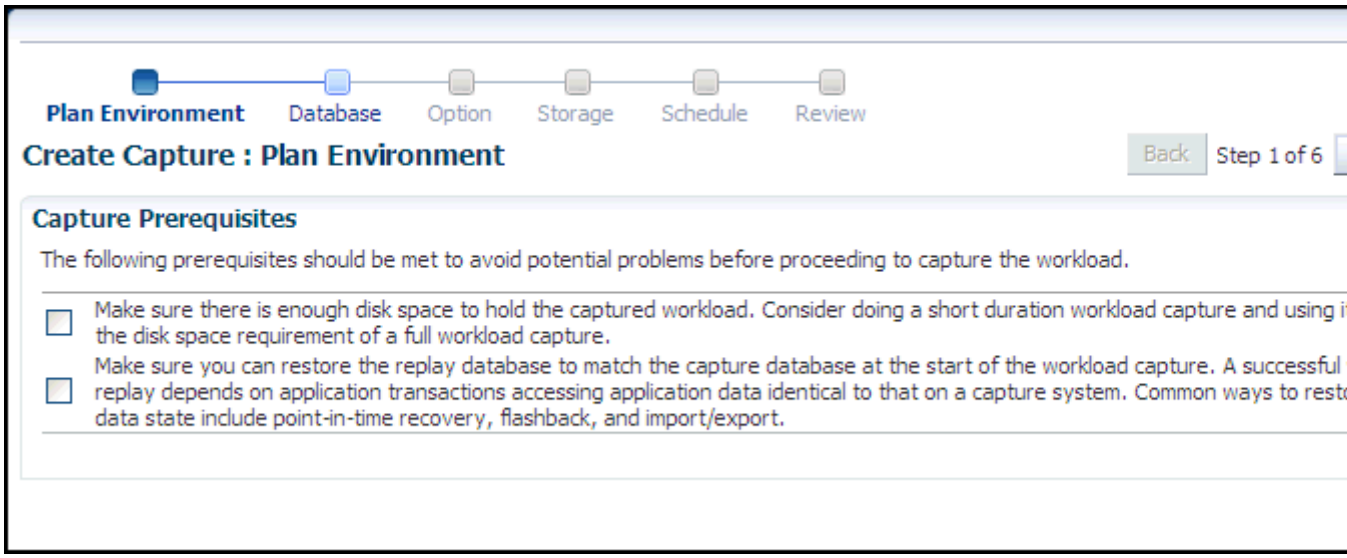
   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。



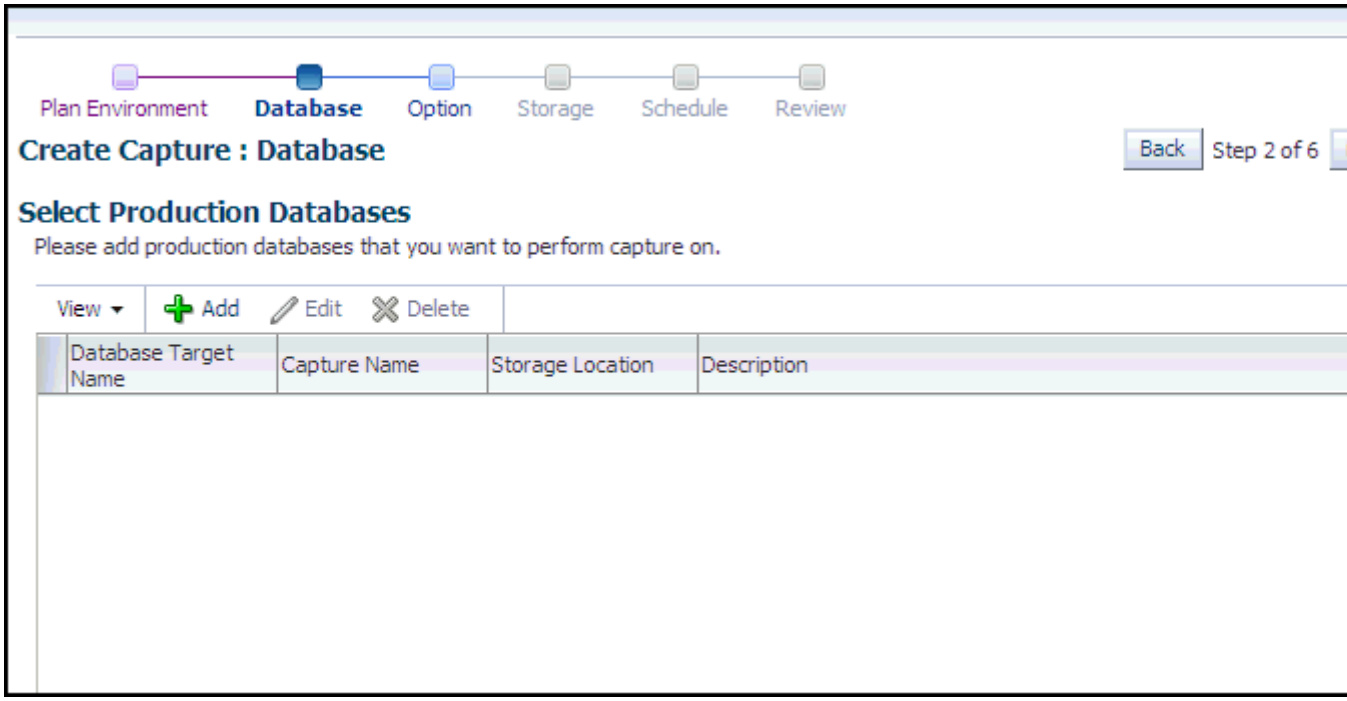
2. データベース・リプレイ・ページの「取得済ワークロード」タブをクリックし、ツールバーの「作成」をクリックします。

   取得の作成: 環境の計画ページが表示されます。



3. このページに説明されているいずれの前提条件も満たしていることを確認してから、両方のチェック・ボックスを選択し、「次へ」をクリックします。

   取得の作成: データベース・ページが表示されます。

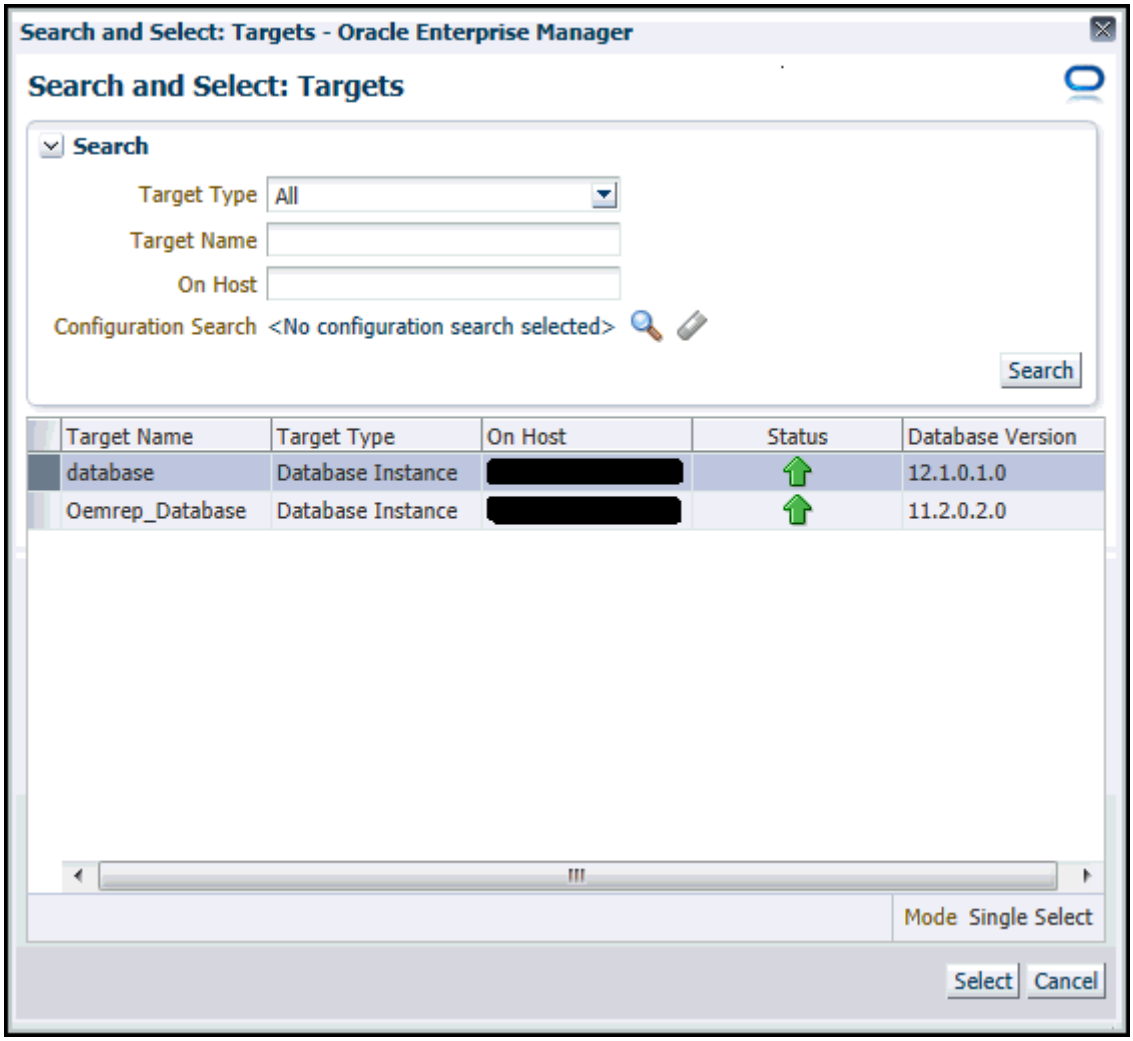


4. 「追加」をクリックします。

   「追加」ポップアップが表示されます。

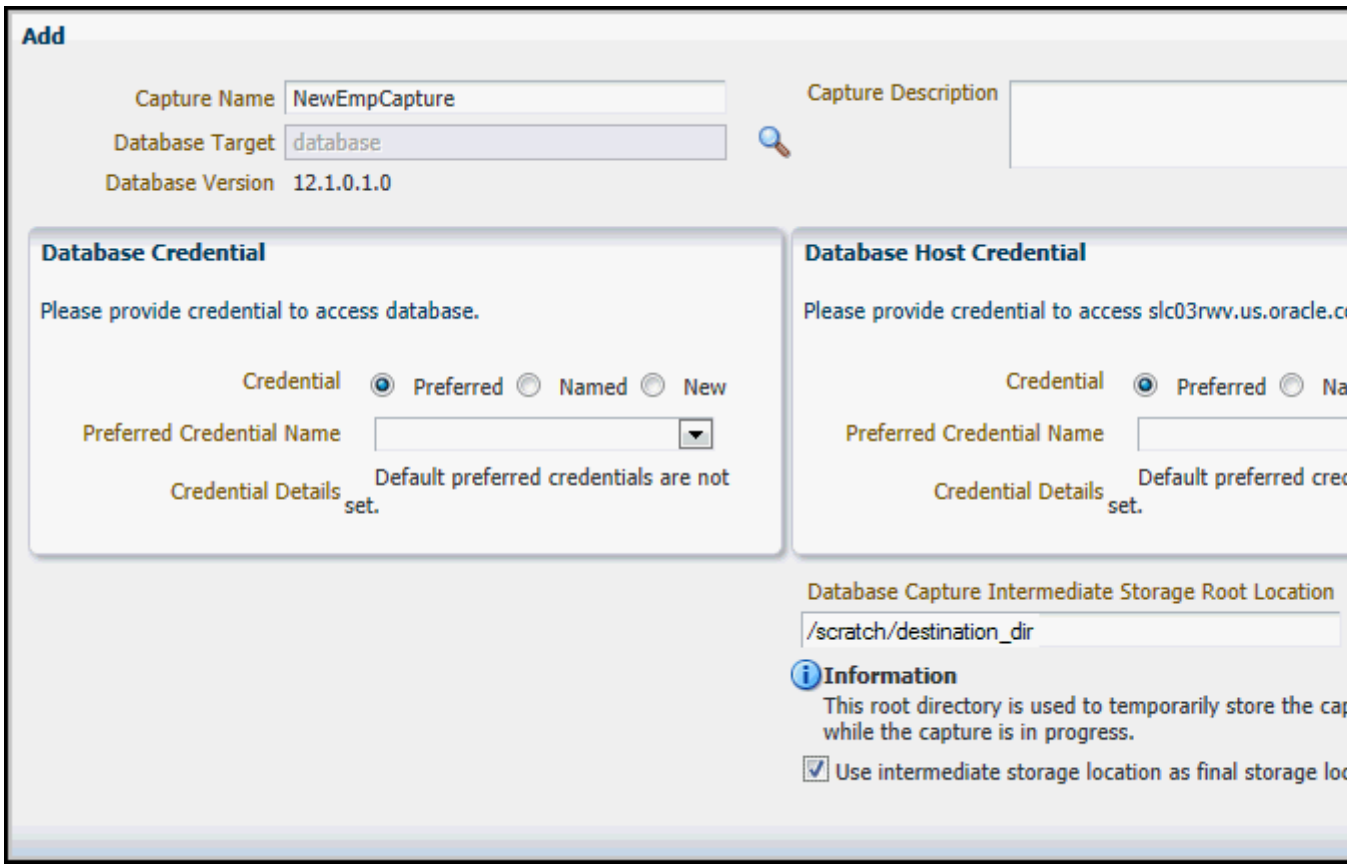
5. 取得名と説明(オプション)を入力し、「ターゲット・データベース」検索アイコンをクリックします。

   「検索と選択: ターゲット」ポップアップが表示されます。



6. 「ターゲット・タイプ」を選択し、任意で構成検索を指定し、一覧からターゲット・データベースを選択して、「選択」をクリックします。

   「データベース資格証明」および「データベース・ホスト資格証明」にセクションが追加された状態で、「追加」ポップアップが再度表示されます。



7. データベース資格証明、データベース・ホスト資格証明、データベース取得の中間記憶域の場所を入力し、「OK」をクリックします。
   - 取得後、中間記憶域の場所を最終の記憶域の場所として使用しないかぎり、ファイルは記憶域の場所にコピーされます。

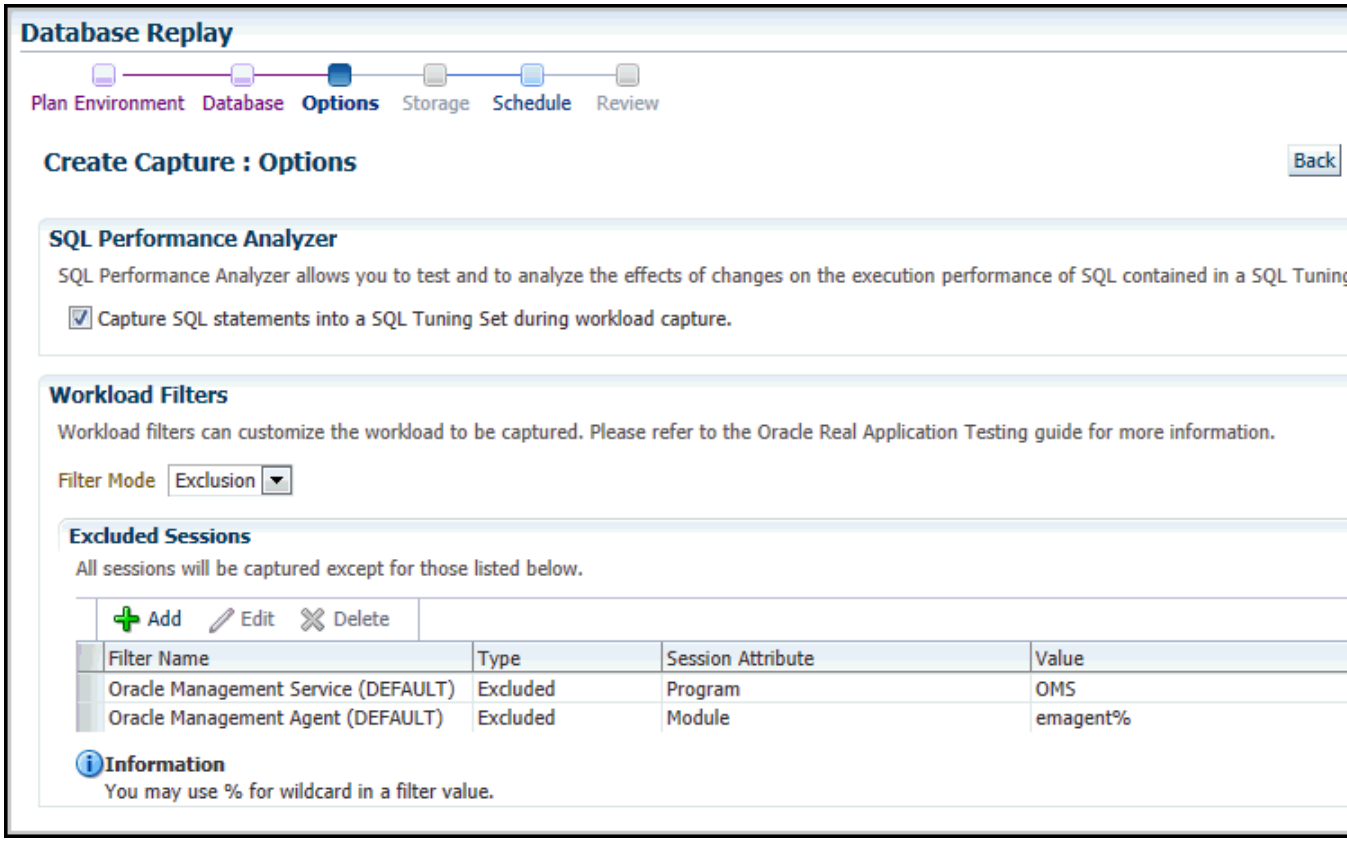
   ノート:

   Enterprise Manager では、Oracle RAC データベースに対する取得については、共有ファイル・システムで構成された Oracle RAC のみをサポートしています。

   これで、選択したターゲット・データベースが、本番データベースの選択表のデータベース・リストに表示されます。

8. 「次へ」をクリックします。

   取得の作成: オプション・ページが表示されます。



9. ワークロードの取得のオプションを選択します。
   - 「SQLパフォーマンス・アナライザ」セクションで、ワークロードの取得時に、SQL文をSQLチューニング・セットに取得するかどうかを選択します。

     データベースのリプレイでは、変更がシステム全体に及ぼす影響が分析されますが、SQLパフォーマンス・アナライザとともにSQLチューニング・セットを使用すれば、変更がSQL文と実行計画にどのように影響するか、SQLを中心とした分析を行うことができます。

     ワークロードの取得時にはSQLチューニング・セットを取得し、ワークロードのリプレイ時には別のSQLチューニング・セットを取得することで、SQL文を再実行することなく、SQLパフォーマンス・アナライザを使用してこれらのSQLチューニング・セットを相互に比較できます。これにより、データベース・リプレイを実行しながら、SQLパフォーマンス・アナライザ・レポートを生成して、変化の前後のSQLパフォーマンスを比較できます。

     ノート:

     SQL 文は、SQL チューニング・セットにデフォルトで取得され、推奨されるワークロード取得オプションです。Oracle RAC では、SQL 文は SQL チューニング・セットへは取得されません。

     ヒント:

     SQL パフォーマンス・アナライザ・レポートを使用した SQL チューニング・セットの比較の詳細は、「API を使用した SQL パフォーマンス・アナライザ・レポートの生成」を参照してください。
   - 「ワークロード・フィルタ」セクションで、「フィルタ・モード」リストの「除外」を選択して除外フィルタを使用するか、「包含」を選択して包含フィルタを使用します。

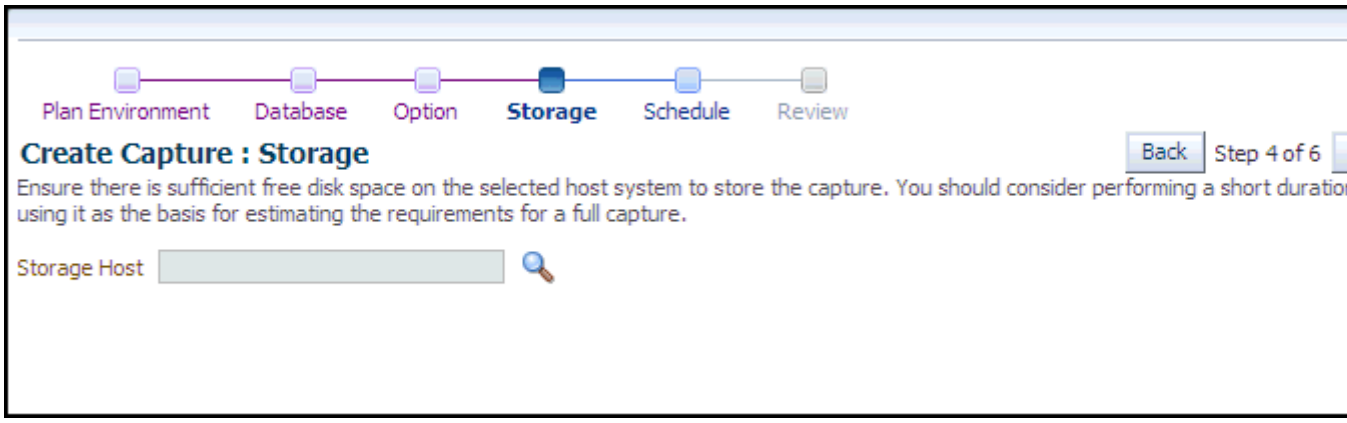
フィルタを追加するには、「追加」をクリックしてフィルタ名、セッション属性および値を該当するフィールドに入力します。

ヒント:

詳細は、「ワークロード取得でのフィルタの使用」を参照してください。

適切なワークロードの取得のオプションを選択したら、「次へ」をクリックします。

取得の作成：記憶域ページが表示されます。

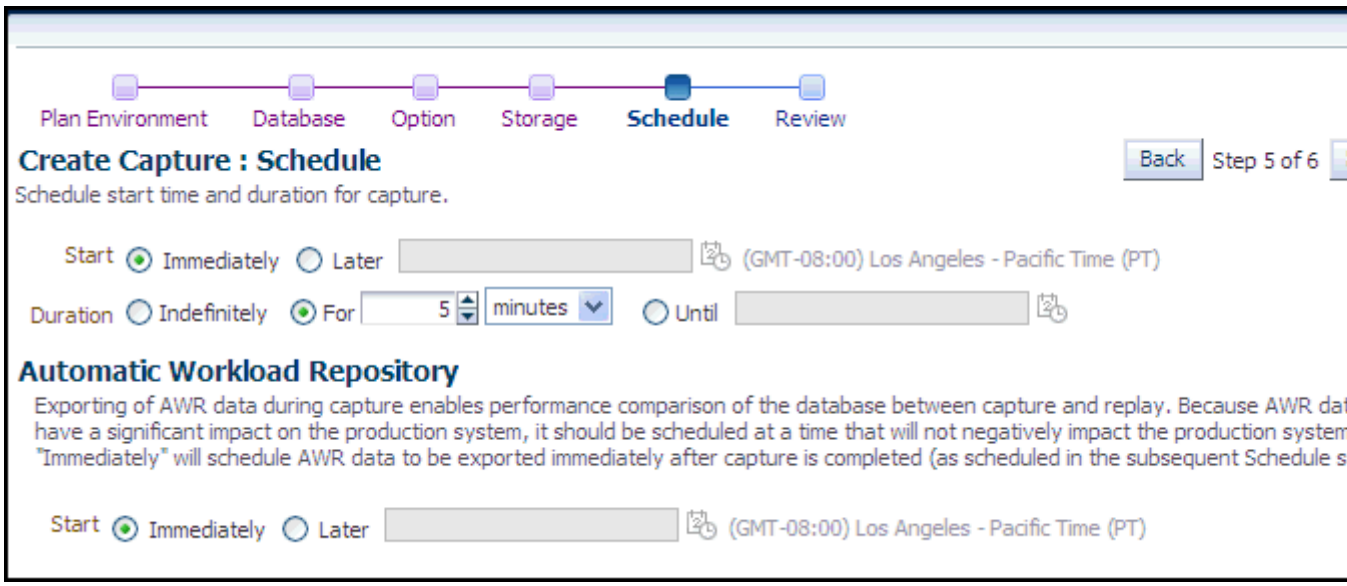


10. 「記憶域ホスト」アイコンをクリックし、リストからターゲットを選択して、「選択」をクリックします。

記憶域ページで、ホスト資格証明と記憶域の場所をリクエストされます。

11. ホスト資格証明を指定し、「参照」をクリックして記憶域の場所を選択し、場所を選択して「OK」、「次へ」の順にクリックします。

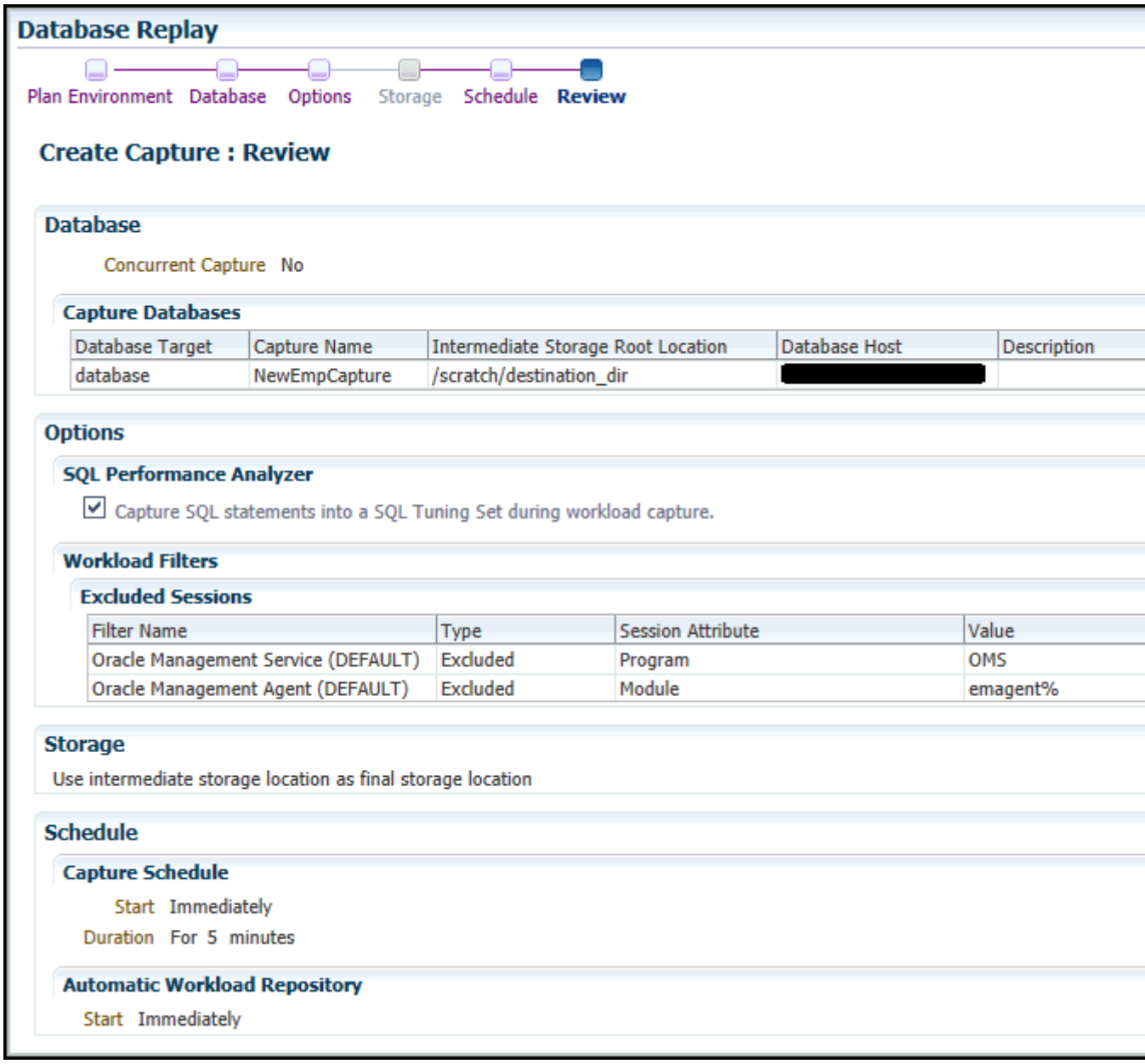
取得の作成：スケジュール・ページが表示されます。



12. 取得の開始時間と期間をスケジュールし、AWRデータのエクスポートをスケジュールして、「次へ」をクリックします。

- デフォルトの取得期間は5分です。テストが必要な対象期間の代理アクティビティを取得する取得期間を変更します。

取得の作成：確認ページが表示されます。



13. すべてのパラメータが意図したとおりに設定されたら、「発行」をクリックして取得ジョブを開始します。
    - デフォルトでは、「ワークロード取得中にSQL文をSQLチューニング・セットに取り込みます。」オプションが有効です。リプレイ終了時にSQLチューニング・セットを比較しない場合は、このオプションの選択を解除します。

データベース・リプレイ・ページが再度表示され、取得が正常に作成されたというメッセージと、「スケジュール済」などの取得のステータスが「取得」リストに表示されます。

14. 取得の詳細は、取得名をダブルクリックしてください。

取得サマリー・ページが表示され、平均アクティブ・セッション、ワークロードの比較、関連する同時取得など、複数の属性があれば表示されます。

ヒント:

本番システムでワークロードを取得したら、取得したワークロードを事前処理する必要があります。詳細は、「デー

タベース・ワークロードの事前処理」を参照してください。

複数のデータベースからのワークロードの同時取得

同時取得とは、複数のデータベースからワークロードを同時に取得することです。

データベース・リプレイ・ワークロードを同時に取得するには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。

2. データベース・リプレイ・ページの「取得済ワークロード」タブをクリックし、ツールバーの「作成」をクリックします。

   取得の作成: 環境の計画ページが表示されます。

3. このページに説明されているどちらの前提条件も満たしていることを確認してから、両方のチェック・ボックスを選択し、「次へ」をクリックします。

   取得の作成: データベース・ページが表示されます。

   ヒント:

   前提条件の詳細は、「データベース・ワークロードの取得の前提条件」を参照してください。

4. 「追加」をクリックします。

   「追加」ポップアップが表示されます。

5. 取得名と説明(オプション)を入力し、「ターゲット・データベース」検索アイコンをクリックします。

   「検索と選択: ターゲット」ポップアップが表示されます。

6. リストからターゲット・データベースを選択し、「選択」をクリックします。

   「データベース資格証明」および「データベース・ホスト資格証明」にセクションが追加された状態で、「追加」ポップアップが再度表示されます。

7. データベース資格証明、データベース・ホスト資格証明、データベース取得の中間記憶域の場所を入力し、「OK」をクリックします。

   - 取得後、中間記憶域の場所を最終の記憶域の場所として使用しないかぎり、ファイルは記憶域の場所にコピーされます。

   これで、選択したターゲット・データベースが、本番データベースの選択表のデータベース・リストに表示されます。

8. 同時取得する別のデータベースを追加します。

   - ステップ4から7の指示に従います。

     取得の作成: データベース・ページが再表示され、ステップ4から7で指定した1つ目のデータベースとともに、取得する別のデータベースが表示されます。

- 同時取得の名前と説明(オプション)を入力し、「次へ」をクリックします。

  取得の作成：オプション・ページが表示されます。

9. ワークロードの取得のオプションを選択します。

   - 「SQLパフォーマンス・アナライザ」セクションで、ワークロードの取得時に、SQL文をSQLチューニング・セットに取得するかどうかを選択します。

     データベースのリプレイでは、変更がシステム全体に及ぼす影響が分析されますが、SQLパフォーマンス・アナライザとともにSQLチューニング・セットを使用すれば、変更がSQL文と実行計画にどのように影響するか、SQLを中心とした分析を行うことができます。

     ワークロードの取得時にはSQLチューニング・セットを取得し、ワークロードのリプレイ時には別のSQLチューニング・セットを取得することで、SQL文を再実行することなく、SQLパフォーマンス・アナライザを使用してこれらのSQLチューニング・セットを相互に比較できます。これにより、データベース・リプレイを実行しながら、SQLパフォーマンス・アナライザ・レポートを生成して、変化の前後のSQLパフォーマンスを比較できます。

     ノート:

     SQL チューニング・セットへの SQL 文の取得はデフォルトで実行されます。これは、推奨されるワークロード取得オプションです。

   - 「ワークロード・フィルタ」セクションで、「フィルタ・モード」リストの「除外」を選択して除外フィルタを使用するか、「包含」を選択して包含フィルタを使用します。

     フィルタを追加するには、「追加」をクリックしてフィルタ名、セッション属性および値を該当するフィールドに入力します。

   適切なワークロードの取得のオプションを選択したら、「次へ」をクリックします。

   取得の作成：記憶域ページが表示されます。

10. 「記憶域ホスト」アイコンをクリックし、リストからターゲットを選択して、「選択」をクリックします。

    記憶域ページで、ホスト資格証明と記憶域の場所をリクエストされます。

11. ホスト資格証明を入力し、記憶域を選択するために「参照」をクリックし、「次」をクリックします。

    取得の作成：スケジュール・ページが表示されます。

12. 取得の開始時間と期間をスケジュールし、AWRデータのエクスポートをスケジュールして、「次へ」をクリックします。

    - デフォルトの取得期間は5分です。テストが必要な対象期間の代理アクティビティを取得する取得期間を変更します。

    取得の作成：確認ページが表示されます。

13. すべてのパラメータが意図したとおりに設定されたら、「発行」をクリックして取得ジョブを開始します。

    - デフォルトでは、「ワークロード取得中にSQL文をSQLチューニング・セットに取り込みます。」オプションが有効です。リプレイ終了時にSQLチューニング・セットを比較しない場合は、このオプションの選択を解除します。

    データベース・リプレイ・ページが再度表示され、取得が正常に作成されたというメッセージと、「スケジュール済」などの取

得のステータスが「取得」リストに表示されます。

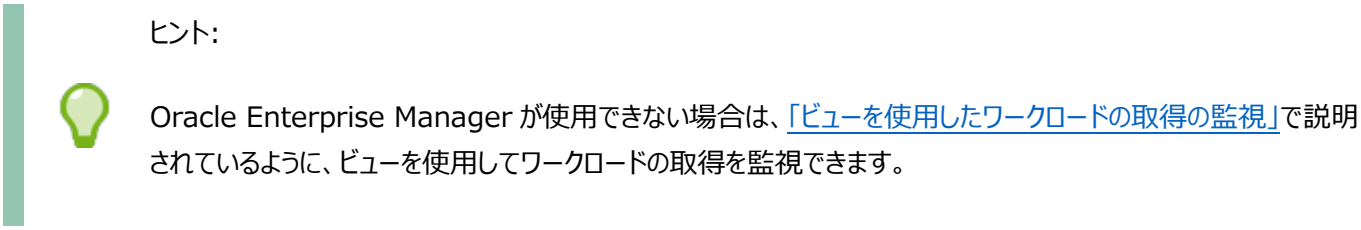
14. 取得の詳細は、取得名をダブルクリックしてください。

取得サマリー・ページが表示され、平均アクティブ・セッション、ワークロードの比較、関連する同時取得など、複数の属性が表示されます。

Enterprise Managerを使用したワークロードの取得の監視

この項では、Enterprise Managerを使用してワークロードの取得を監視する方法について説明します。ワークロードの取得を監視するための主要ツールは、Oracle Enterprise Managerです。Enterprise Managerを使用すると、次の操作を実行できます。

- アクティブなワークロードの取得の監視または停止
- 完了したワークロード取得の表示



この項では、次の項目について説明します。

- アクティブなワークロードの取得の監視
- アクティブなワークロードの取得の停止
- 完了したワークロード取得の表示

アクティブなワークロードの取得の監視

この項では、Enterprise Managerを使用してアクティブなワークロードの取得を監視する方法について説明します。

アクティブなワークロードの取得を監視するには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。

2. ステータスが「完了」以外の取得のデータベース・リプレイ・ページにある「取得済ワークロード」タブで、「取得」表から必要な取得の名前をクリックします。

   詳細な統計、取得の進行中に動的に更新される平均アクティブ・セッションのグラフ、取得された要素と取得されなかった同じ要素間のデータの比較、および関連する同時取得(存在する場合)が表示された、データベース・リプレイ・ページの「サマリー」タブが開きます。

   - 「平均アクティブ・セッション」グラフに表示されている「取得されていません」データは、現在取得されていないデータベース・アクティビティ(データベース・セッション)を表します。

- 「比較」セクションの「合計」列の値は、データベースの取得されたアクティビティと取得されていないアクティビティのすべてを表します。取得の作成ウィザードの「オプション」のステップで指定したワークロード・フィルタによって決められているフィルタリングは、主に、あるアクティビティが取得または取得されていない理由を意味します。また、バックグラウンド・アクティビティ、データベース・スケジューラ・ジョブ、およびリプレイ不可能なコールは取得されません。
- 右上隅のリフレッシュ・アイコンをクリックして、実行中に取得を更新できます。

3. データベース・リプレイ・ページに戻るには、「データベース・リプレイ」ブレッドクラムをクリックします。

アクティブなワークロードの取得の停止

この項では、Enterprise Managerを使用したアクティブなワークロードの取得の停止方法について説明します。

アクティブなワークロードの取得を停止するには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。

2. ステータスが「ドラフト」の取得のデータベース・リプレイ・ページの「取得済ワークロード」タブで、停止する取得の「取得」表の名前をクリックします。

   取得サマリー・ページが表示されます。

3. 「取得の停止」ボタンをクリックします。

   ボタン・ラベルが「取得の停止中」に変更されます。処理が完了すると、「ステータス」が「停止中」に変更されます。

完了したワークロード取得の表示

この項では、Enterprise Managerを使用したワークロード取得全体の管理方法について説明します。

完了したワークロードの取得を表示するには:

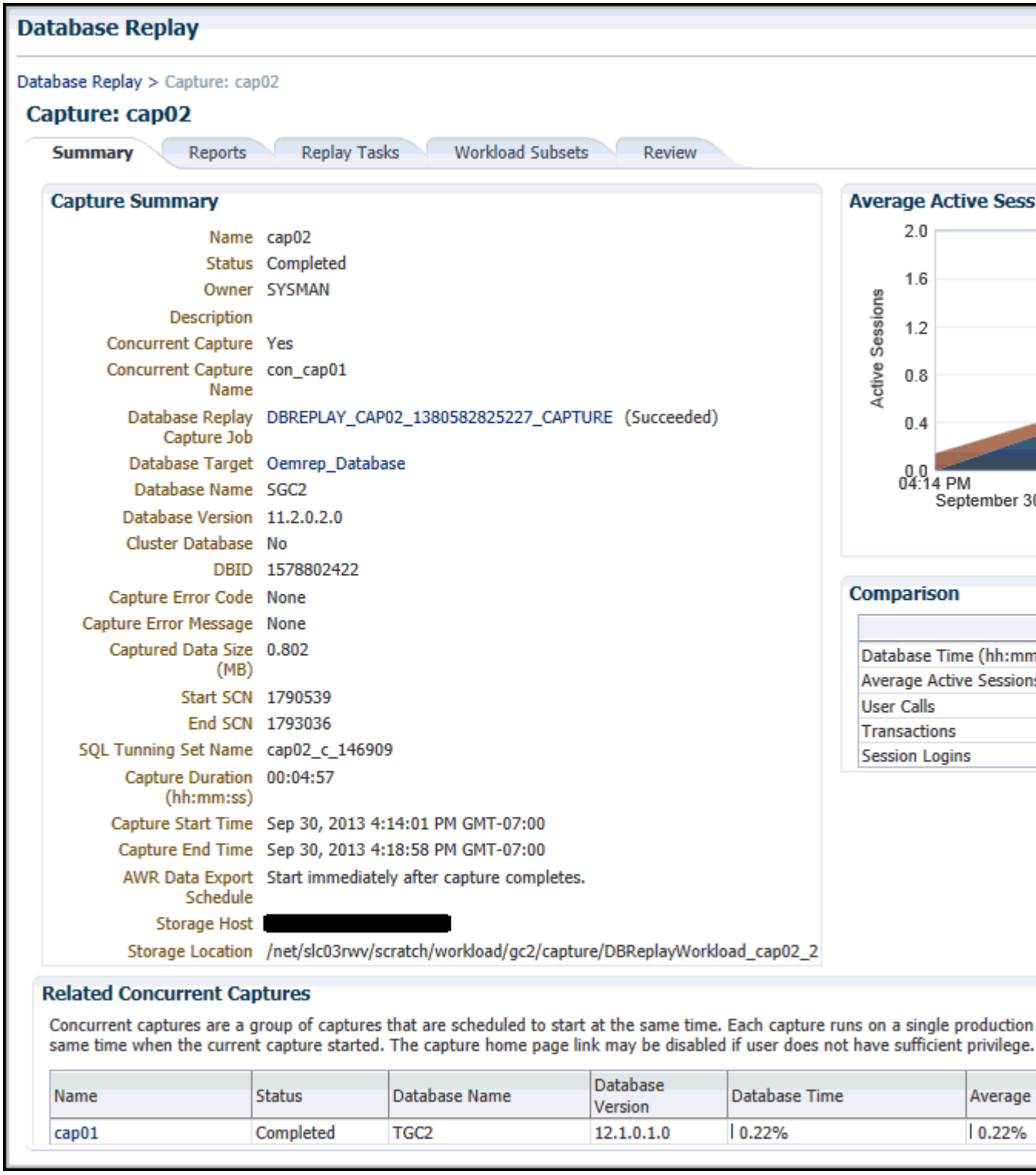
1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。

2. データベース・リプレイ・ページにある「取得済ワークロード」タブで、ステータスが「完了」の取得の名前をクリックします。

   「平均アクティブ・セッション」グラフに、取得中に記録された動的データではなく、取得期間の集計データが表示されることを除き、データベース・リプレイ・ページの「サマリー」タブには、進行中の取得に関するコンテンツが表示されます。



「平均アクティブ・セッション」チャートには、取得されたセッション・アクティビティが取得されていないセッション・アクティビティ(バックグラウンド・アクティビティやフィルタ処理されたセッションなど)と比較されてグラフィカルに表示されます。このチャートは、取得期間に使用可能なアクティブ・セッション履歴(ASH)データが存在する場合にのみ表示されます。

「比較」の下には、ワークロード取得の様々な統計が表示されます。

- 「取得」列

  取得されたセッション・アクティビティの統計が表示されます。

- 「合計」列

  合計セッション・アクティビティの統計が表示されます。

- 「合計の割合」列

  ワークロードで取得された合計セッション・アクティビティの割合が表示されます。

3. データベース・リプレイ・ページに戻るには、「データベース・リプレイ」ブレッドクラムをクリックします。

ヒント:

ワークロードの取得レポートへのアクセスの詳細は、「取得およびリプレイ済ワークロードの分析」を参照してください。

Enterprise Manager外部のワークロードのインポート

PL/SQLインタフェースまたは異なるEnterprise Managerインスタンスを通して取得したワークロードをEnterprise Managerにインポートして、ワークロードが元々Enterprise Manager内で作成されたように管理することができます。ワークロードを取得中にしたり、取得を完了して、ワークロードをファイル・システムに格納できます。また、Enterprise Manager内で作成されたワークロードの場合と同様に、ワークロードを事前処理およびリプレイすることもできます。

この機能は、Cloud Control Databaseプラグイン12.1.0.5以降のリリースで利用できます。

Enterprise Manager外部にデータベース・ワークロードをインポートするには:

1. 「データベース・リプレイ」ページの「取得済ワークロード」タブをクリックし、ツールバーの「インポート」をクリックします。

   「ワークロードのインポート: ソース」ページが表示されます。

2. 次の3つの選択肢のいずれかを選択して、取得されたワークロードをインポートし、「次へ」をクリックします。

   - 取得が完了したワークロードをファイル・システムのディレクトリからインポート

     このオプションは、通常APIを使用して作成されたワークロードに適用され、後続の処理用にEnterprise Managerにインポートします。この場合、Enterprise Managerは取得データベースを管理していない可能性があります。

   - 取得が完了したワークロードをデータベース・ターゲットからインポート

     この場合、Enterprise Managerは取得データベースをすでに管理している可能性があります。取得は、このデータベースで実行された可能性があるか、前述のオプションの場合と同様にロードされた可能性があります。

   - データベース・ターゲットで実行中の取得にアタッチ

     このオプションは前述のオプションと似ていますが、すでに完了している取得ではなく、実行中の取得である点が異なります。

   「ワークロードのインポート: データベース」ページが表示されます。

3. 「データベース・ターゲット」フィールドの横の検索アイコンをクリックして、表示されるポップアップからデータベースを選択します。

ノート:

ワークロードをロードするデータベースのターゲット・バージョンは、ワークロードの取得に使用したバージョン以

上である必要があります。たとえば、Oracle Database 12x を使用して取得した場合、取得の読取りで選択するデータベースはバージョン 12x 以上である必要があります。

- データベースおよびホスト資格証明が要求されます。
- 前述のステップで、取得が完了したワークロードをファイル・システムのディレクトリからインポートを選択した場合は、ワークロードの場所も要求されます。
- 統合リプレイは、少なくとも2つの取得ディレクトリを含む別のディレクトリ構造を持っています。したがって、ワークロード・ディレクトリに統合リプレイが含まれている場合は、インポート操作中にEnterprise Managerが統合リプレイを認識できるように、チェック・ボックスを有効にする必要があります。

4. 前述のステップで必要な入力を行い、「次へ」をクリックします。

   「ワークロードのインポート: ワークロード」ページが表示されます。

   - ステップ2で、「完了した取得済ワークロードをファイル・システムのディレクトリからインポートします。」を選択した場合は、このページに「ワークロードのロード」ボタンが表示されます。
   - ステップ2で、「完了した取得済ワークロードをデータベース・ターゲットからインポートします。」または「データベース・ターゲットで実行中の取得にアタッチします。」を選択した場合は、このページに「ワークロードの検出」ボタンが表示されます。

5. ステップ2の選択に従って表示されるボタンに応じて、ワークロードのロードまたはワークロードの検出のいずれかをクリックします。

   ワークロードが見つかった場合は、検出されたワークロード表に表示されます。

6. 「次へ」をクリックして、ワークロードをロードするか、ワークロードの1つを選択し、「次へ」をクリックしてワークロードのインポートを継続します。

   「ワークロードのインポート: リプレイ」ページが表示されるのは、次の条件の場合のみです。

   - ステップ2で、「完了した取得済ワークロードをファイル・システムのディレクトリからインポートします。」を選択した場合。
   - 1つ以上のリプレイがワークロードに含まれている場合。

7. オプション: 必要に応じて1つ以上のリプレイを選択し、リプレイ・タスク名を選択して、「次へ」をクリックします。

   「ワークロードのインポート: 確認」ページが表示されます。

8. すべてが目的どおりに表示されたら、「送信」をクリックします。

   「データベース・リプレイ」ページには、ジョブが正常に送信されたことを示すメッセージが表示されます。「取得済ワークロード」表で、ロードまたはインポートされたワークロードの「ステータス」列に「進行中」と表示されます。

ヒント:

「確認」ステップで送信した取得済ワークロード名をクリックして、ジョブの進行状況を確認できます。「取得サマリー」ページが表示され、データベース・リプレイ・インポート・ジョブ・リンクをクリックして、ジョブ実行ステップの進行状況を確認できます。

既存のワークロードからのサブセットの作成

長時間をカバーする大きなワークロードを取得した場合は、テストを迅速化するために、ワークロードの特定の部分をリプレイするだけで済みます。データベース・リプレイ・ワークロードのサブセット化機能は、既存の取得済ワークロードの一部を抽出することにより、新しいワークロードを作成できます。

Enterprise Managerには、テスト・システムでリプレイに使用できる既存のワークロードからデータのサブセットを抽出するウィザードが用意されています。抽出された各サブセットは、自身のワークロード上または統合リプレイの他のワークロードでリプレイできる正規のワークロードです。

リプレイを実行するには、ワークロードを事前処理する必要があります。

この機能は、Cloud Control Databaseプラグイン12.1.0.5以降のリリースで利用できます。

ワークロードからサブセットを抽出するには:

1. 「データベース・リプレイ」ページの「取得済ワークロード」タブから、サブセットを抽出するワークロードを選択し、「サブセット」をクリックします。

   「サブセット・ワークロード: 定義」ページが表示され、ワークロードの「アクティブ・セッション履歴」グラフが表示されます。

2. ワークロードから抽出するサブセットの開始および終了時間を選択します。

   a. ページの下部にある「サブセット」表の上の「追加」をクリックします。

      サブセットの作成ポップアップが表示されます。

   b. スナップショットまたはカレンダ時間のいずれかを選択して、開始および終了時間を指定し、「OK」をクリックします。

      ノート:

      すべてのパフォーマンス・データが、選択したカレンダで使用できるとは限らないため、スナップショット時間が推奨される選択肢になります。

      選択した期間は、グレー表示されたセグメントとして「アクティブ・セッション」グラフに表示されます。

   c. オプション: 前述のステップで選択した場合とは異なる期間で1つ以上の追加のサブセットを定義します。

   d. オプション: 「拡張パラメータ」セクションで、サブセット・ワークロードの終了後に未完了のコールを含めるかどうかを指定します。デフォルトでは、サブセットのワークロードが開始されると、未完了のコールが含まれます。

      - これらのパラメータでは、定義された境界の外部コールを組み込むことができます。たとえば、トランザクションの境界として、開始および終了時間を指定すると、指定された開始時間より前にトランザクションが開始され、指定された終了時間後も継続される場合があります。

3. 「次へ」をクリックします。

   「サブセット・ワークロード: データベース」ページが表示されます。

4. 「データベース・ターゲット」フィールドの横の検索アイコンをクリックして、表示される「検索と選択: ターゲット」ポップアップから、サブセット化用のデータベースを選択します。

データベースおよびホスト資格証明が要求されます。

5. 前述のステップで必要な入力を行い、「次へ」をクリックします。

   「サブセット・ワークロード: 場所」ページが表示されます。

   a. ソース・ホストおよびステージング・データベース・ホストが同じ場合は、場所が事前入力されるので、ソース・ワークロード・ファイルの場所を指定する必要はありません。

   b. ソース・ホストおよびステージング・データベースが同じでない場合は、次の手順を実行します。

      - 「ホスト」フィールドに表示されるホスト名からワークロード・ファイルにアクセスするかどうか、また、ソース・ホストから、表示される「宛先ホスト」にファイルをコピーするかどうかを選択します。

        直接アクセスは、ワークロードのサブセット化に使用するデータベースが、指定したファイル・システムの場所を使用して元のワークロードに直接アクセスできることを意味します。これは、通常は元のワークロードがネットワーク共有された場所に格納されている場合です。

        アクセス権のコピーは、2つのホストが共有ネットワーク・パスにないことを意味します。Enterprise Managerが元のワークロードを現在の場所から、サブセット・データベース・ホスト上の指定された場所にコピーできるように、ソース・ホスト資格証明を指定する必要があります。

      - 前述の選択内容に応じて、ワークロード・ファイルを含むディレクトリの場所を指定するか、宛先ホストの場所を指定します。

6. 「サブセット」フィールドで、各サブセットの記憶域の場所を指定し、「次へ」をクリックします。

   「サブセット・ワークロード: スケジュール」ページが表示されます。

7. サブセット・ジョブを開始する時間を指定し、「次へ」をクリックします。

   「サブセット・ワークロード: 確認」ページが表示されます。

8. すべてが目的どおりに表示されたら、「送信」をクリックします。

   「データベース・リプレイ」ページには、ジョブが正常に送信されたことを示すメッセージが表示されます。「リプレイ・タスク」表のサブセットの「ステータス」列に、「進行中」と表示されます。

> ヒント:
>
> 「確認」ステップで送信したサブセット名をクリックして、ジョブの進行状況を確認できます。「取得サマリー」ページが表示され、データベース・リプレイ・サブセット・ジョブリンクをクリックして、ジョブ実行ステップの進行状況を確認できます。

新しい場所からのワークロードのコピーまたは移動

2つの目的で、コピー機能を使用できます。目的は、次のとおりです。

- ソースから別のホストおよび場所に取得ファイルを複製する。
- 新しいホストおよび場所に取得ファイルを移動し、元の場所からソース・ファイルを削除する。

この機能は、Cloud Control Databaseプラグイン12.1.0.5以降のリリースで利用できます。

新しい場所にワークロードをコピーするには:

1. 「データベース・リプレイ」ページの「取得済ワークロード」タブで、新しい場所にコピーするワークロードを選択し、「コピー」をクリックします。

   「ワークロードのコピー」ページが表示され、選択したワークロード・ディレクトリの現在のソースの場所が表示されます。

2. 必要に応じて、記憶域ホストの資格証明を指定または変更します。現在の記憶域ホストで定義済の資格証明が自動的に取得されます。

3. 「コピー後」ラジオ・ボタンを有効のままにします(ソースの場所の元のワークロードが保持されます)。

4. ワークロード・ディレクトリの新しい場所の「記憶域ホスト」を選択します。

5. 新しい記憶域ホストの資格証明を指定します。

6. ワークロードの新しい「宛先ロケーション」のディレクトリを選択します。

7. ジョブをスケジュールし、「発行」をクリックします。

   「データベース・リプレイ」ページが再度表示され、ジョブが発行され、記憶域の場所が正常に更新されたことを示すメッセージが表示されます。

ヒント:

「ジョブ・アクティビティ」ページに移動して、送信メッセージに表示されたジョブ名を検索し、そのリンクをクリックして「ジョブ実行」ページにアクセスすることで、ジョブの進行状況をチェックできます。

APIを使用したデータベース・ワークロードの取得

この項では、APIを使用してデータベース・ワークロードを取得する方法について説明します。また、「Enterprise Managerを使用したデータベース・ワークロードの取得」で説明されているように、Oracle Enterprise Managerを使用してデータベース・ワークロードを取得できます。

DBMS_WORKLOAD_CAPTUREパッケージを使用してデータベース・ワークロードを取得する手順は、次のとおりです。

- ワークロード取得フィルタの定義
- ワークロードの取得の開始
- ワークロードの取得の停止
- ワークロードの取得のAWRデータのエクスポート
- ワークロードの取得のAWRデータのインポート

関連項目:

- DBMS_WORKLOAD_CAPTUREパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

ワークロード取得フィルタの定義

この項では、ワークロード取得フィルタを追加および削除する方法について説明します。ワークロード取得に対するワークロード・

フィルタの使用については、「ワークロード取得でのフィルタの使用」を参照してください。

ワークロード取得にフィルタを追加するには:

- ADD_FILTERプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.ADD_FILTER (
                              fname => 'user_ichan',
                              fattribute => 'USER',
                              fvalue => 'ICHAN');
END;
/
```

この例では、ADD_FILTERプロシージャは、user_ichanというフィルタを追加します。このフィルタは、ユーザー名ICHANに属するすべてのセッションを除外するために使用できます。

この例のADD_FILTERプロシージャでは、次のパラメータを使用します。

- fname: 追加するフィルタの名前を指定する必須パラメータ。
- fattribute: フィルタを適用する属性を指定する必須パラメータ。有効値は、PROGRAM、MODULE、ACTION、SERVICE、INSTANCE_NUMBERおよびUSERです。
- fvalue: フィルタを適用する属性に対応する値を指定する必須パラメータ。一部の属性(モジュールやアクションなど)では、%などのワイルドカードを使用できます。

ワークロード取得からフィルタを削除するには:

- DELETE_FILTERプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.DELETE_FILTER (fname => 'user_ichan');
END;
/
```

この例では、DELETE_FILTERプロシージャは、ワークロードの取得からuser_ichanというフィルタを削除します。

この例のDELETE_FILTERプロシージャは、必須パラメータfnameを使用します。このパラメータは、削除するフィルタの名前を指定します。DELETE_FILTERプロシージャは、完了した取得に属するフィルタは削除せず、まだ開始されていない取得のフィルタのみに適用されます。

ワークロードの取得の開始

この項では、ワークロード取得を開始する方法について説明します。
ワークロードの取得を開始する前に、データベース・ワークロードを取得するための前提条件を満たす必要があります。前提条件は、「データベース・ワークロードの取得の前提条件」に記載されています。また、「ワークロードの取得のオプション」で説明されているように、ワークロードの取得のオプションを確認する必要があります。

取得されたワークロードのリプレイを開始する前にリプレイ・システムを開始ポイントにリストアできるように、適切に定義されたワークロードの開始ポイントを取得することが重要です。ワークロードの取得の開始ポイントを適切に定義するには、ワークロードの取得が開始されたときにアクティブなユーザー・セッションが存在しないことが理想的です。アクティブ・セッションでトランザクションが進行中の場合、それらのトランザクションはその後のデータベース・リプレイで適切にリプレイされませんが、これは、トランザクションのうち、ワークロードの取得を開始した後に実行されたコールの部分しかリプレイされないためです。この問題を避けるには、ワーク

ロードの取得を開始する前にSTARTUP RESTRICTを使用してデータベースを制限モードで再起動します。ワークロードの取得が開始されると、データベースは自動的に無制限モードに切り替わり、ワークロードの取得中は通常の操作を続行できます。ワークロード取得前のデータベースの再起動の詳細は、「データベースの再起動」を参照してください。

ワークロード取得を開始するには:

- START_CAPTUREプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.START_CAPTURE (name => 'dec10_peak',
                               dir => 'dec10',
                               duration => 600,
                               capture_sts => TRUE,
                               sts_cap_interval => 300,
                               plsql_mode => 'extended',
                               encryption => 'AES256');
END;
/
```

この例では、dec10_peakという名前のワークロードが600秒間取得され、dec10という名前のデータベース・ディレクトリ・オブジェクトによって示されるファイル・システム・ディレクトリに格納されます。ワークロードの取得と並行して、SQLチューニング・セットも取得されます。

この例のSTART_CAPTUREプロシージャでは、次のパラメータを使用します。

- name: 取得するワークロードの名前を指定する必須パラメータ。
- dir: 取得したワークロードを格納するディレクトリを指すディレクトリ・オブジェクトを指定する必須パラメータ。
- duration: ワークロードの取得が終了するまでの時間(秒)を指定するパラメータ。値を指定しない場合、ワークロードの取得は、FINISH_CAPTUREプロシージャがコールされるまで続きます。
- capture_sts: ワークロードの取得と並行してSQLチューニング・セットを取得するかどうかを指定するパラメータ。このパラメータをTRUEに設定した場合、ワークロードの取得中に1つのSQLチューニング・セットを取得し、ワークロードのリプレイ中にもう1つ別のSQLチューニング・セットを取得することで、SQL文を再実行しなくても、SQLパフォーマンス・アナライザを使用してそれらのSQLチューニング・セットを比較できます。この方法では、データベース・リプレイの実行中に、SQLパフォーマンス・アナライザ・レポートを取得して、変更の前と後のSQLパフォーマンスを比較することができます。また、「ワークロードの取得のAWRデータのエクスポート」で説明されているように、EXPORT_AWRプロシージャを使用して、結果のSQLチューニング・セットをそのAWRデータとともにエクスポートできます。

  この機能は、Oracle RACではサポートされていません。DBMS_WORKLOAD_CAPTUREを使用して定義したワークロード取得フィルタは、SQLチューニング・セットの取得には適用されません。このパラメータのデフォルト値はFALSEです。
- sts_cap_interval: カーソル・キャッシュからのSQLチューニング・セット取得の継続時間(秒)を指定するパラメータ。デフォルト値は300です。このパラメータの値をデフォルト値未満に設定すると、一部のワークロードで追加オーバーヘッドが生じる可能性があるため、そのような設定は推奨されません。
- オプションのplsql_modeパラメータでは、取得およびリプレイ時にPL/SQLがDBリプレイでどのように処理されるか指定します。

  plsql_modeパラメータには次の2つの値を設定できます。

- top_level: 最上位レベルのPL/SQLコールのみ取得およびリプレイされます。これは、Oracle Database 12cリリース2 (12.2.0.1)より前のDBリプレイによるPL/SQLの処理方法です。これがデフォルト値です。
- extended: 最上位レベルのPL/SQLコールとPL/SQLからコールされたSQLの両方が取得されます。ワークロードがリプレイされる際、リプレイは最上位レベルと拡張レベルのいずれかで実行されます。

- encryptionパラメータでは、取得ファイルの暗号化に使用されるアルゴリズムを指定します。

  encryptionパラメータでは、次の暗号化標準を使用できます。

  - NULL - 取得ファイルは暗号化されません(デフォルト値)
  - AES128 - 取得ファイルはAES128を使用して暗号化されます
  - AES192 - 取得ファイルはAES192を使用して暗号化されます
  - AES256 - 取得ファイルはAES256を使用して暗号化されます

ノート:

暗号化ありで START_CAPTURE を実行するには、oracle.rat.database_replay.encryption (大/小文字を区別)識別子を使用してパスワードを設定する必要があります。パスワードはソフトウェア・キーストアに格納されます。ソフトウェア・キーストアの作成の詳細は、『Oracle Database Advanced Security ガイド』を参照してください。

例: パスワードベースのソフトウェア・キーストアの設定

次の文は、パスワードベースのソフトウェア・キーストアを作成します。

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'MYKEYSTORE' IDENTIFIED BY password;
```

次の文は、パスワードベースのソフトウェア・キーストアを開き、パスワードベースのソフトウェア・キーストアのバックアップを作成します。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY password;
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password WITH BACKUP;
```

次の文は、タグDBREPLAYを指定してクライアント'oracle.rat.database_replay.encryption'のシークレットsecret_keyをパスワードベースのソフトウェア・キーストアに追加します。シークレットを追加する前に、パスワードベースのソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT ADD SECRET secret_key FOR CLIENT 'oracle.rat.database_replay.encryption'
USING TAG 'DBREPLAY'
IDENTIFIED BY password WITH BACKUP;
```

次の文は、パスワードベースのソフトウェア・キーストアを閉じます。

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY password;
```

ノート:

データベース取得およびデータベース・リプレイを実行するには、ソフトウェア・キーストアをオープンしたままにしておく必要があります。

ワークロードの取得の停止

この項では、ワークロード取得を停止する方法について説明します。

ワークロード取得を停止するには:

- FINISH_CAPTUREプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.FINISH_CAPTURE ();
END;
/
```

この例では、FINISH_CAPTUREプロシージャは、ワークロードの取得を終了し、データベースを通常の状態に戻します。

ヒント:

本番システムでワークロードを取得したら、取得したワークロードを事前処理する必要があります。詳細は、「データベース・ワークロードの事前処理」を参照してください。

ワークロードの取得のAWRデータのエクスポート

AWRデータをエクスポートすると、ワークロードの詳細な分析が可能になります。このデータは、2つのワークロードの取得(またはリプレイ)に対して、リプレイの期間比較レポートまたはAWR期間比較レポートを実行する場合には必須です。

AWRデータをエクスポートするには:

- EXPORT_AWRプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.EXPORT_AWR (capture_id => 2);
END;
/
```

この例では、取得IDが2のワークロード取得に対応するAWRスナップショットがエクスポートされます。また、ワークロードの取得中に取得されたSQLチューニング・セットもエクスポートされます。

EXPORT_AWRプロシージャでは、AWRスナップショットをエクスポートする取得のIDを指定する必須パラメータcapture_idを使用します。capture_idパラメータの値がDBA_WORKLOAD_CAPTURESビューのID列に表示されます。

ノート:

このプロシージャは、対応するワークロードの取得が現在のデータベースで実行され、元の取得期間に対

応するAWRスナップショットがまだ使用可能である場合にのみ、機能します。

関連項目:

DBA_WORKLOAD_CAPTURESビューの詳細は、『Oracle Databaseリファレンス』を参照してください

ワークロードの取得のAWRデータのインポート

AWRデータは取得システムからエクスポートすると、取得済ワークロードをリプレイするテスト・システムなどの別のシステムにインポートできます。AWRデータをインポートすると、ワークロードの詳細な分析が可能になります。このデータは、2つのワークロードの取得(またはリプレイ)に対して、リプレイの期間比較レポートまたはAWR期間比較レポートを実行する場合には必須です。

AWRデータをインポートするには:

- 次の例に従って、IMPORT_AWRファンクションを使用します。

```
CREATE USER capture_awr
SELECT DBMS_WORKLOAD_CAPTURE.IMPORT_AWR (capture_id => 2,
                                         staging_schema => 'capture_awr')
  FROM DUAL;
```

この例では、取得IDが2のワークロード取得に対応するAWRスナップショットがcapture_awrという名前のステージング・スキーマを使用してインポートされます。

この例のIMPORT_AWRプロシージャでは、次のパラメータを使用します。

- 必須パラメータcapture_idでは、AWRスナップショットをインポートする取得のIDを指定します。capture_idパラメータの値がDBA_WORKLOAD_CAPTURESビューのID列に表示されます。
- 必須パラメータstaging_schemaでは、取得ディレクトリからSYS AWRスキーマにAWRスナップショットをインポートする際に、ステージング領域として使用される現在のデータベースの既存のスキーマ名を指定します。

ノート:

staging_schema パラメータで指定したスキーマに AWR 表と同じ名前の表が含まれる場合、このファンクションは失敗します。

関連項目:

DBA_WORKLOAD_CAPTURESビューの詳細は、『Oracle Databaseリファレンス』を参照してください

APIの使用による既存のワークロード取得の暗号化および復号化

この項では、APIを使用して既存のワークロード取得を暗号化および復号化する方法について説明します。

ワークロード取得時には、接続文字列やSQLテキスト、バインド値といった各種情報が保存されます。機密データが含まれている場合、この情報は暗号化できます。「ワークロードの取得の開始」で説明されているとおりに、ワークロード取得時に暗号化を

有効にできます。

既存のワークロード取得の暗号化

この項では、既存のワークロード取得を暗号化する方法について説明します。

既存のワークロード取得を暗号化するには:

- ENCRYPT_CAPTUREプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.ENCRYPT_CAPTURE(src_dir => 'dec10',
                                        dst_dir => 'dec10_enc',
                                        encryption => 'AES128');
END;
/
```

この例のENCRYPT_CAPTUREプロシージャでは、次のパラメータを使用します。

- src_dirパラメータは、暗号化するワークロード取得が格納されているディレクトリを指しています。
- dst_dirパラメータは、暗号化された取得が暗号化後に保存されるディレクトリを指しています。
- encryptionパラメータでは、ワークロード取得の暗号化に使用されるアルゴリズムを指定します。

ノート:

DBMS_WORKLOAD_CAPTURE.ENCRYPT_CAPTURE を実行する前に、ソフトウェア・キーストアに oracle.rat.database_replay.encryption (大/小文字を区別)識別子を格納しておく必要があります。

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

暗号化されたワークロード取得の復号化

この項では、暗号化されたワークロード取得を復号化する方法について説明します。

暗号化されたワークロード取得は、DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTUREプロシージャを使用して復号化できます。

暗号化されたワークロード取得を復号化するには:

- DECRYPT_CAPTUREプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTURE(src_dir => 'dec10_enc',
                                        dst_dir => 'dec10');
END;
/
```

この例のDECRYPT_CAPTUREプロシージャでは、次のパラメータを使用します。

- src_dirパラメータは、暗号化された取得が格納されているディレクトリを指しています。
- dst_dirパラメータは、復号化された取得が復号化後に保存されるディレクトリを指しています。

ノート:

DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTURE を実行する前に、ソフトウェア・キーストアに oracle.rat.database_replay.encryption (大/小文字を区別)識別子を格納しておく必要があります。

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

ビューを使用したワークロードの取得の監視

この項では、ワークロードの取得を監視するために表示できるビューの概要を示します。また、「Enterprise Managerを使用したワークロードの取得の監視」で説明されているように、Oracle Enterprise Managerを使用してワークロードの取得を監視できます。

これらのビューにアクセスするには、次のDBA権限が必要です。

- DBA_WORKLOAD_CAPTURESビュー: 現在のデータベースで取得されたワークロードの取得をすべて示します。
- DBA_WORKLOAD_FILTERSビュー: 現在のデータベースに定義されたワークロードの取得に対して使用されるワークロード・フィルタをすべて示します。

関連項目:

- DBA_WORKLOAD_CAPTURESビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- DBA_WORKLOAD_FILTERSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

11 データベース・ワークロードの事前処理

ワークロードを取得し、テスト・システムの設定が完了したら、取得したデータを事前処理する必要があります。取得されたワークロードの事前処理によって、ワークロードのリプレイに必要なすべてのメタデータが作成されます。事前処理は、取得したワークロードをリプレイする前に、ワークロードごとに1回行う必要があります。取得したワークロードは、事前処理を行うと、リプレイ・システムで繰り返しリプレイできるようになります。

取得したワークロードを事前処理するには、まず、取得したすべてのデータ・ファイルを取得システムの格納先のディレクトリから、事前処理を実行するインスタンスのディレクトリに移動する必要があります。事前処理は、リソースを大量に必要とするため、次の条件を満たすシステムで実行する必要があります。

- 本番システムから切り離されている
- リプレイ・システムと同じバージョンのOracle Databaseが実行されている

Oracle Real Application Clusters(Oracle RAC)では、事前処理用にリプレイ・システムのデータベース・インスタンスを1つ選択します。このインスタンスは、ローカル・ファイル・システムまたは共有ファイル・システムに格納可能な、事前処理を必要とする取得済データ・ファイルにアクセスできる必要があります。取得システムの取得ディレクトリのパスが各インスタンスで別の物理ディレクトリに解決される場合、それらを前処理が実行される1つの取得ディレクトリにマージする必要があります。すべてのディレクトリには、同じディレクトリ・ツリーが必要であり、これらのディレクトリに含まれるすべてのファイルは、取得ディレクトリと同じ相対パスを持つディレクトリに移動される必要があります。

通常、取得したワークロードはリプレイ・システムで事前処理します。取得したワークロードをリプレイ・システムから切り離されたシステムで事前処理する場合は、事前処理完了後に、すべての事前処理済データ・ファイルを、事前処理を行ったシステムの格納先ディレクトリからリプレイ・システムのディレクトリに移動する必要があります。

この章の構成は、次のとおりです。

Enterprise Managerを使用した単一のデータベース・ワークロードの準備

単一のワークロードの準備には、いくつかのタスクが含まれています。たとえば:

- データベース・リプレイ・タスクの作成
- リプレイ・タスクからのリプレイの作成
- テスト・データベースの準備
- ワークロードの前処理とリプレイ・クライアントのデプロイ

取得したワークロードを事前処理する前に、本番システムでワークロードを取得する必要があります。詳細は、「データベース・ワークロードの取得」を参照してください。

ノート:

テスト・データベースの準備が必要なのは、まだ準備を行っていない場合のみです。

この後の各項では、これらのタスクの実行方法について説明します。

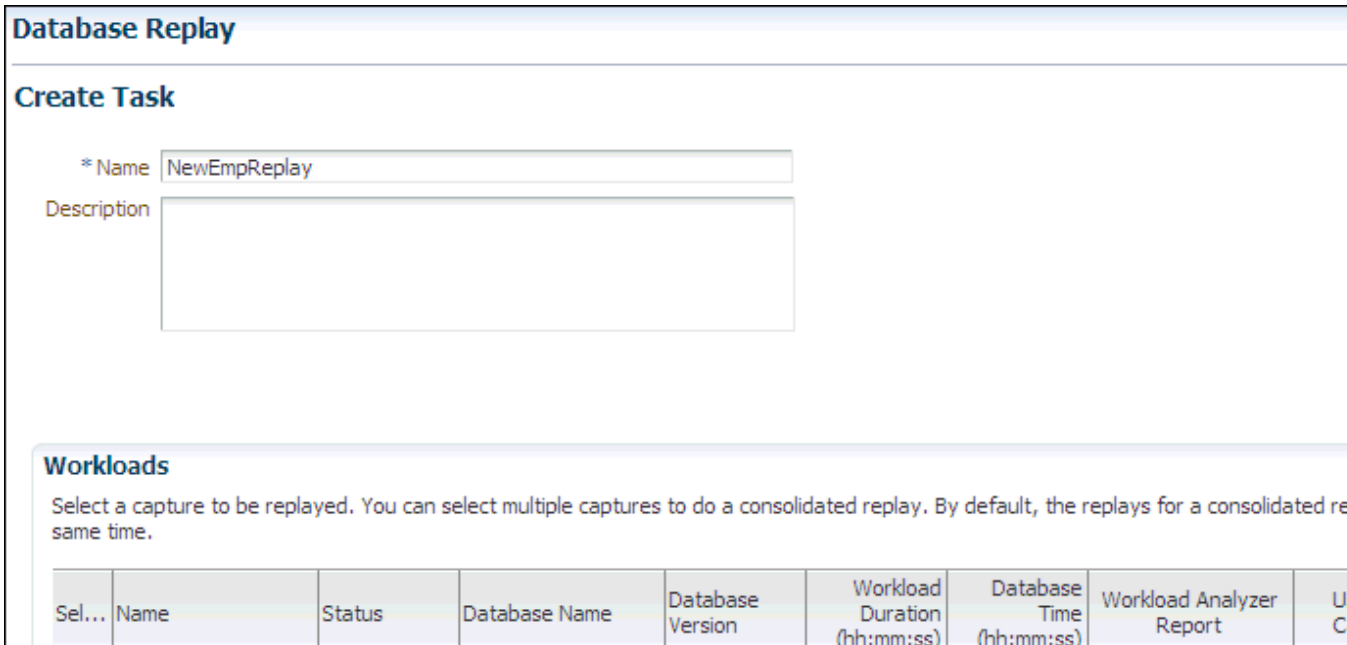
データベース・リプレイ・タスクの作成

データベース・リプレイ・タスクを作成する前に、リプレイする取得に取得済ユーザー・コールがあることを確認します。

データベース・リプレイ・タスクを作成するには:

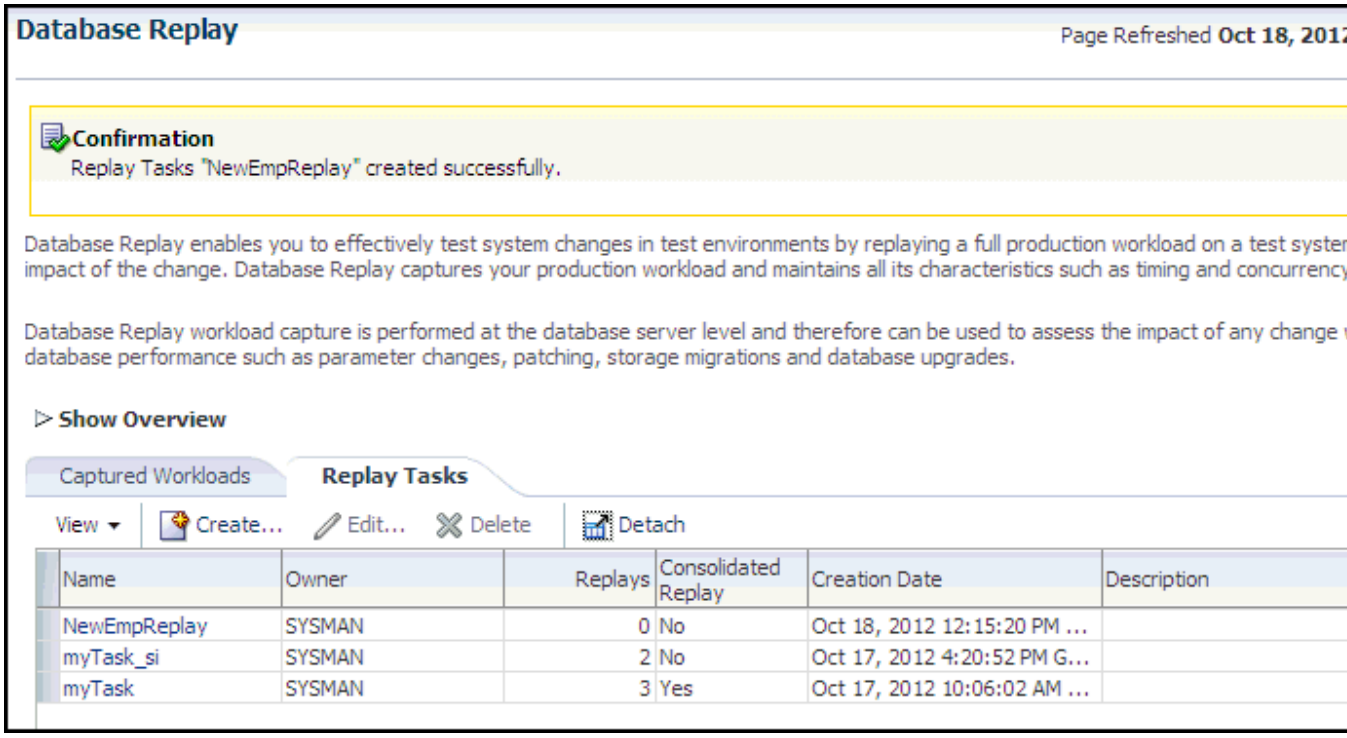
1. データベース・リプレイ・ページで「リプレイ・タスク」タブをクリックし、ツールバーの「作成」をクリックします。

   タスクの作成ページが表示されます。



2. タスクの名前を指定し、リプレイする取得を選択して、「発行」をクリックします。統合されたリプレイで、2つ以上の取得を選択します。

   データベース・リプレイ・ページが再度表示され、「リプレイ・タスク」タブの表に新しく作成されたリプレイ・タスクが表示されます。



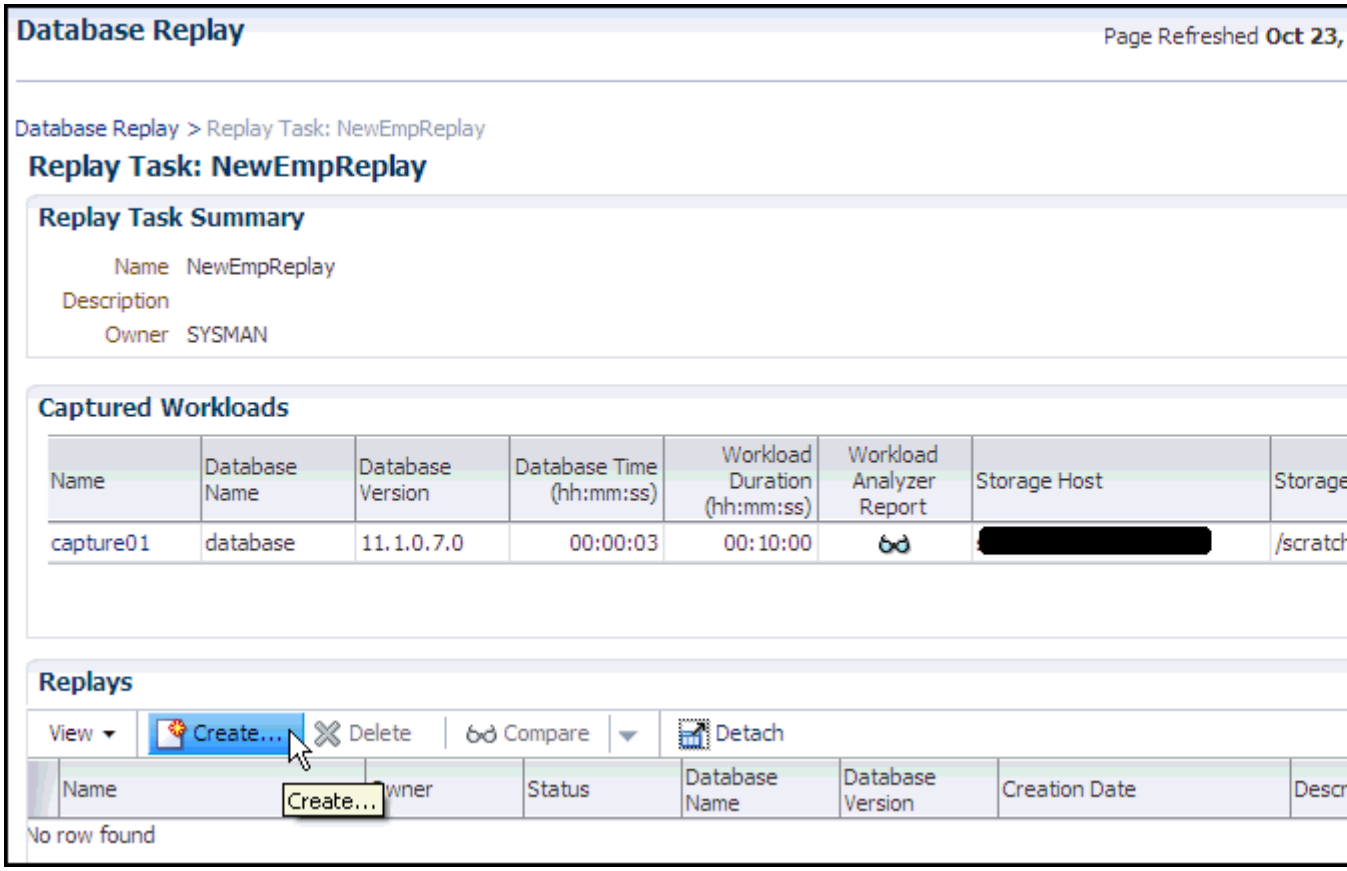
リプレイ・タスクからのリプレイの作成

ここでは、リプレイ・タスクからリプレイを作成する方法について説明します。

リプレイを作成するには:

1. データベース・リプレイ・ページで「リプレイ・タスク」タブをクリックします。
2. 表の必要なリプレイ・タスクのリンクをクリックします。

   取得のリプレイ・タスク・ページが表示されます。



3. 「リプレイ」セクションの「作成」をクリックします。

   「リプレイの作成」ポップアップが表示されます。

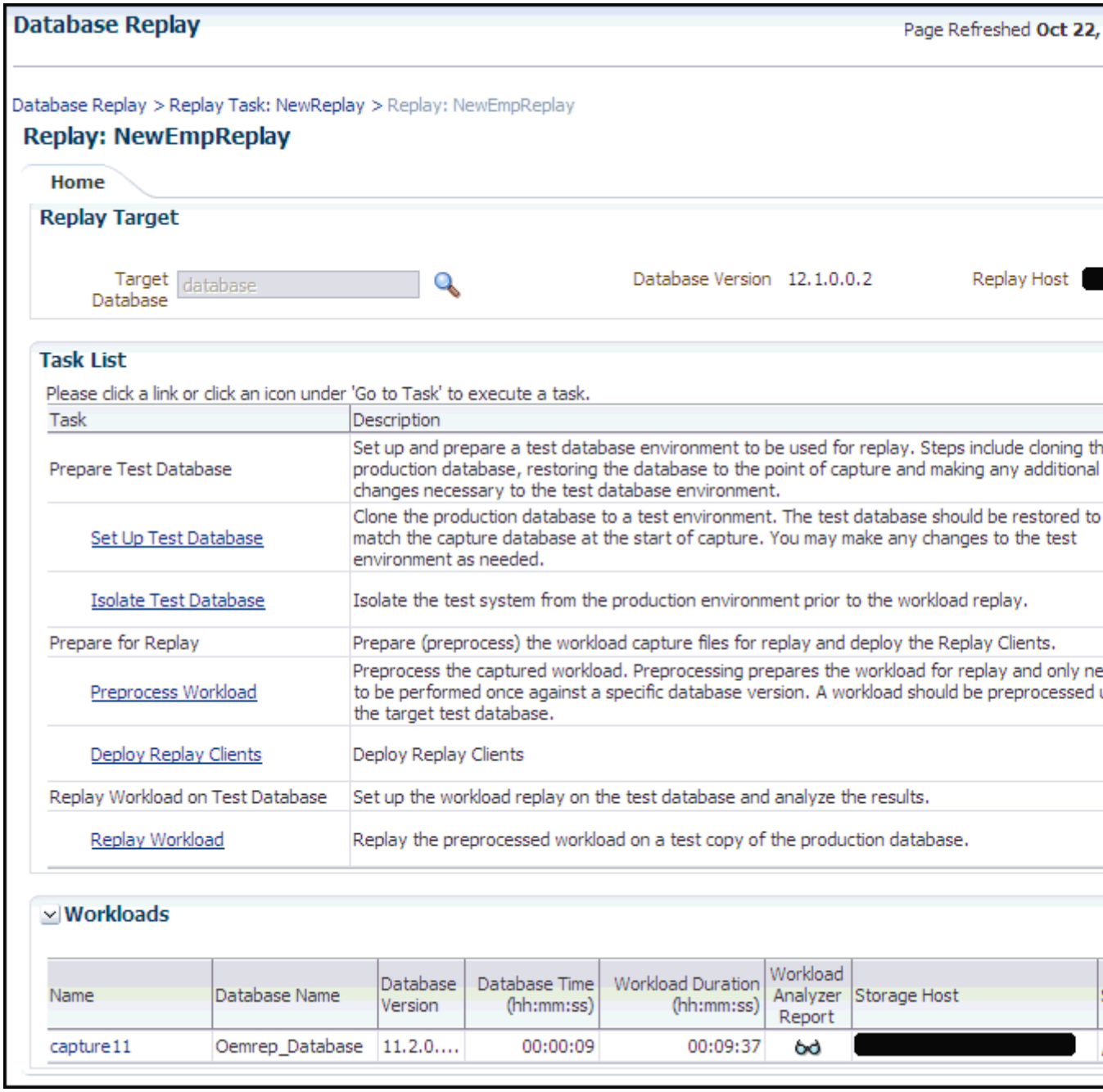
4. 必要な名前と説明(オプション)を入力し、「ターゲット・データベース」アイコンをクリックします。

   「検索と選択: ターゲット」ポップアップが表示されます。

5. 必要なデータベースを選択して、「選択」をクリックします。

6. 「リプレイの作成」ポップアップの「OK」をクリックします。

   必要なタスクを実行するリンク付きの「タスク・リスト」が含まれるリプレイのデータベース・リプレイ・ページが表示されます。



次の項で説明されている、「タスク・リスト」の最初のタスクに進みます。

テスト・データベースの準備

ここでは、テスト・データベースの準備に含まれているタスクについて説明します。たとえば:

- テスト・データベースの設定
- テスト・データベースの分離

ノート:

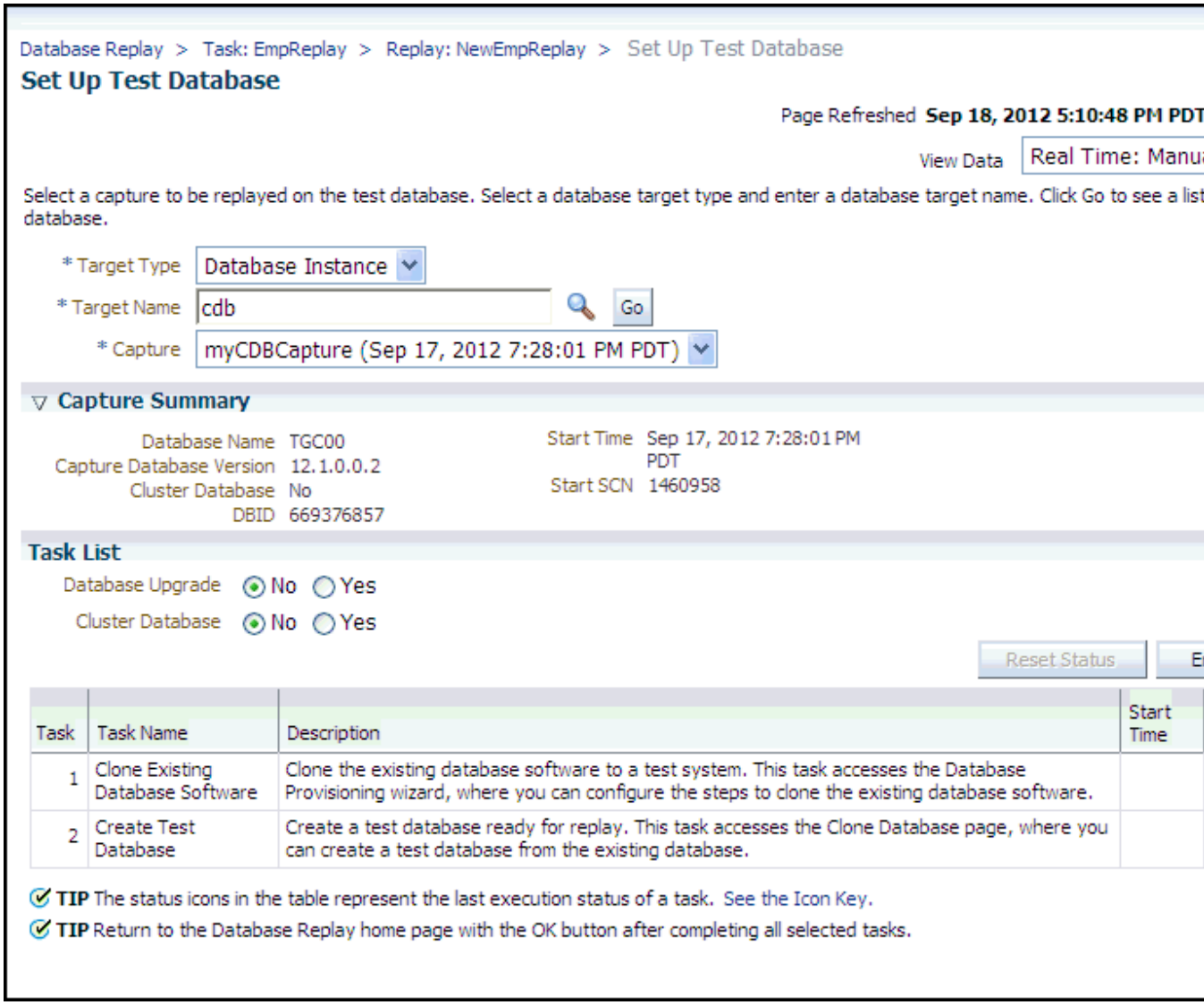
このタスクは任意で実行します。テスト・データベースをすでに設定している場合、「ワークロードの前処理とリプレイ・クライアントのデプロイ」に進みます。

次の手順は、各タスクの実行方法を説明しており、どのような順番で実行してもかまいません。

テスト・データベースを設定するには:

1. 特定のリプレイのリプレイ・ページで、「テスト・データベースの設定」タスクのリンクをクリックします。

   テスト・データベースの設定ページが表示されます。



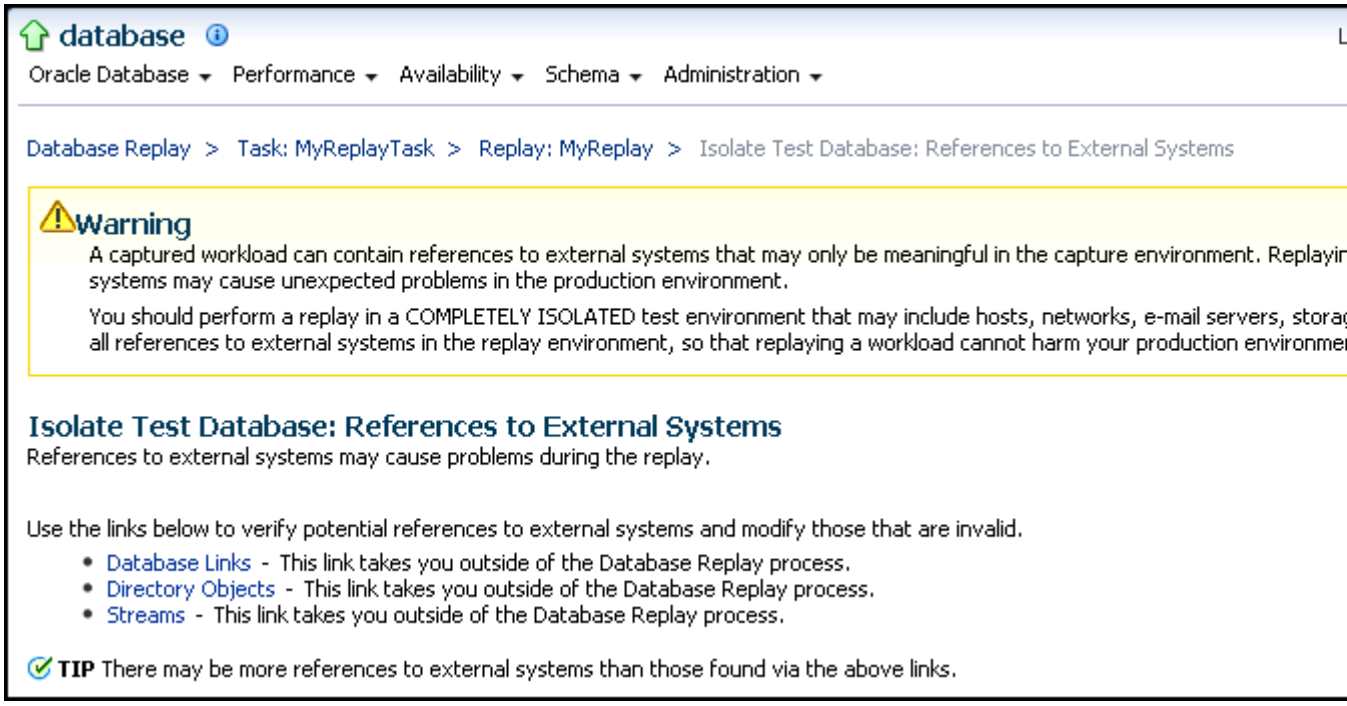
2. データベースをアップグレードするかどうかを選択し、クラスタ・データベースであるかどうかを示します。
3. 「既存データベース・ソフトウェアのクローニング」サブタスクの「タスクに移動」アイコンをクリックするか、先にテスト・データベースを作成する場合は「全タスクの有効化」をクリックします。
4. ウィザードのオンライン・ヘルプに記載されている指示に従います。

   タスクが完了すると、各タスクの「ステータス」列にチェックマークが表示されます。
5. データベース・リプレイ・ページに戻るには、「OK」をクリックします。

テスト・データベースを分離するには:

1. 特定のリプレイのリプレイ・ページで、「テスト・データベースの分離」タスクのリンクをクリックします。

   外部システムへの参照はリプレイ中に問題の原因になる可能性があることを説明するページが表示されます。



2. 表示されるリンクを使用して、可能性のある外部システムへの参照を検証し、無効な参照を変更して「OK」をクリックします。

   リプレイ・サマリー・ページが再度表示されます。

ワークロードの前処理とリプレイ・クライアントのデプロイ

リプレイの最後の準備は、ワークロードの処理およびリプレイ・クライアントのデプロイです。たとえば:

- ワークロードの事前処理

  取得済の各ワークロードに対し、ワークロードをリプレイするデータベースのバージョンごとに1回前処理を実行します。ワークロードを前処理したら、テスト・データベースのバージョンがワークロードを前処理したバージョンと同じであるかぎり、それ以降のリプレイ・タスクでは、再度前処理をする必要なくそれをリプレイできます。たとえば、MyReplay1およびMyReplay2という2つのリプレイ名を持つリプレイ・タスクがある場合、MyReplay1を前処理した後は、ディレクトリ・オブジェクトを直接再使用するだけでMyReplay2をリプレイできます。

  前処理には、ワークロード・アナライザ・レポートを使用できます。

- リプレイ・クライアントのデプロイ

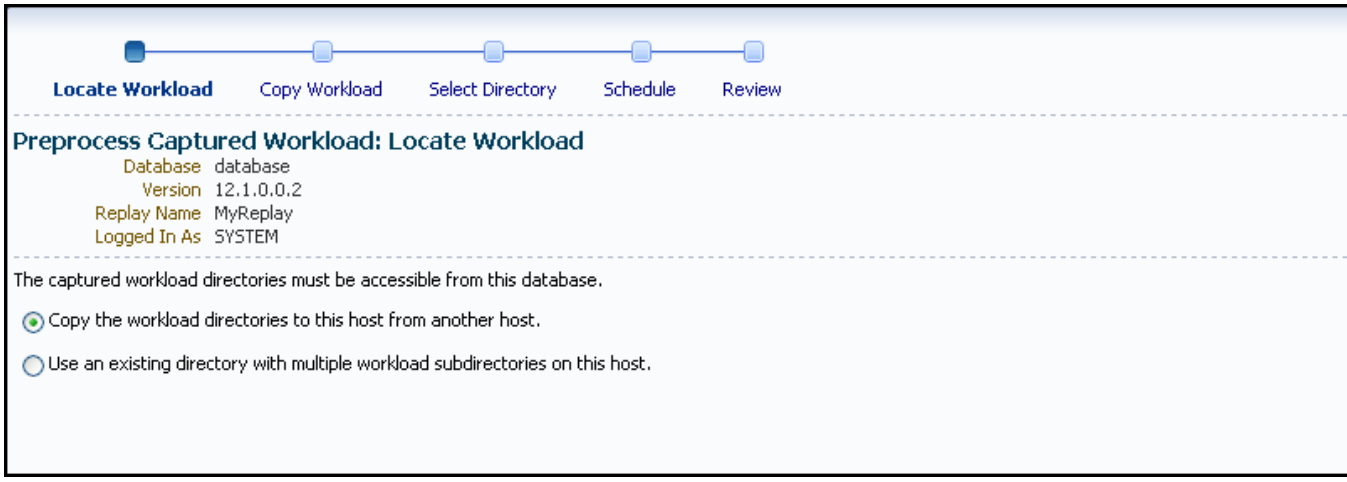
  リプレイ・クライアント・ホストが、「データベース・ターゲット名」フィールドに指定したテスト・データベースのOracleホームにアクセスできる場合は、リプレイ・クライアントを別のリプレイ・クライアント・ホストにデプロイする必要はありません。

次の手順は、各タスクを完了する方法を説明しています。

ワークロードを事前処理するには:

1. 特定のリプレイのリプレイ・ページで、「ワークロードの事前処理」タスクのリンクをクリックします。

   「取得されたワークロードの前処理: ワークロードの検索」ページが表示されます。

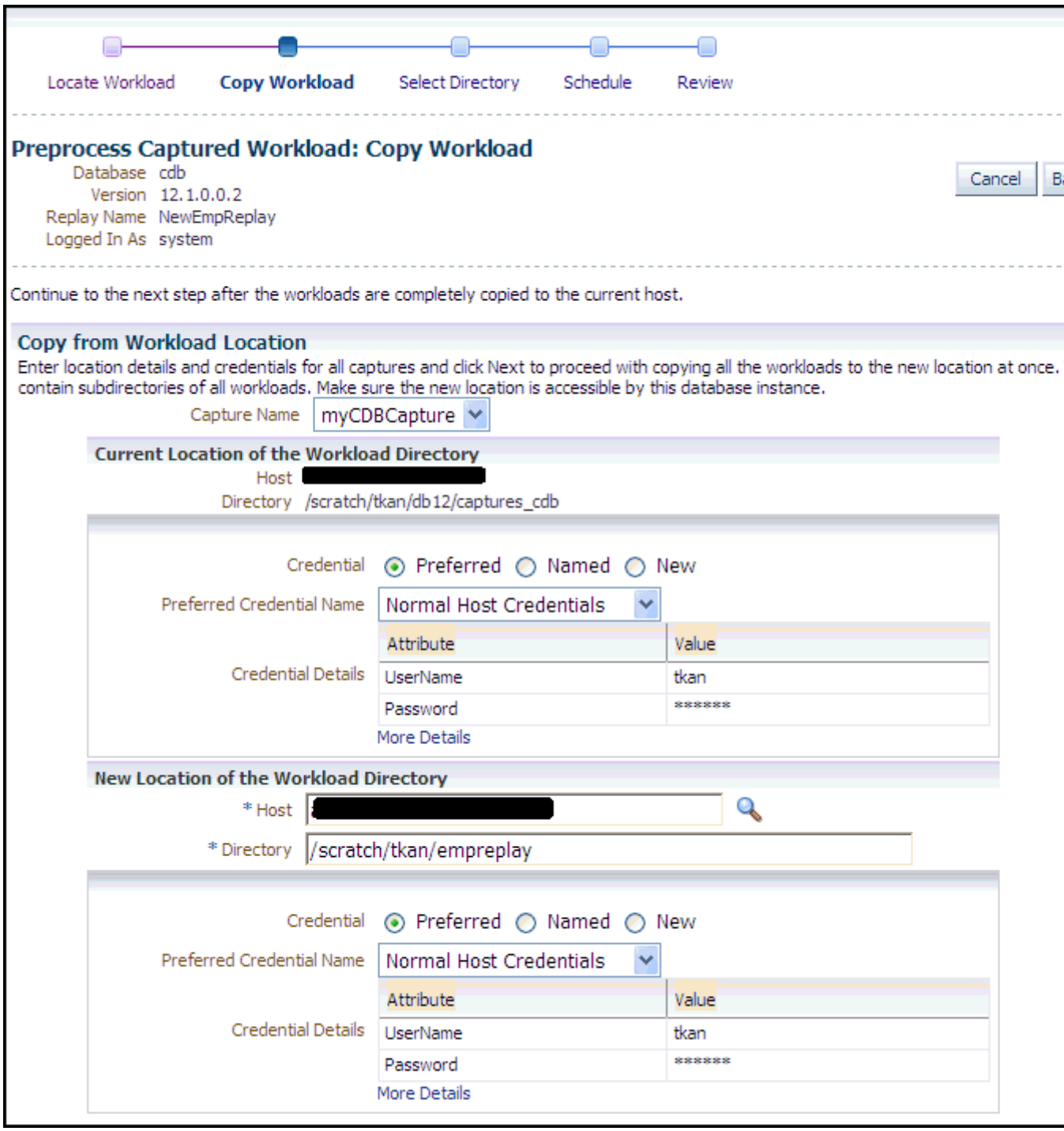


2. 必要なワークロードの場所オプションを選択して、「次」をクリックします。

ノート:

最初はコピー・オプションを選択する必要があります。

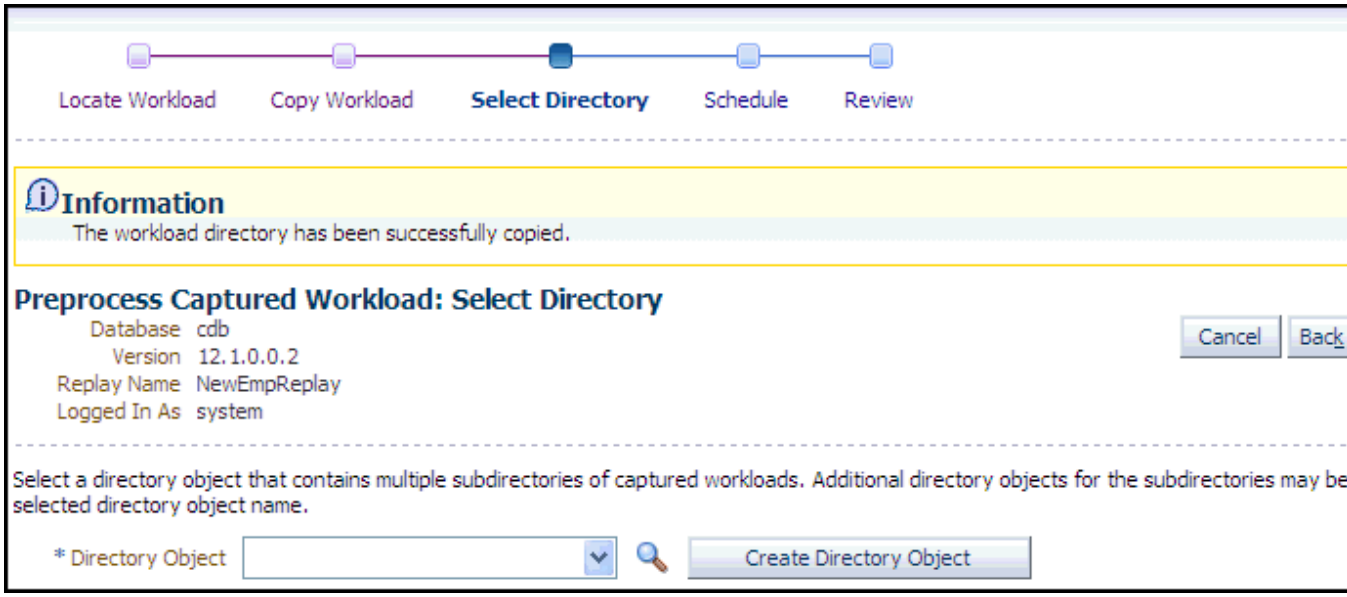
「取得されたワークロードの前処理: ワークロードのコピー」ページが表示されます。



3. 必要な資格証明およびワークロードのコピーおよび前処理先の新しい場所を指定し、「次」をクリックします。

   - 統合されたリプレイには複数のソース・ワークロードがあるため、複数のソース資格証明をワークロード・ディレクトリの現在の場所に入力しなければならないことがあります。統合リプレイの詳細は、「Enterprise Managerを使用したデータベース統合リプレイの使用」を参照してください。

   システムは処理中に進捗状況の棒グラフを表示して応答し、コピー操作が終了した後に取得されたワークロードの前処理: ディレクトリを選択ページが表示されます。



4. ディレクトリ・オブジェクトを指定するかワークロードが含まれる場所をポイントする新しいディレクトリ・オブジェクトを作成します。前のステップでワークロードを新しい場所にコピーするよう選択をした場合、ディレクトリ・オブジェクトが「ワークロード・ディレクトリの新しい場所」で指定した正しい場所を確実にポイントするようにします。

   システムは取得サマリーを表示して応答します。「詳細の取得」セクションを展開して、ワークロード・プロファイルとワークロード・フィルタを表示できます。統合されたリプレイに取得サマリーは表示されません。

   「次へ」をクリックして、「取得されたワークロードの前処理: スケジュール」ページを表示します。

5. 前処理ジョブのスケジュールを入力します。

   - 必要な独自のジョブ名を指定するか、システム提供の名前をそのまま使用します。ジョブ・システムは、自動的に大文字でジョブに名前を付けます。
   - ジョブを発行後すぐに実行するか、または後で実行するかを示します。
   - ホスト資格証明を指定します。これは、オペレーティング・システムでの前処理ジョブの実行に使用されます。

   「次へ」をクリックし、「取得されたワークロードの前処理: 確認」ページを表示します。

6. 意図したとおりに設定されていることを確認して、「発行」をクリックします。

   データベース・リプレイ・ページが表示され、入力にエラーがないものと見なされ、ページ上部の確認メッセージに「ワークロードの準備ジョブJOBNAMEは正常に作成されました。」と表示されます。

7. 「JOBNAME」リンクをクリックして、ジョブのステータスを確認します。「ワークロード・リプレイ」タスクに進む前に、ジョブが完了している必要があります。

ノート:

試行後に期間比較レポートを生成するには、テスト・データベースに追加の PL/SQL パッケージをインストールする必要があるという内容のメッセージが、「タスク・リスト」に表示される場合があります。「ワークロード・リプレイ」タスクに進む前に、この問題を解決するには、「PL/SQL パッケージのインストール」をクリックします。

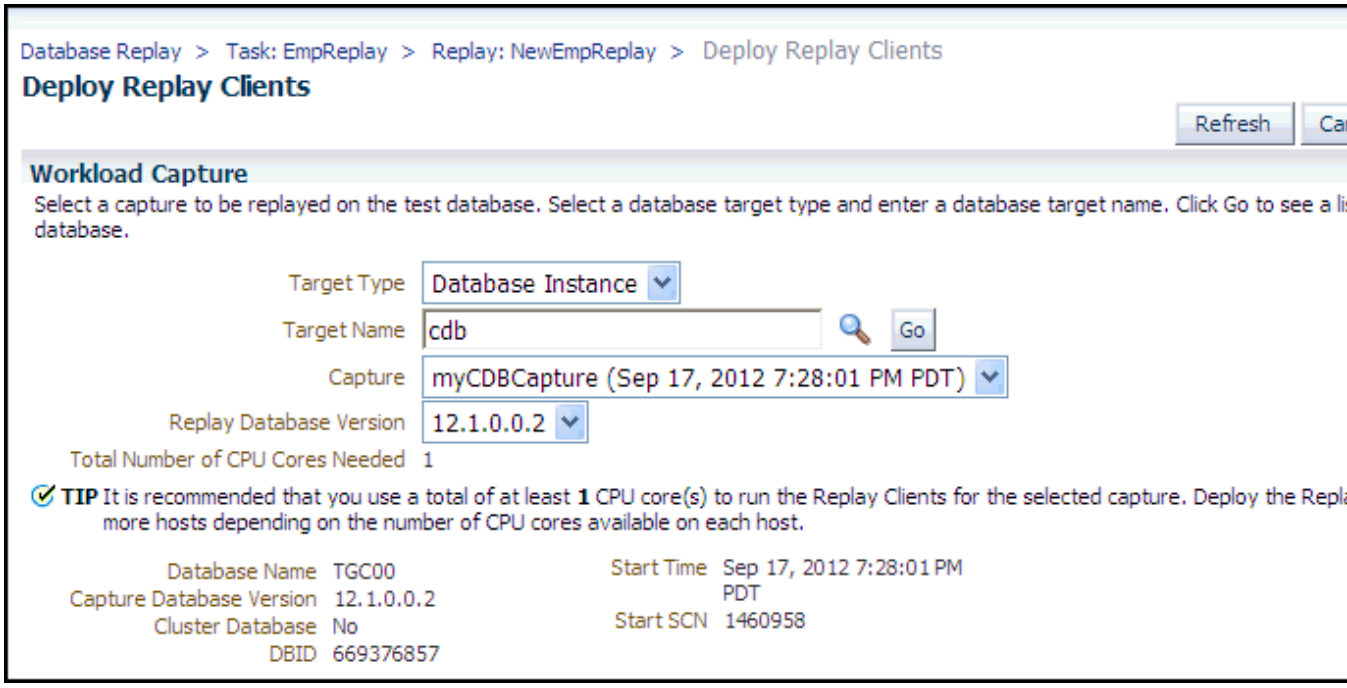
ヒント:

取得したワークロードを事前処理したら、テスト・システムでリプレイできます。詳細は、「データベース・ワークロードのリプレイ」を参照してください。

リプレイ・クライアントをデプロイするには:

1. 特定のリプレイのリプレイ・ページで、「リプレイ・クライアントのデプロイ」タスクのリンクをクリックします。

   リプレイ・クライアントのデプロイ・ページが表示されます。



2. 関連するワークロード取得に定義されているデフォルト値をそのまま使用するか、それらの値をオーバーライドして「続行」をクリックします。

   Oracleデータベース・クライアントのプロビジョニング・ウィザードが表示されます。

3. ウィザードの各ステップのオンライン・ヘルプに記載されている指示に従います。

   「確認」ステップで「発行」をクリックすると、設定したスケジュールに従ってデプロイメント・プロシージャが実行され、リプレイ・サマリー・ページが再度表示されます。

APIを使用したデータベース・ワークロードの事前処理

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して取得したワークロードを事前処理する方法について説明します。また、「Enterprise Managerを使用した単一のデータベース・ワークロードの準備」で説明されているように、Oracle Enterprise Managerを使用して取得済ワークロードを前処理することも可能です。
取得したワークロードを事前処理する前に、本番システムでワークロードを取得する必要があります。詳細は、「データベース・ワークロードの取得」を参照してください。

取得済のワークロードを処理するには:

- PROCESS_CAPTUREプロシージャを使用します。

```
BEGIN
```

```
  DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE (capture_dir => 'dec06',
                                        plsql_mode => 'extended');
END;
/
```

この例では、dec06ディレクトリに格納されている取得済ワークロードが事前処理されます。

この例のPROCESS_CAPTUREプロシージャでは、必須パラメータcapture_dirを使用します。このパラメータは、事前処理する取得済ワークロードが含まれているディレクトリを指定します。

オプションのplsql_modeパラメータで、PL/SQLの処理モードを指定します。

plsql_modeパラメータには次の2つの値を設定できます。

- top_level: メタデータは最上位レベルのPL/SQLコールに対してのみ生成されます。これはリプレイの唯一のオプションになります。これがデフォルト値です。
- extended: メタデータは最上位レベルのPL/SQLコールとPL/SQLからコールされたSQLの両方に対して生成されます。新しいディレクトリppe_X.X.X.X (Xは現在のOracleバージョンを表す)は、取得ルートディレクトリの下に作成されます。取得は、plsql_modeパラメータのこの同じ値で実行されている必要があります。リプレイでは'TOP_LEVEL'または'EXTENDED'のいずれかを使用できます。

  extended値は、plsql_modeパラメータをextendedに設定して取得されたワークロードにのみ設定できます。extendedを指定しても、取得がextendedモードで実行されなかった場合は、エラー・メッセージが表示されます。

ノート:

暗号化されたワークロード取得で PROCESS_CAPTURE を実行するには、識別子 oracle.rat.database_replay.encryption (大/小文字を区別)を使用してパスワードを設定する必要があります。パスワードはソフトウェア・キーストアに格納されます。ワークロード取得が暗号化されているかどうかは、DBA_WORKLOAD_CAPTURES ビューから確認できます。

ヒント:

取得したワークロードを事前処理したら、テスト・システムでリプレイできます。詳細は、「データベース・ワークロードのリプレイ」を参照してください。

関連項目:

- DBMS_WORKLOAD_CAPTUREパッケージに対するSTART_CAPTUREプロシージャのplsql_modeパラメータの詳細は、「ワークロードの取得の開始」を参照してください。
- DBMS_WORKLOAD_REPLAY.PROCESS_CAPTUREプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ワークロード・アナライザのコマンドライン・インタフェースの実行

ワークロード・アナライザは、ワークロード取得ディレクトリを分析して、取得ワークロードのうち、データ不足、ワークロード取得中に発生したエラー、データベース・リプレイによってサポートされていない使用機能などのために正確にリプレイできない部分を識別するJavaプログラムです。ワークロード分析の結果は、分析対象である取得ディレクトリ内に置かれるwcr_cap_analysis.htmlという名前のHTMLレポートに保存されます。エラーを防ぐことができる場合、ワークロード分析レポートにはリプレイ前に実行できる予防措置が示されます。エラーを修正できない場合はワークロード分析レポートにそのエラーに関する記述が加えられるので、リプレイ時にそのエラーを考慮しておくことができます。ワークロード・アナライザの実行はデフォルトのオプションであり、実行することを強く推奨します。

ノート:

Oracle Enterprise Managerを使用してワークロードの取得を事前処理する場合は、コマンドライン・インタフェースでワークロード・アナライザを実行する必要はありません。Oracle Enterprise Managerでは、ワークロードの事前処理の一環としてワークロード・アナライザを実行できます。

ワークロード・アナライザは、Oracle Database Enterprise Editionリリース11.2.0.2以上が実行されているシステムの$ORACLE_HOME/rdbms/jlib/ディレクトリにある2つのJARファイル(dbranalyzer.jarおよびdbrparser.jar)で構成されます。ワークロード・アナライザには、Java 1.5以上および$ORACLE_HOME/jdbc/lib/ディレクトリにojdbc6.jarファイルが必要です。

ワークロード・アナライザを実行するには:

1. コマンドライン・インタフェースで、1行で次のjavaコマンドを実行します。

```
java -classpath
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/rdbms/jlib/dbranalyzer.jar:
oracle.dbreplay.workload.checker.CaptureChecker
<capture_directory> <connection_string>
```

   capture_directoryパラメータには、取得ディレクトリのオペレーティング・システム・パスを入力します。このディレクトリには、ワークロード取得のAWRデータもエクスポートされている必要があります。connection_stringパラメータには、リリース11.1以上のOracle Databaseの接続文字列を入力します。

   このコマンドの例は次のとおりです。

```
java -classpath
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/rdbms/jlib/dbranalyzer.jar:
oracle.dbreplay.workload.checker.CaptureChecker /scratch/capture
jdbc:oracle:thin:@myhost.mycompany.com:1521:orcl
```

2. 求められたら、ターゲット・データベースのDBMS_WORKLOAD_CAPTUREパッケージのEXECUTE権限およびターゲット・データベースのSELECT_CATALOGロールを持つデータベース・ユーザーのユーザー名とパスワードを入力します。

12 データベース・ワークロードのリプレイ

取得されたワークロードの事前処理後に、Oracle Databaseの同じバージョンを稼働しているリプレイ・システムで繰り返しリプレイすることができます。通常は、事前処理されたワークロードがリプレイされるリプレイ・システムは、本番システムとは別のテスト・システムである必要があります。

この章では、テスト・システムでデータベース・ワークロードをリプレイする方法について説明します。内容は次のとおりです。

- データベース・ワークロードのリプレイのステップ
- Enterprise Managerを使用したデータベース・ワークロードのリプレイ
- Enterprise Managerを使用したリプレイ・スケジュールおよびパラメータの設定
- Enterprise Managerを使用したワークロードのリプレイの監視
- Enterprise Manager外部のリプレイのインポート
- APIを使用したデータベース・ワークロードのリプレイ
- APIを使用したワークロードのリプレイの監視

ヒント:

データベース・ワークロードをリプレイする前に、次の操作を行う必要があります。

- 本番システムでのワークロードの取得(「データベース・ワークロードの取得」)
- 取得されたワークロードの事前処理(「データベース・ワークロードの事前処理」)

データベース・ワークロードのリプレイのステップ

リプレイ・システムを適切に準備し、ワークロードのリプレイを適切に計画すると、リプレイの正確性が保証されます。データベースのワークロードをリプレイする前に、次のステップを完了して、リプレイ・システムとワークロードのリプレイを準備します。

- リプレイ・ディレクトリの設定
- データベースのリストア
- 外部システムへの参照の解決
- 接続の再マッピング
- ユーザーの再マッピング
- リプレイ・オプションの指定
- ワークロード・リプレイ時のフィルタの使用
- リプレイ・クライアントの設定

リプレイ・ディレクトリの設定

取得したワークロードは、事前処理してリプレイ・システムにコピーしておく必要があります。事前処理したワークロードのコピー先ディレクトリのディレクトリ・オブジェクトが、リプレイ・システムに存在している必要があります。

データベースのリストア

ワークロードをリプレイする前に、リプレイ・システムのアプリケーション・データの状態を、ワークロードの取得の開始時における取得システムのデータの状態と論理的に同じにする必要があります。これによって、リプレイ時のリプレイの相違が最小限に抑えられます。データベースのリストア方法は、ワークロードの取得前に使用したバックアップ方法に応じて異なります。たとえば、取得システムのバックアップにRMANを使用した場合は、テスト・データベースの作成にRMANのDUPLICATE機能を使用できます。詳細は、「データベース・ワークロードの取得の前提条件」を参照してください。

リプレイ・システムで適切なアプリケーション・データを使用してデータベースを作成したら、テストするシステム変更(データベースやオペレーティング・システムのアップグレードなど)を行います。データベース・リプレイの主な目的は、取得したワークロードでシステム変更の影響をテストすることです。したがって、実行したシステム変更によって、取得したワークロードで実施するテストが定義されることになります。

外部システムへの参照の解決

取得したワークロードに、データベース・リンクや外部表などの外部システムへの参照が含まれている場合があります。通常、このような外部との対話は、リプレイ時に他の本番システムに影響が及ばないように再構成する必要があります。ワークロードのリプレイ前に解決する必要がある外部参照は、次のとおりです。

- データベース・リンク

  通常、リプレイ・システムが他のデータベースと相互作用するのは好ましくありません。したがって、リプレイに必要なデータが含まれている適切なデータベースを指すように、すべてのデータベース・リンクを再構成する必要があります。

- 外部表

  外部表によって参照されるディレクトリ・オブジェクトを試用して指定されたすべての外部ファイルは、リプレイ時にデータベースに対して使用可能である必要があります。これらのファイルのコンテンツは、取得時と同一である必要があり、外部表の定義に使用されるファイル名およびディレクトリ・オブジェクトも有効である必要があります。

- ディレクトリ・オブジェクト

  本番システムのディレクトリへの参照は、データベースのリストア後にリプレイ・システムに存在するディレクトリ・オブジェクトを適切に再定義することによって再構成する必要があります。

- URL

  ワークロードの取得時にアクセスしたWebサービスがリプレイ時に適切なURLを指すように、データベースに格納されているURL/URIを構成する必要があります。ワークロードが本番システムに格納されているURLを参照する場合、リプレイ時にそのテスト・システム・ネットワークを切り離す必要があります。

- 電子メール

  リプレイ時に電子メール通知が再送信されないようにするには、送信電子メールのリクエストを無視するように、リプレイ・システムにアクセス可能な電子メール・サーバーを構成する必要があります。

ヒント:

リプレイ時に他の本番システムに影響が及ばないようにするために、リプレイは、本番環境のホストにアクセスできない切り離されたプライベート・ネットワーク内で実行することをお薦めします。

接続の再マッピング

ワークロードの取得時に、本番システムへの接続に使用される接続文字列が取得されます。リプレイが正常に完了するには、この接続文字列をリプレイ・システムに再マッピングする必要があります。これによって、リプレイ・クライアントは再マッピングされた接続を使用してリプレイ・システムに接続できます。

Oracle Real Application Clusters(Oracle RAC)データベースの場合、すべての接続文字列をロード・バランシング接続文字列にマップできます。これは、リプレイ・システムのノード数が取得システムとは異なる場合に特に便利です。また、ワークロードを特定のインスタンスに適用する場合は、サービスを使用するか、または再マッピングした接続文字列でインスタンス識別子を明示的に指定することができます。

ユーザーの再マッピング

ワークロードの取得時、本番システムに接続するために使用されるデータベース・ユーザーまたはスキーマのユーザー名が取得されます。取得したユーザー名は、新しいユーザーまたはスキーマに再マッピングするよう選択できます。

リプレイ・オプションの指定

データベースをリストアして接続とユーザーを再マッピングしたら、適切なリプレイ・オプションを設定できます。たとえば:

- 同期方法の指定
- セッションの接続速度の制御
- セッション内のリクエスト速度の制御

同期方法の指定

synchronizationパラメータでは、データベース・リレーに使用する同期方法を指定します。

このパラメータをTIMEに設定すると、リプレイでは取得と同じ実時間が使用されます。すべてのデータベース・セッション・ログイン時間は、取得どおり正確にリプレイされます。同様に、データベース・セッション内のトランザクション間のすべてのタイミングは、取得どおり維持およびリプレイされます。この同期方法により、ほとんどのワークロードで適切なリプレイが実行されます。

このパラメータをSCNに設定すると、取得したワークロード内のCOMMITの順序がリプレイ時に確認され、すべてのリプレイ・アクションがすべての依存COMMITアクションの完了後にのみ実行されます。この同期方法では、ほとんどのワークロードで大幅な遅延が生じる可能性があります。この場合は、synchronizationパラメータとしてTIMEを使用することをお薦めします。

このパラメータをOBJECT_IDに設定すると、関連するすべてのCOMMITアクションが完了するまで、すべてのリプレイ・アクションは実行されません。関連するCOMMITアクションは、次の条件を満たしている必要があります。

- ワークロードの取得内の指定したアクションの前に発行されている。

- 指定したアクションが暗黙的または明示的に参照しているデータベース・オブジェクトが1つ以上修正されている。

このパラメータをOBJECT_IDに設定すると、ワークロードの取得時に同じデータベース・オブジェクトを参照しないCOMMITアクションに関して、ワークロードのリプレイ時の同時実行性が向上します。

セッションの接続速度の制御

connect_time_scaleパラメータによって、ワークロードの取得を開始した時点から各セッションが接続した時点までの経過時間をスケール変更できます。このオプションでは、パーセント値を指定して、リプレイ時のセッションの接続時間を操作できます。デフォルト値は100で、すべてのセッションへの接続が取得時のとおりに試行されます。このパラメータを0(ゼロ)に設定すると、すべてのセッションへの接続が即時試行されます。

セッション内のリクエスト速度の制御

ユーザー思考時間とは、リプレイされたユーザーが単一セッションでコールの発行から次のコールの発行までに待機する経過時間です。リプレイの速度を制御するには、think_time_scaleパラメータを使用してリプレイ時のユーザー思考時間をスケール変更します。

リプレイ時のユーザー・コールの実行が取得時より遅い場合は、think_time_auto_correctパラメータをTRUEに設定して、データベース・リプレイで遅延の回復を試行できます。これによってリプレイ・クライアントでのコール間の思考時間を短縮できるため、リプレイ時の経過時間全体が取得時の経過時間により近くなります。

リプレイ時のユーザー・コールの実行が取得時よりも速い場合は、think_time_auto_correctパラメータをTRUEに設定しても思考時間は変更されません。取得された経過時間に一致させるために、リプレイ・クライアントがコール間の思考時間を長くすることはありません。

ワークロード・リプレイ時のフィルタの使用

デフォルトでは、取得したすべてのデータベース・コールが、ワークロードのリプレイ時にリプレイされます。ワークロード・フィルタを使用して、ワークロード・リプレイ時にワークロードに含めるデータベース・コール、または除外するデータベース・コールを指定できます。

ワークロード・リプレイ・フィルタは、定義後、ワークロード・リプレイで使用できるようにリプレイ・フィルタ・セットに追加されます。包含フィルタおよび除外フィルタという2種類のワークロード・フィルタがあります。包含フィルタでは、リプレイされるデータベース・コールを指定できます。除外フィルタでは、リプレイされないデータベース・コールを指定できます。ワークロード・リプレイでは包含フィルタまたは除外フィルタを使用できますが、両方を使用することはできません。ワークロード・フィルタは、リプレイ・フィルタ・セットの作成時に包含または除外フィルタと判断されます。

リプレイ・クライアントの設定

リプレイ・クライアントはマルチスレッド化されたプログラム($ORACLE_HOME/binディレクトリにあるwrcという名の実行可能ファイル)で、スレッドごとに取得済セッションからワークロードを発行します。リプレイが開始される前に、データベースはリプレイ・クライアントの接続を待機します。この時点で、リプレイ・クライアントを設定して起動する必要があり、このリプレイ・クライアントがリプレイ・システムに接続し、ワークロードに取得された内容に基づいてリクエストを送信します。

リプレイ・クライアントを開始する前に、次の項目を確認します。

- リプレイ・クライアント・ソフトウェアが実行場所のホストにインストールされていること
- リプレイ・クライアントがリプレイ・ディレクトリにアクセスできること

- リプレイ・ディレクトリに事前処理された取得済のワークロードが含まれていること
- リプレイ・ユーザーが正しいユーザーID、パスワードおよび権限を持っていること(リプレイ・ユーザーにはDBAロールが必要であり、SYSユーザーはリプレイ・ユーザーになることができない)
- リプレイ・クライアントは、データベースが実行されているシステムでは起動しません。
- リプレイ・クライアントは、データベース・ファイルが存在しているファイル・システムとは別のファイル・システム上の取得ディレクトリを読み取ります。

  このため、取得ディレクトリをリプレイ・クライアントが実行されるシステムにコピーします。リプレイの完了後、取得ディレクトリは削除できます。

これらの前提条件を満たしたら、wrc実行可能ファイルを使用してリプレイ・クライアントの設定および起動に進むことができます。wrc実行可能ファイルでは、次の構文を使用します。

```
wrc [user/password[@server]] MODE=[value] [keyword=[value]]
```

パラメータuser、password、およびserverは、リプレイ・データベースへの接続に使用するユーザー名、パスワード、および接続文字列を指定します。パラメータmodeは、wrc実行可能ファイルの実行モードを指定します。使用可能な値はreplay(デフォルト)、calibrate、およびlist_hostsです。パラメータkeywordは、実行に使用するオプションを指定し、これは選択したモードにより異なります。使用可能なキーワードおよび対応する値を表示するには、引数なしでwrc実行可能ファイルを実行します。

次の項では、wrc実行可能ファイルの実行時に選択できるモードについて説明します。

- リプレイ・クライアントの較正
- リプレイ・クライアントの起動
- ホスト情報の表示

リプレイ・クライアントの較正

1つのリプレイ・クライアントからデータベースとの複数のセッションを開始できるため、取得されたセッションごとにリプレイ・クライアントを起動する必要はありません。起動する必要があるリプレイ・クライアントの数は、ワークロード・ストリームの数、ホストの数およびホストごとのリプレイ・クライアントの数によって異なります。

リプレイ・クライアントおよび特定のワークロードをリプレイするために必要なホストの数を見積もるには、wrc実行可能ファイルをcalibrateモードで実行します。

較正モードでは、wrc実行可能ファイルで次のキーワードを使用できます。

- replaydir: リプレイする事前処理された取得済のワークロードを含むディレクトリを指定します。指定しない場合は、デフォルトで現在のディレクトリが使用されます。
- process_per_cpu: 各CPUで実行可能なクライアント・プロセスの最大数を指定します。デフォルト値は4です。
- threads_per_process: 1つのクライアント・プロセスで実行可能なスレッドの最大数を指定します。デフォルト値は50です。

次の例は、較正モードでwrc実行可能ファイルを実行する方法を示しています。

```
%> wrc mode=calibrate replaydir=./replay
```

この例では、現在のディレクトリ内のreplayサブディレクトリに格納されている取得済のワークロードのリプレイに必要なリプレイ・クライアントおよびホストの数を見積もるために、wrc実行可能ファイルを実行しています。次の出力例では、21個以上のリプレイ・クライアントを6個のCPUで分割して使用することが推奨されています。

```
Workload Replay Client: Release 12.1.0.0.1 - Production on Fri Sept 30
13:06:33 2011

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Report for Workload in: /oracle/replay/
-----------------------

Recommendation:
Consider using at least 21 clients divided among 6 CPU(s).

Workload Characteristics:
- max concurrency: 1004 sessions
- total number of sessions: 1013

Assumptions:
- 1 client process per 50 concurrent sessions
- 4 client process per CPU
- think time scale = 100
- connect time scale = 100
- synchronization = TRUE
```

リプレイ・クライアントの起動

ワークロードをリプレイするために必要となるリプレイ・クライアントの数を決定したら、wrc実行ファイルがインストールされているホスト上でこれらの実行ファイルをreplayモードで実行し、リプレイ・クライアントを開始する必要があります。各リプレイ・クライアントは、起動されると、データベースとの1つ以上のセッションを開始してワークロードをリプレイします。

リプレイ・モードでは、wrc実行可能ファイルで次のキーワードを使用できます。

- useridおよびpassword: リプレイ・クライアントのリプレイ・ユーザーのユーザーIDおよびパスワードを指定します。指定しない場合は、デフォルトでsystemユーザーが使用されます。
- server: リプレイ・システムへの接続に使用する接続文字列を指定します。指定しない場合は、デフォルトで空の文字列が使用されます。
- replaydir: リプレイする事前処理された取得済のワークロードを含むディレクトリを指定します。指定しない場合は、デフォルトで現在のディレクトリが使用されます。
- workdir: クライアント・ログが書き込まれるディレクトリを指定します。このパラメータは、デバッグを目的としてdebugパラメータとともにのみ使用します。
- debug: デバッグ・データを作成するかどうかを指定します。次の値を指定できます。
    - on

      デバッグ・データが作業ディレクトリのファイルに書き込まれます
    - off

      デバッグ・データは書き込まれません(デフォルト値)

ノート:

wrc 実行可能ファイルをデバッグ・モードで実行する前に、詳細を Oracle サポート・サービスに問い合せてください。

- connection_override: DBA_WORKLOAD_CONNECTION_MAPビューに格納されている接続マッピングを無視するかどうかを指定します。TRUEに設定すると、DBA_WORKLOAD_CONNECTION_MAPビューに格納されている接続の再マッピングは無視され、serverパラメータで指定した接続文字列が使用されます。FALSEに設定すると、すべてのリプレイ・スレッドがDBA_WORKLOAD_CONNECTION_MAPビューに格納されている接続の再マッピングを使用して接続します。これがデフォルトの設定です。
- walletdirは、自動ログイン・ソフトウェア・キーストア・ディレクトリの場所をポイントします。デフォルト値は空の文字列です。walletdirは暗号化されたワークロード取得のリプレイにおいて必須です。暗号化されていない取得の場合はwalletdirは設定できず、デフォルトで空の文字列となります。

ノート:

- 暗号化されたワークロード取得の場合、oracle.rat.database_replay.encryption (大/小文字を区別)識別子を設定する必要があります。パスワードは自動ログイン・ソフトウェア・キーストアに格納されます。
- 暗号化されたワークロード取得のリプレイ時に、別個にクライアント側ソフトウェア・キーストアを設定する必要があります

```
rm -rf keystore_location
mkdir keystore_location
mkstore -wrl keystore_location -create
mkstore -wrl keystore_location -createEntry 'oracle.rat.database_replay.encryption' secret_key
mkstore -wrl keystore_location -createSSO
```

secret_key は、ワークロード取得の暗号化時に作成されたサーバー側ソフトウェア・キーストアで使用された secret
があります。

すべてのリプレイ・クライアントが接続すると、データベースによって取得済のワークロードのストリームがすべての使用可能なリプレイ・クライアントに自動的に配分され、ワークロードのリプレイが開始可能になります。V$WORKLOAD_REPLAY_THREADビューで、リプレイ・クライアントのステータスを監視できます。リプレイが終了すると、すべてのリプレイ・クライアントの接続が自動的に切断されます。

例: 暗号化されていない取得に対するリプレイ・モードでのwrc実行可能ファイルの実行

次の例は、wrc実行可能ファイルをリプレイ・モードで実行する方法を示しています。

```
%> wrc system/password@test mode=replay replaydir=./replay
```

この例では、wrc実行可能ファイルによってリプレイ・クライアントが起動され、現在のディレクトリ内のreplayサブディレクトリに格納されている取得済のワークロードがリプレイされます。

例: 暗号化された取得に対するリプレイ・モードでのwrc実行可能ファイルの実行

次に、暗号化されたワークロード取得についてwrc実行可能ファイルをリプレイ・モードで実行する方法の例を示します。

```
%> wrc system/password@test mode=replay replaydir=./replay walletdir=/tmp/replay_encrypt_cwallet
```

この例では、wrc実行可能ファイルによってリプレイ・クライアントが起動され、現在のディレクトリ内のreplayサブディレクトリに格納されている取得済のワークロードがリプレイされます。walletdirは自動ログイン・ソフトウェアのキーストア・ディレクトリの場所をポイントします。

ホスト情報の表示

wrc実行可能ファイルをlist_hostsモードで実行すると、ワークロードの取得およびワークロードのリプレイに関連したホストを表示することができます。

list_hostsモードでは、wrc実行型ファイルはキーワードreplaydirを使用してリプレイ対象の前処理済ワークロードの取得を含むディレクトリを指定します。指定しない場合は、デフォルトで現在のディレクトリが使用されます。

次の例は、list_hostsモードでwrc実行可能ファイルを実行する方法を示しています。

```
%> wrc mode=list_hosts replaydir=./replay
```

この例では、現在のディレクトリ内のreplayサブディレクトリに格納されているワークロード取得またはリプレイに関連したすべてのホストを示すために、wrc実行可能ファイルを実行しています。次の出力例では、ワークロードの取得およびその後の3回のリプレイに関連したホストが示されています。

```
Workload Replay Client: Release 12.1.0.0.1 - Production on Fri Sept 30
13:44:48 2011

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Hosts found:
Capture:
        prod1
        prod2
Replay 1:
        test1
Replay 2:
        test1
        test2
Replay 3:
        testwin
```

Enterprise Managerを使用したデータベース・ワークロードのリプレイ

この項では、Enterprise Managerを使用したデータベース・ワークロードのリプレイ方法について説明します。

続行する前に、リプレイ・タスクが作成済でリプレイ・タスクからリプレイが作成されている必要があります。これを行う方法の詳細は、"「Enterprise Managerを使用した単一のデータベース・ワークロードの準備」を参照してください。

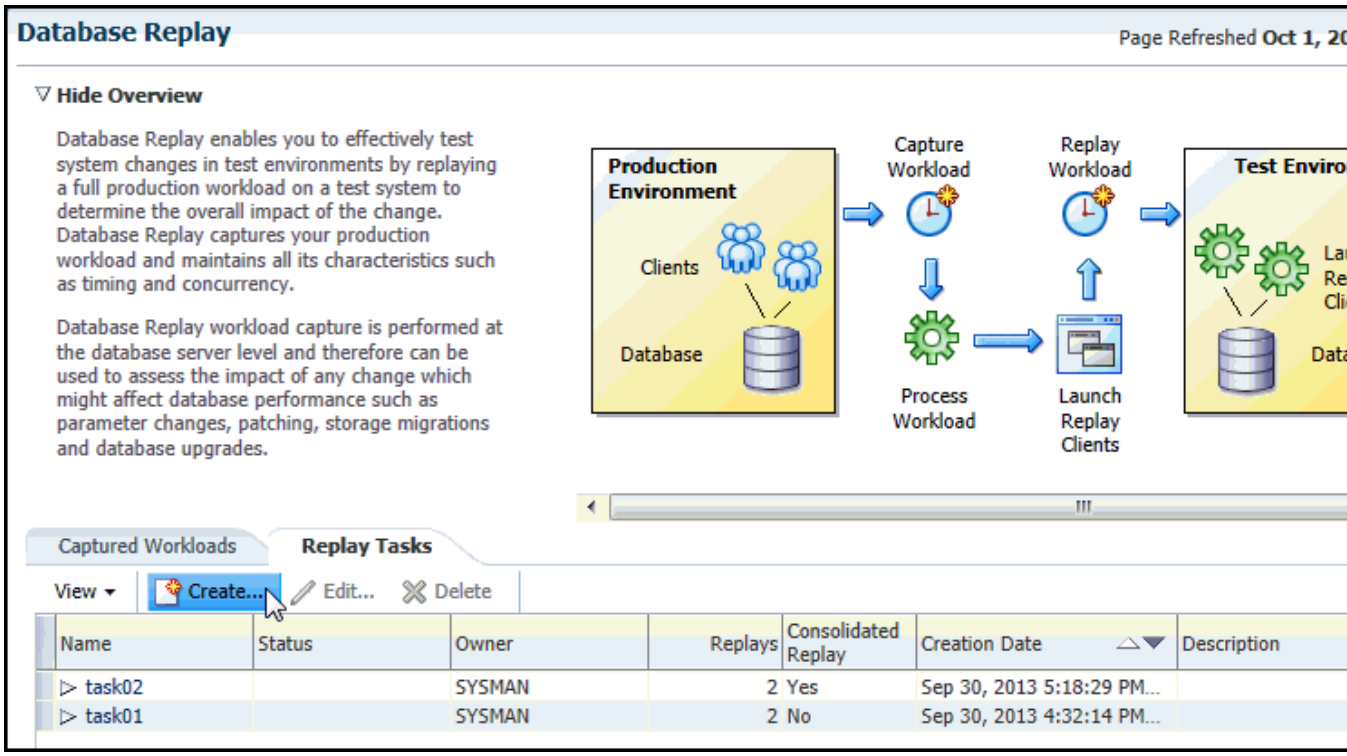
データベース・ワークロードをリプレイするための主要ツールは、Oracle Enterprise Managerです。Oracle Enterprise Managerを使用できない場合、「APIを使用したデータベース・ワークロードのリプレイ」で説明されているように、APIを使用してデータベース・ワークロードをリプレイすることも可能です。

Enterprise Managerを使用してデータベース・ワークロードをリプレイするには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

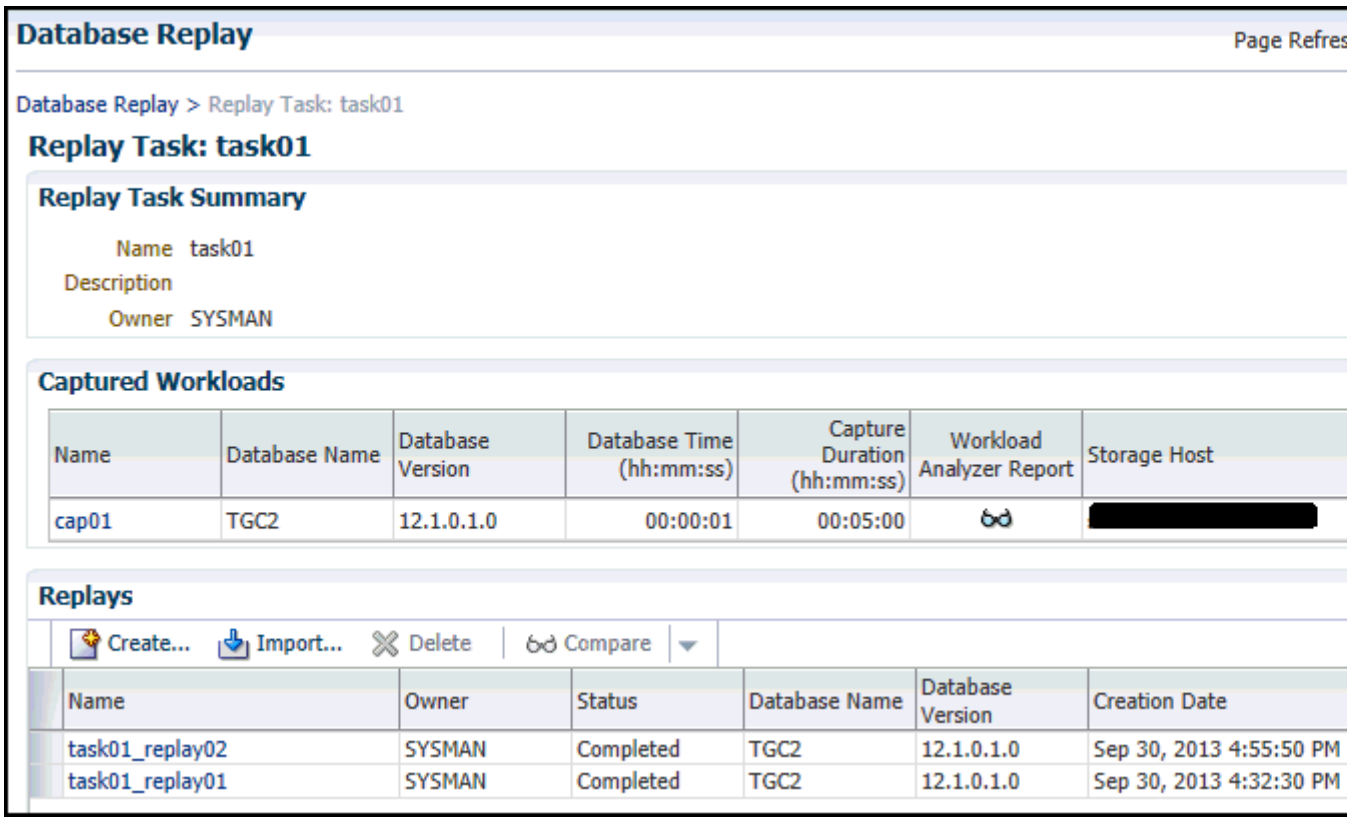
   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。



2. 「リプレイ・タスク」タブを選択し、表の希望のリプレイ・タスクのリンクをクリックします。

   リプレイのリプレイ・タスク・ページが表示されます。



3. 「リプレイ」セクションの「作成」をクリックしてリプレイを作成します。

   「リプレイの作成」ポップアップが表示されます。

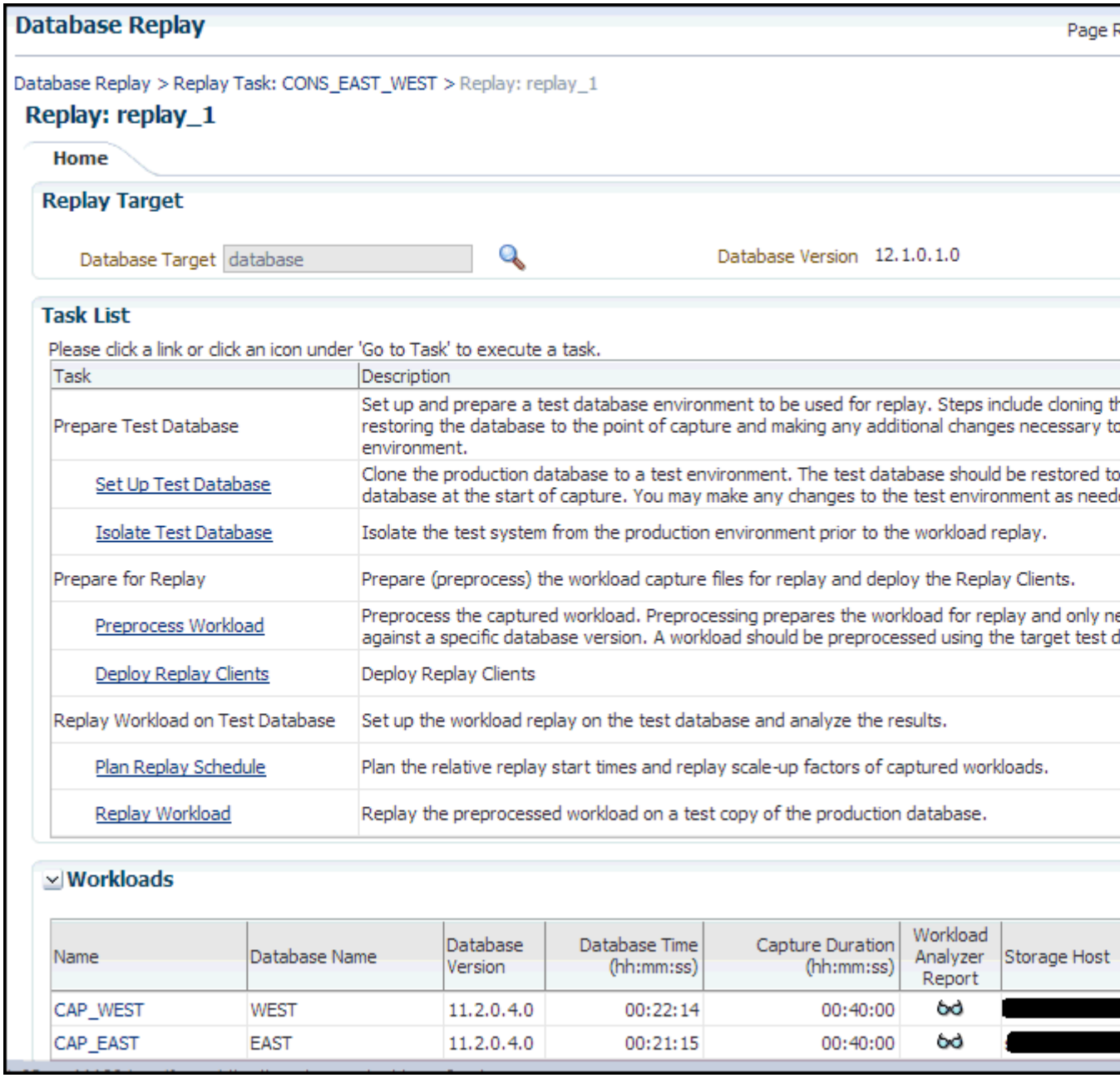
4. 必要な名前と説明(オプション)を入力し、「ターゲット・データベース」アイコンをクリックします。

   「検索と選択: ターゲット」ポップアップが表示されます。

5. 適切なデータベースを選択し、「選択」をクリックします。

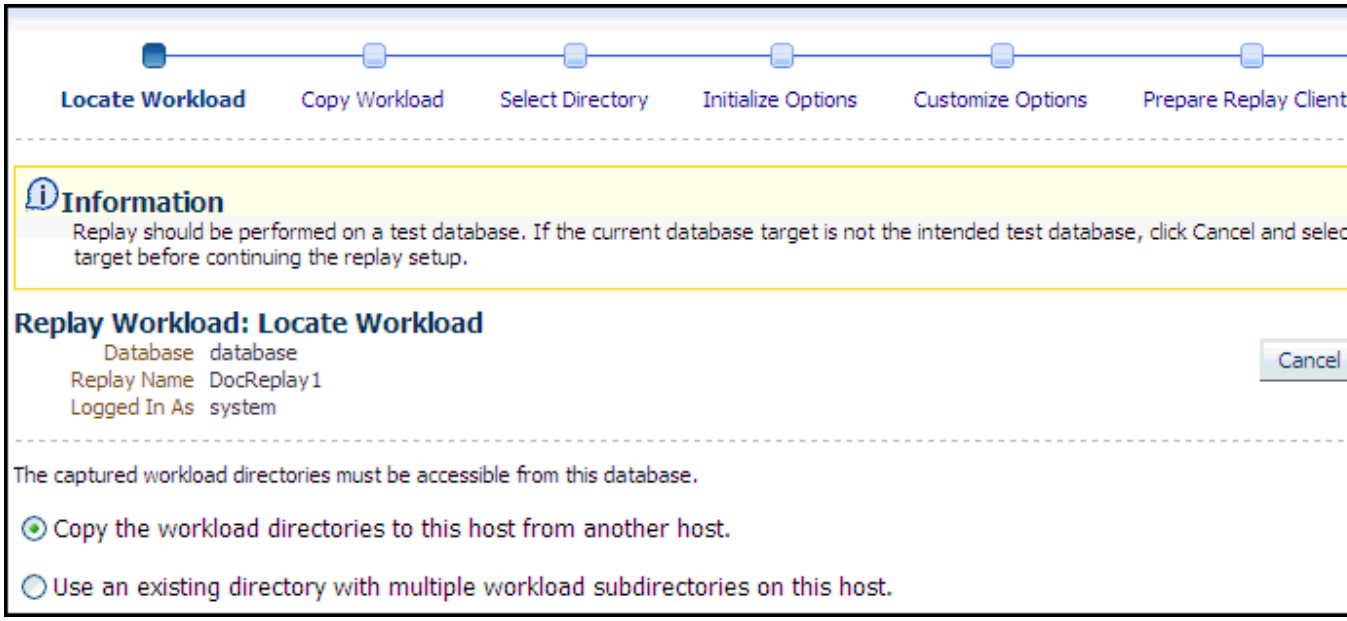
6. 「リプレイの作成」ポップアップの「OK」をクリックします。

   リプレイを実行するリンク付きの「タスク・リスト」が含まれるリプレイのデータベース・リプレイ・ページが表示されます。



7. 「ワークロード・リプレイ」タスクのリンクをクリックします。

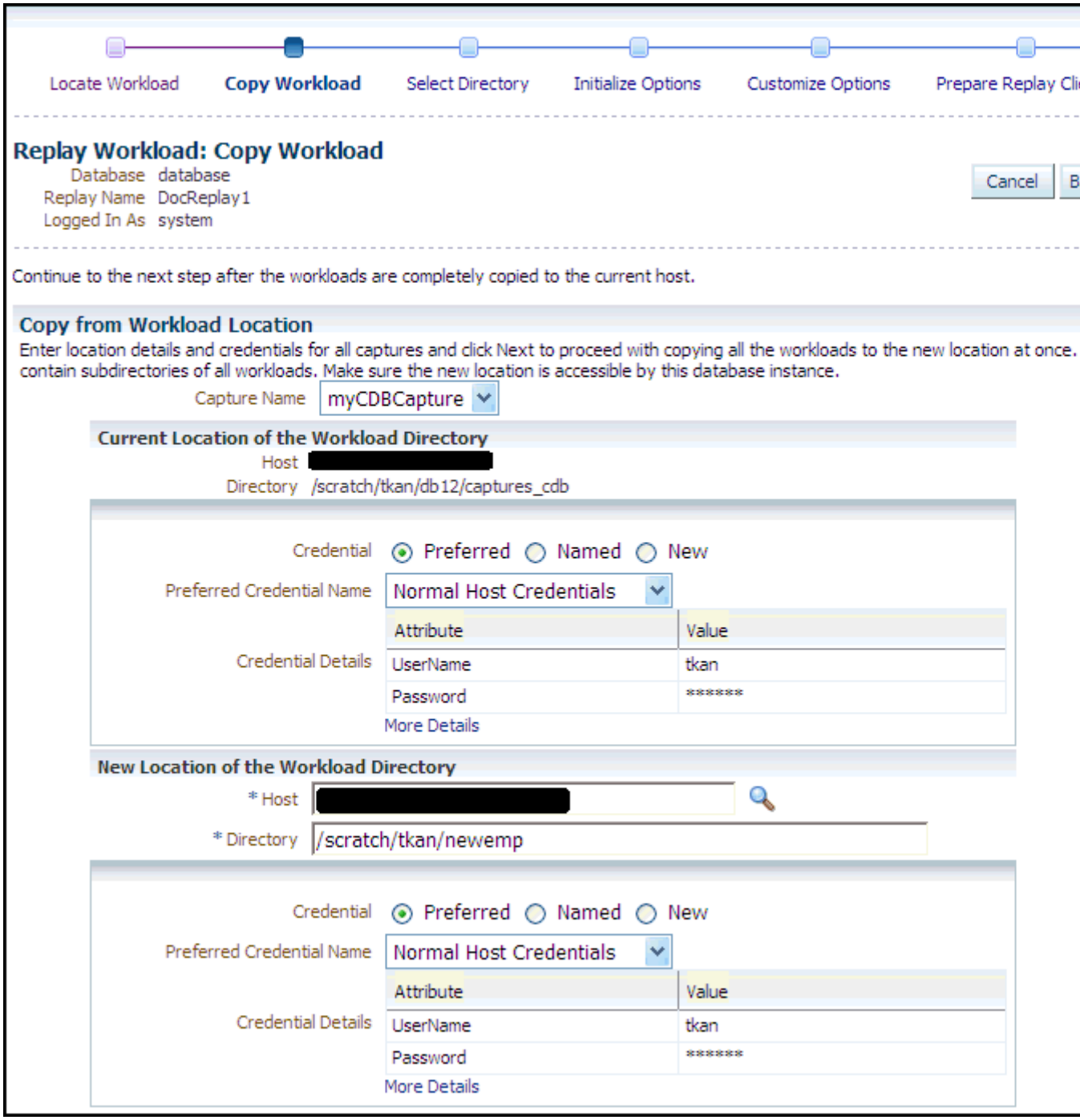
   「ワークロード・リプレイ: ワークロードの検索」ページが表示されます。



8. 必要なワークロードの場所オプションを選択します。

   リプレイ・クライアントがアクセス可能なリプレイ場所に格納されている場所からワークロードをまだコピーしていない場合、ワークロードをコピーするオプションを選択します。それ以外の場合は、リプレイするワークロードを含む既存のリプレイ・ディレクトリを使用するオプションを選択します。

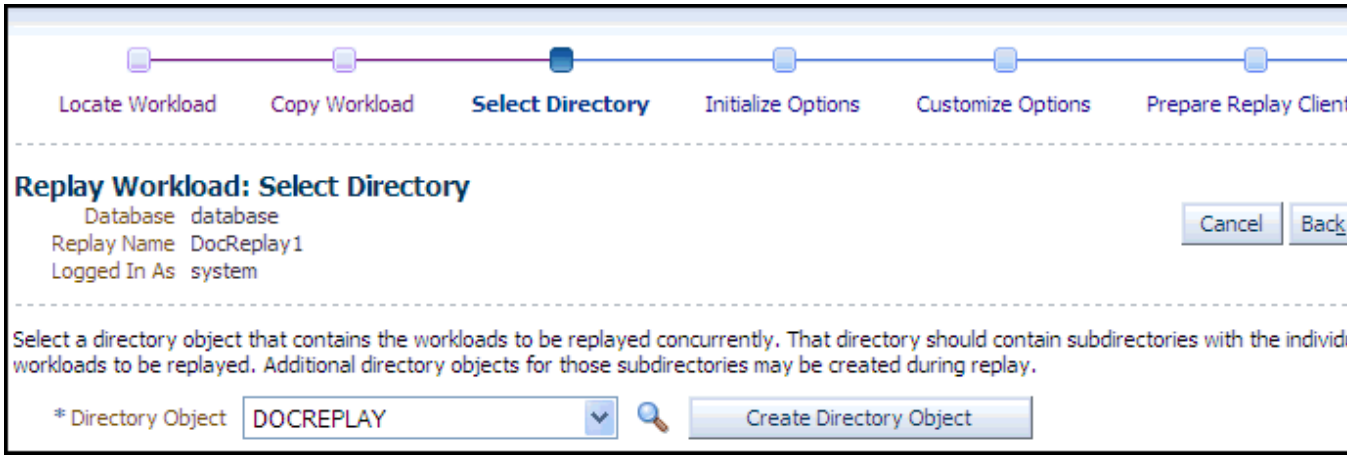
   「次へ」をクリックして、ワークロード・リプレイ: ワークロードのコピー・ページを表示します。



9. 必要な資格証明およびワークロードをコピーする新たなワークロード・ディレクトリの場所を指定して、「次へ」をクリックします。

    - 統合されたリプレイには複数のソース・ワークロードがあるため、複数のソース資格証明をワークロード・ディレクトリの現在の場所に入力しなければならないことがあります。統合リプレイの詳細は、「Enterprise Managerを使用したデータベース統合リプレイの使用」を参照してください。

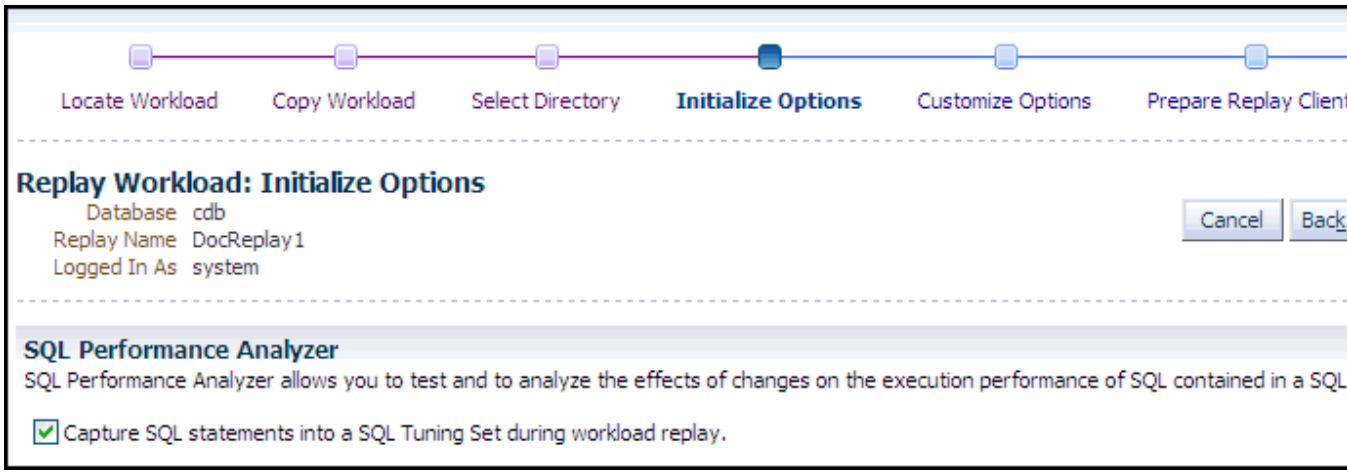
システムは処理中に進捗状況の棒グラフを表示して応答し、コピー操作が終了した後にワークロード・リプレイ: ディレクトリを選択ページが表示されます。



10. ディレクトリ・オブジェクトを指定するかワークロードが含まれる場所をポイントする新しいディレクトリ・オブジェクトを作成します。前のステップでワークロードを新しい場所にコピーするよう選択をした場合、ディレクトリ・オブジェクトが「ワークロード・ディレクトリの新しい場所」で指定した正しい場所を確実にポイントするようにします。

システムは取得サマリーを表示して応答します。「詳細の取得」セクションを展開して、ワークロード・プロファイルとワークロード・フィルタを表示できます。ワークロード取得アナライザ・レポートとデータベース取得レポートも生成できます。統合されたリプレイに取得サマリーは表示されません。

「次へ」をクリックして、ワークロード・リプレイ: 初期化オプション・ページを表示します。



11. 「SQLパフォーマンス・アナライザ」セクションで、デフォルトでは有効になっており推奨されているSQL文の取得オプションを保持または無効化します。リプレイ終了時にSQLチューニング・セットを比較しない場合は、このオプションを無効化できます。

- SQLパフォーマンス・アナライザでは、SQLチューニング・セット内のSQL文のパフォーマンスに関する環境の変更の影響の分析を開始できます。データベースのアップグレード、初期化パラメータの変更、Exadataシミュレーションまたはカスタム試験の結果をテストする、SQLパフォーマンス・アナライザ・タスクを作成および分析できます。このタスクでは、試行前の変更と試行後の変更の結果を比較します。

  データベースのリプレイでは、変更がシステム全体に及ぼす影響が分析されますが、SQLパフォーマンス・アナライザとともにSQLチューニング・セットを使用すれば、変更がSQL文と実行計画にどのように影響するか、SQLを中心とした分析を行うことができます。

  ワークロードのリプレイ中にSQLチューニング・セットを取得することで、SQLパフォーマンス・アナライザを使用して、そのSQLチューニング・セットとワークロードの取得時に取得された別のSQLチューニング・セットを比較できます(SQL文を再実行する必要はありません)。この方法では、データベース・リプレイの実行中に、SQLパ

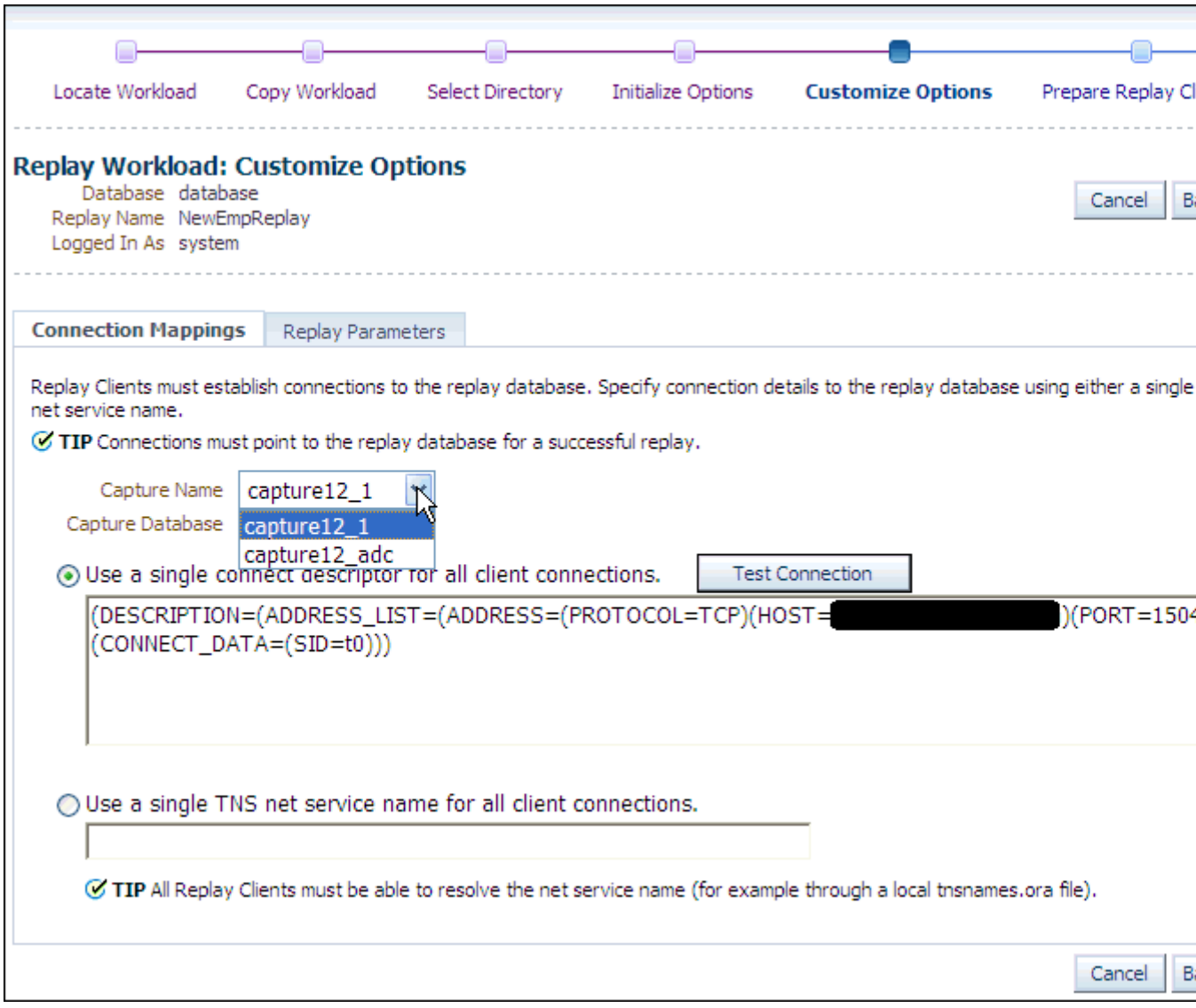
フォーマンス・アナライザ・レポートを取得して、変更の前と後のSQLパフォーマンスを比較することができます。

- 「ソースを指定」セクションでは、リプレイの初期オプションはオプションのカスタマイズ・ページの接続マッピングおよびパラメータを参照しています。接続はワークロードとともに取得されます。

ノート:

このセクションは、統合リプレイまたは Oracle RAC では表示されません。

「次へ」をクリックして、ワークロード・リプレイ: オプションのカスタマイズ・ページを表示します。



12. 取得した接続文字列を、リプレイ・システムを指す接続文字列に再マッピングします。それぞれの取得の接続は再マッピングする必要があることに注意してください。たとえば、前述の図では、capture12_1およびcapture12_adcの両方の接続を再マッピングする必要があります。

(統合リプレイでは、ワークロードごとに接続を再マッピングできます。ワークロードを選択するには、「取得名」ドロップダウンを利用します。)

「接続マッピング」タブをクリックします。取得した接続文字列を再マッピングするには、複数の方法があります。次のいずれかを選択できます。

- すべてのクライアント接続に単一の接続記述子を使用する: このオプションを選択して、使用する接続記述子を入力します。接続記述子が、リプレイ・システムを示す必要があります。

  接続をテストするには、「接続のテスト」をクリックします。接続記述子が有効な場合は、正常に接続されたことを示す情報メッセージが表示されます。

- すべてのクライアント接続に単一のTNSネット・サービス名を使用する: このオプションを選択して、使用するネット・サービス名を入力します。すべてのリプレイ・クライアントは、ローカルのtnsnames.oraファイルを使用してネット・サービス名を解決する必要があります。

- ワークロード内の取得されたクライアント接続記述子ごとに個別の接続記述子またはネット・サービス名を使用する: このオプションを選択して、取得システムの値ごとに、リプレイ・クライアントが使用するリプレイ・システムの値を入力します。「初期化オプション」ステップで「前回のリプレイのオプションを使用」オプションを選択した場合、別の接続記述子の使用に関するオプションが選択され、前回のリプレイ・システムの値が次のテキスト・フィールドに表示されます。

ノート:

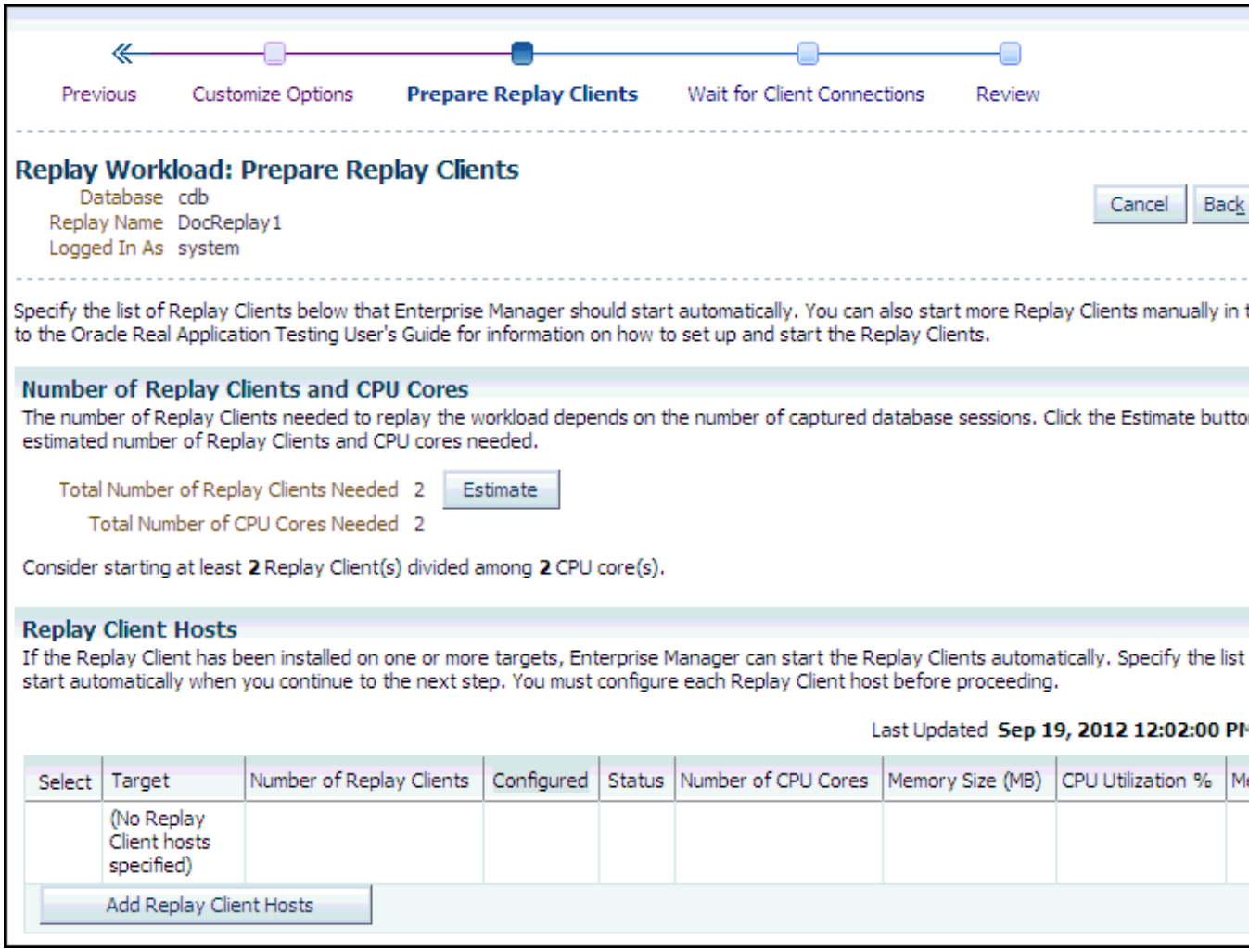
このオプションは、統合リプレイにはありません。

13. リプレイの一部を制御するリプレイ・オプションを、リプレイ・パラメータを使用して指定します。

リプレイの動作を変更するには、「リプレイ・パラメータ」タブをクリックし、リプレイ・パラメータごとに目的の値を入力します。デフォルト値を使用することをお薦めします。リプレイ・パラメータの設定の詳細は、次を参照してください。

リプレイ・パラメータを設定したら、「次へ」をクリックします。

「ワークロード・リプレイ: リプレイ・クライアントの準備」ページが表示されます。



14. リプレイ・クライアントでリプレイの準備ができていることを確認します。

処理を行う前に、リプレイ・クライアントを設定する必要があります。

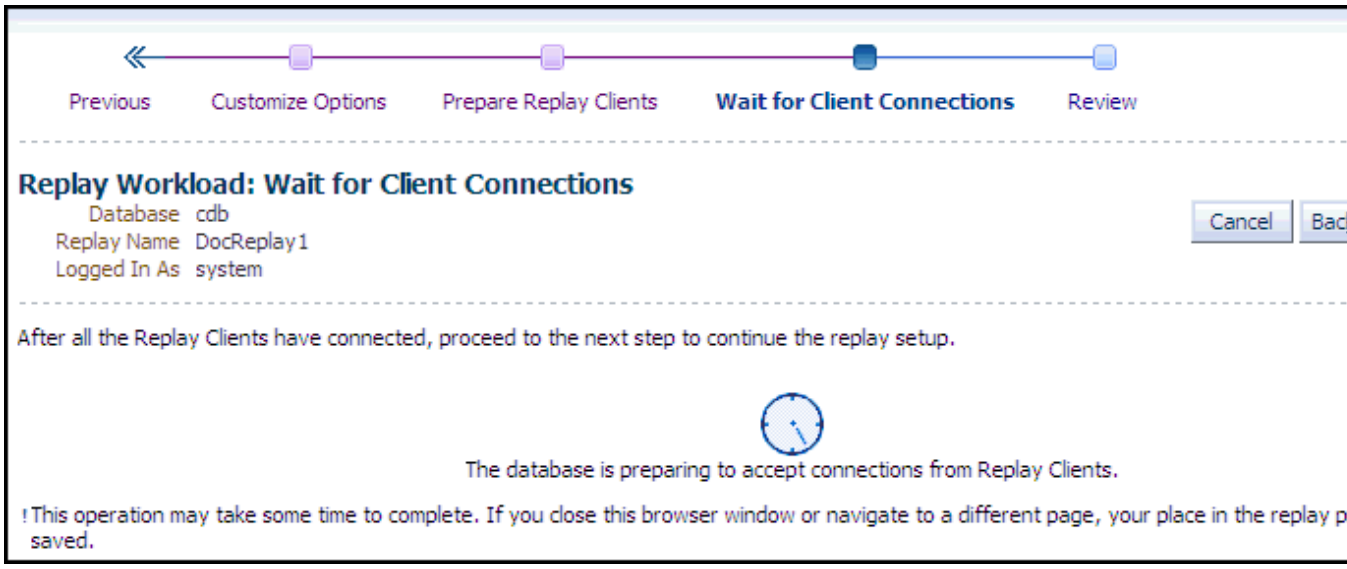
- 「見積り」をクリックして、リプレイに必要なリプレイ・クライアントとCPUの数を特定します。
- 「リプレイ・クライアント・ホストの追加」をクリックして、リプレイ・クライアントのホストを追加します。(リプレイ・クライアントのホストを追加しない場合は、Enterprise Managerを使用しないでコマンドラインからリプレイ・クライアントを開始して続行できます)。

  「検索と選択: リプレイ・クライアント・ホスト」ポップアップが表示されます。

  - ターゲット名を指定して「実行」をクリックするか、「実行」をクリックして使用可能なホストのリスト全体を表示します。
  - ホストを選択して、「選択」をクリックします。
- 「リプレイ・クライアント・ホスト」表の「ターゲット」列にホスト名が表示されたら、見積り結果で推奨されたリプレイ・クライアント数を指定し、ホストにリストされたCPU数が見積り結果の最小推奨数を満たすことを確認します。取得されたワークロードごとに、少なくとも1つのリプレイ・クライアントを起動する必要があります。
- 構成済の列で、「いいえ」リンクをクリックして、表示されたポップアップ内でリプレイ・クライアントを構成します。
- ポップアップに値を入力した後に、「適用」をクリックします。構成済の列の「リプレイ・クライアント・ホスト」表に「はい」と表示されます。

15. 「次へ」をクリックして、リプレイ・クライアントを起動して、ワークロード・リプレイ: クライアント接続の待機ページ

を表示します。



ノート:

プロセスのこのステップに「エンタープライズ」メニューからたどりついた場合は、リプレイ・ジョブおよびリプレイ結果記憶域ホスト用の資格証明も入力する必要があります。

- リプレイ・クライアントを起動すると、リプレイ・クライアント接続が「クライアント接続」表に表示されます。
- リプレイ・クライアントが接続されると、時計の下のテキストが変わります。
- 1つ以上のリプレイ・クライアントが接続されると、「クライアント接続」表が移入されます。

すべてのリプレイ・クライアントが接続されたら、ページ下部にホストの資格証明書を入力しリプレイ・ジョブを開始して、「次へ」をクリックして、ワークロード・リプレイ: 確認ページを表示します。

16. ワークロードのリプレイに関して定義したオプションおよびパラメータを確認します。
    - ワークロードのリプレイ・ジョブを正常に発行するには、「接続されているリプレイ・クライアント」の値を1以上にする必要があります。
    - 「発行」ボタンは、1つ以上のリプレイ・クライアントが接続されている場合のみ有効です。
17. 表示内容がすべて意図したとおりである場合は、「発行」をクリックしてリプレイ・ジョブを発行します。

リプレイが開始された後、「ワークロード・リプレイは開始されました。」というシステム・メッセージが表示された状態で、このリプレイのデータベース・リプレイ・ページの「ホーム」タブが再表示されます。

関連項目:

- 「接続の再マッピング」
- 「リプレイ・オプションの指定」
- 「リプレイ・クライアントの設定」
- 「アクティブなワークロードのリプレイの監視」

Enterprise Managerを使用したリプレイ・スケジュールおよびパラメータの設定

リプレイ・スケジュール機能では、統合リプレイに含める取得済ワークロードのインスタンスをスケール・アップし、リプレイ内のインスタンスの相対再生スケジューリングを制御できます。各インスタンスのアクティブ・セッション・チャートの配置を更新することで、取得済ワークロード・インスタンスの相対スケジューリングが視覚的に表示されます。

この機能は、Cloud Control Databaseプラグイン12.1.0.5以降のリリースで利用できます。

この機能では、次のタスクを実行できます。

- 様々な時間間隔で実行できるようにインスタンスをオフセット

  各ワークロード・インスタンスの相対リプレイ開始時間を調整し、取得ワークロードのスケジュール済インスタンスにアクティブ・セッションの平均ピーク時間を配置できます。このワークロード・ピークの配置により、システムの負荷が最大化される可能性があり、その結果、様々なワークロード条件でテスト・システムがどのように反応するかを実験できます。

- 取得済ワークロードのインスタンスを追加してリプレイ・ワークロードをスケール・アップ

  追加した各インスタンスは、他のインスタンスとは独立してリプレイされます。

  ワークロードの複数のインスタンスを指定してワークロードをスケール・アップすると、デフォルトおよび推奨の構成では、インスタンスのいずれかでDML文がリプレイされます。追加のすべてのインスタンスは文の問合せ(読取り専用)をリプレイするのみです。

  たとえば、ワークロードに従業員データベースへのSQL挿入がある場合、通常は挿入を実行するインスタンスを1つのみにし、それ以外のインスタンスでは、この挿入でデータベースを変更するユーザー・コールを省略します。ただし、「問合せのみリプレイ」チェック・ボックスの選択を解除して、ワークロードのすべての文をリプレイすることで、インスタンスのデフォルト設定をオーバーライドできます。

リプレイ・スケジュールの計画ページにアクセスするには:

1. 「データベース・リプレイ」ホームページで、「リプレイ・タスク」タブをクリックします。
2. 複数のワークロードについて既存のリプレイ・タスクの名前をクリックして、「リプレイ・タスク」ページに移動します。
3. 「作成」をクリックして、新しいリプレイを作成します。
4. 「リプレイの作成」ポップアップで必要な情報を指定して、「OK」をクリックします。

   「リプレイ」表に新しいリプレイが表示されます。
5. 「リプレイ」表の新しいリプレイの名前をクリックします。

   「リプレイ」ページにタスク・リストが表示されます。
6. リプレイ・スケジュールの計画リンクをクリックします。

   リプレイ・スケジュールの計画ページが表示されます。

リプレイをスケール・アップするには:

1. ワークロード・インスタンスの追加ボタンの横のドロップダウンで、インスタンスを追加する取得ワークロードを選択します。
2. ワークロード・インスタンスの追加をクリックしてインスタンスを追加します。

時間間隔をスケジュールするには:

1. リプレイ遅延列で、最初の取得インスタンスをデフォルト値の0のままにするか、インスタンスの実行が開始されるまでの必要な時間(分)を調整します。
2. テスト用にパフォーマンス・スパイクをどのように実行するかを表す値をすべて設定するまで、取得インスタンスごとに前述のステップを繰り返します。

ワークロード・ピークを自動配置するには:

1. 自動配置ボタンをクリックします。

問合せのみのインスタンスを指定するには:

1. 問合せのみにするインスタンスごとに、「問合せのみリプレイ」チェック・ボックスを有効にします。

更新済スケジュールを確認するには:

1. 「データベース・リプレイ」ページの「リプレイ・タスク」タブで、スケジュール済リプレイを含む「リプレイ・タスク」リンクをクリックします。

   「リプレイ・タスク」ページが表示されます。
2. 「リプレイ」表でスケジュール済のタスクをクリックします。
3. 「リプレイ」ページの「確認」タブをクリックします。

Enterprise Managerを使用したワークロードのリプレイの監視

この項では、Enterprise Managerを使用したワークロード・リプレイの監視方法について説明します。ワークロードのリプレイを監視するための主要ツールは、Oracle Enterprise Managerです。Enterprise Managerを使用すると、次の操作を実行できます。

- アクティブなワークロードのリプレイの監視または停止
- 完了したワークロードのリプレイの表示

Oracle Enterprise Managerが使用できない場合は、「APIを使用したワークロードのリプレイの監視」で説明されているように、APIおよびビューを使用してワークロードのリプレイを監視できます。

この項では、次の項目について説明します。

アクティブなワークロードのリプレイの監視

この項では、Enterprise Managerを使用してアクティブなワークロードのリプレイを監視する方法について説明します。

アクティブなワークロードのリプレイを監視するには:

1. データベース・リプレイ・ページで「リプレイ・タスク」タブをクリックします。
2. リプレイの進捗状況を監視するリプレイが含まれるリプレイ・タスクの名前をクリックします。
3. リプレイ・タスク・ページの「リプレイ」セクションから、リプレイの作成ウィザードで処理用に発行したリプレイの名前をクリックします。(このリプレイの「ステータス」列には「進行中」と表示されています。)

   データベース・リプレイ・ページの「ホーム」タブが表示され、「リプレイ・サマリー」に実行のステータスが表示されます。

- リプレイの進行中には、「リプレイの進捗状況」チャートのリプレイ線が動的に更新されます。「リフレッシュ」ボタンをクリックすると、チャートを更新できます。
- 「リプレイの進捗状況」チャートのユーザー・コール線は、同じ時期の取得と比較して、データベースがどの程度のワークロードを処理したかを示しています。
- リプレイが完了するまで、「リプレイ相違サマリー」のデータは使用できません。

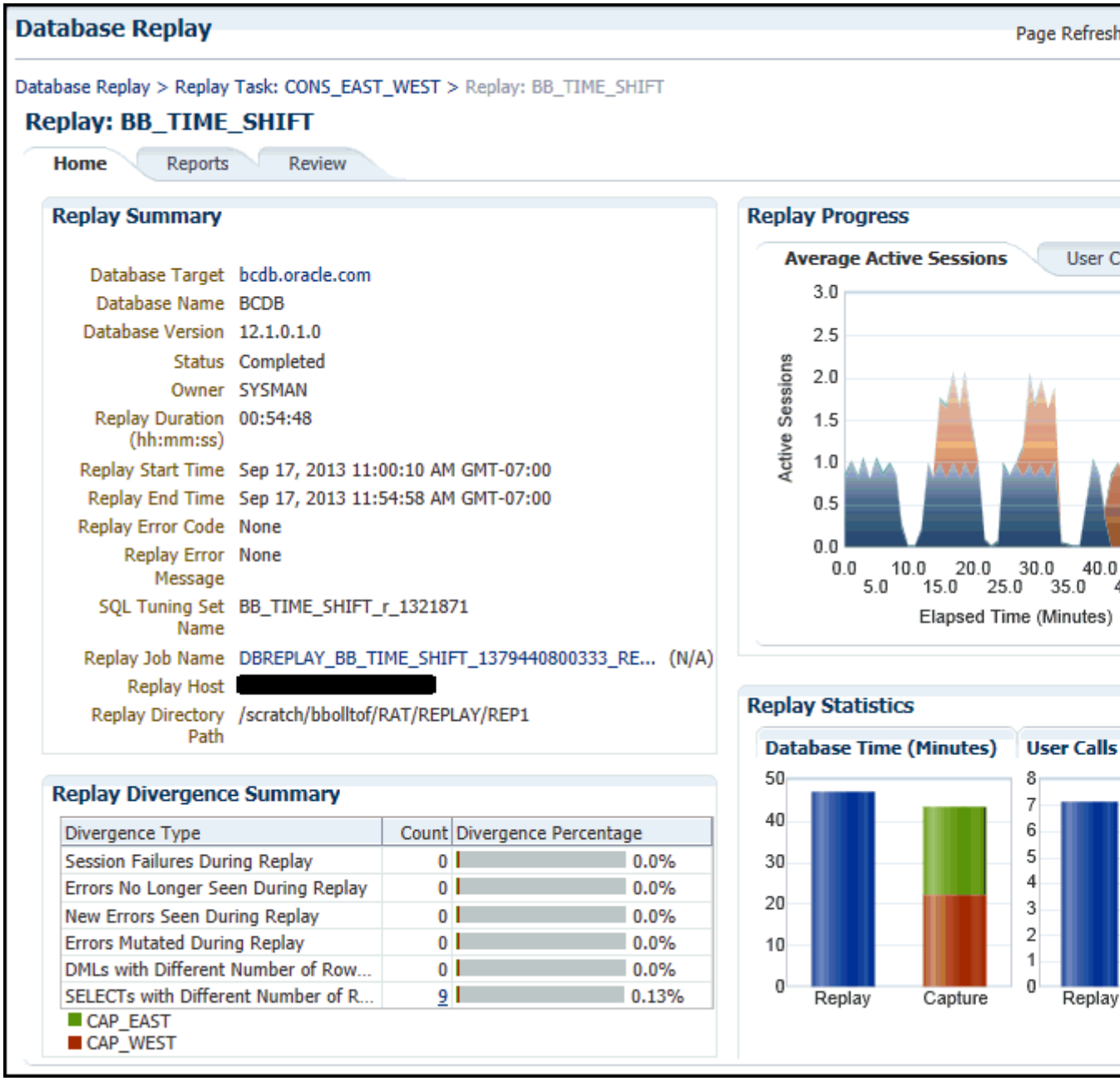
完了したワークロードのリプレイの表示

この項では、完了したワークロードのリプレイをEnterprise Managerを使用して表示する方法について説明します。

完了したワークロードのリプレイを表示するには:

1. データベース・リプレイ・ページで「リプレイ・タスク」タブをクリックします。
2. 表示する完了済リプレイが含まれるリプレイ・タスクの名前をクリックします。
3. リプレイ・タスク・ページの「リプレイ」セクションから、リプレイの作成ウィザードで処理用に発行したリプレイの名前をクリックします。(このリプレイの「ステータス」列には「完了」と表示されています。)

   データベース・リプレイ・ページの「ホーム」タブが表示され、「リプレイ・サマリー」に完了のステータスが表示されます。



- 「ユーザー・コール」チャートのリプレイ線は、開始から完了までのリプレイ全体を通して、リプレイの進捗状況をグラフィカルに示します。

  グラフに、ワークロードの取得中にユーザー・コールに関して経過した時間と比較して、同じワークロードのリプレイにかかった時間が表示されます。「リプレイ」の線が「取得」の線の上側または左側に表示される場合、リプレイ・システムは取得システムよりも高速にワークロードを処理しています。

- 「リプレイ相違サマリー」には、リプレイ・システムと取得システム間のエラーおよびデータの矛盾点が、リプレイ時に違いが出たデータベース・コールとして表示されます。リプレイの質の測定値として違いが出た合計コール数の割合を使用できます。

  違いが出たコールの詳細を表示するには、「件数」列で違いが出たコールのタイプに対応するリンクをクリックし、リプレイ時に違いのあったコール・ページにアクセスします。「リプレイ時に違いのあったコール」ページには、取得されたワークロードと違いがあるリプレイ済コールの最も関連性が高いセットが、共通属性値および指定したフィルタ条件に基づいてグループ化されて表示されます。違いがある特定のコールに関する詳細(コールの属性、SQLテキスト、バインド変数など)を表示するには、「SQL ID」列の対応するリンクをクリックすると、違いの出た

文をリプレイするページが表示されます。

4. データベース・リプレイ・ページに戻るには、「データベース・リプレイ」ブレッドクラムをクリックします。

関連項目:

ワークロード・リプレイ・レポートへのアクセスの詳細は、「取得およびリプレイ済ワークロードの分析」を参照してください

Enterprise Manager外部のリプレイのインポート

Enterprise Manager外部のワークロードのインポートと同様に、リプレイをEnterprise Managerにインポートして管理できます。リプレイをインポートするには、1つ以上のワークロードおよびリプレイを含めることができるリプレイ・タスクからインポートします。リプレイ・タスクは、データベース・リプレイの階層の最上部にあり、これらの他の従属コンポーネントのコンテナとして機能します。

インポート対象のリプレイは、テスト・データベースで実行したり、リプレイを完了して、ファイル・システムにリプレイ・ディレクトリを格納できます。

この機能は、Cloud Control Databaseプラグイン12.1.0.5以降のリリースで利用できます。

Enterprise Manager外部のリプレイをインポートするには:

1. 「データベース・リプレイ」ページの「リプレイ・タスク」タブをクリックし、目的のリプレイ・タスクを選択します。

   「リプレイ・タスク」ページが表示されます。

2. 「リプレイ」セクションの「インポート」をクリックします。

   「リプレイのインポート: ソース」ページが表示されます。

3. 次の3つの選択肢のいずれかを選択して、リプレイをインポートし、「次へ」をクリックします。

   - 完了した1つ以上のリプレイをファイル・システムのディレクトリからインポート

     このオプションは、通常APIを使用して作成されたリプレイに適用され、後続の処理用にEnterprise Managerにインポートします。この場合、Enterprise Managerは必ずしもリプレイ・データベースを管理していない可能性があります。

   - 完了した1つ以上のリプレイをデータベース・ターゲットからインポート

     この場合、Enterprise Managerはリプレイ・データベースをすでに管理している可能性があります。リプレイは、このデータベースで実行された可能性があるか、前述のオプションの場合と同様にロードされた可能性があります。

   - データベース・ターゲットで実行中のこのリプレイ・タスクのリプレイにアタッチ

     このオプションは前述のオプションと似ていますが、すでに完了しているリプレイではなく、実行中のリプレイである点が異なります。

   「リプレイのインポート: データベース」ページが表示されます。

4. 「データベース・ターゲット」フィールドの横の検索アイコンをクリックして、表示されるポップアップからデータベースを選択します。

ノート:

リプレイを読み取るデータベースのターゲット・バージョンは、リプレイ・タスクで使用されるバージョン以上である必要があります。たとえば、Oracle Database 12x で使用されるリプレイ・タスクの場合、リプレイの読取りで選択するデータベースはバージョン 12x 以上である必要があります。

- データベースおよびホスト資格証明が要求されます。
- 前述のステップで、完了した1つ以上のリプレイをファイル・システムのディレクトリからインポートを選択した場合は、ワークロードの場所も要求されます。
- リプレイ・タスクでは、リプレイ・タスクに含まれるワークロードの数に基づいて、統合リプレイであるかどうかを判別できます。統合リプレイの場合は、ワークロードの場所に統合リプレイのディレクトリを入力するよう求められます。

5. 前述のステップで必要な入力を行い、「次へ」をクリックします。

   「リプレイのインポート: リプレイ」ページが表示されます。

   - ステップ3で、「1つ以上の完了したリプレイをファイル・システムのディレクトリからインポートします。」を選択した場合は、このページに「リプレイのロード」ボタンが表示されます。
   - ステップ3で、「1つ以上の完了したリプレイをデータベース・ターゲットからインポートします。」または「データベース・ターゲットで実行中のこのリプレイ・タスクのリプレイにアタッチします。」を選択した場合は、このページに「リプレイの検出」ボタンが表示されます。

6. ステップ3の選択に従って表示されるボタンに応じて、「リプレイのロード」または「リプレイの検出」のいずれかをクリックします。

   リプレイが1つ以上見つかった場合は、検出されたリプレイ表に表示されます。

7. 「次へ」をクリックして、リプレイをロードするか、リプレイを1つ以上選択し、「次へ」をクリックしてリプレイのインポートを継続します。

   「リプレイのインポート: 確認」ページが表示されます。

8. すべてが目的どおりに表示されたら、「送信」をクリックします。

   「データベース・リプレイ」ページには、ジョブが正常に送信されたことを示すメッセージが表示されます。表で、インポートしたリプレイの「ステータス」列に「進行中」と表示されます。

ヒント:

「確認」ステップで送信したリプレイ名をクリックして、ジョブの進行状況を確認できます。「リプレイ・サマリー」ページが表示され、データベース・リプレイ・インポート・ジョブ・リンクをクリックして、ジョブ実行ステップの進行状況を確認できます。

APIを使用したデータベース・ワークロードのリプレイ

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用してデータベース・ワークロードをリプレイする方法について説明します。また、「Enterprise Managerを使用したデータベース・ワークロードのリプレイ」で説明されているように、Oracle Enterprise Managerを使用してデータベース・ワークロードをリプレイできます。

DBMS_WORKLOAD_REPLAYパッケージを使用したデータベース・ワークロードのリプレイは、次に示す複数のステップで構成されているプロセスです。

- リプレイ・データの初期化
- 接続の再マッピング
- ユーザーの再マッピング
- ワークロードのリプレイ・オプションの設定
- ワークロード・リプレイ・フィルタおよびリプレイ・フィルタ・セットの定義
- リプレイのタイムアウト・アクションの設定
- ワークロードのリプレイの開始
- ワークロード・リプレイの一時停止
- ワークロード・リプレイの再開
- ワークロード・リプレイの取消し
- ワークロード・リプレイに関する情報の取得
- ワークロード・リプレイの相違データのロード
- ワークロード・リプレイに関する情報の削除
- ワークロードのリプレイのAWRデータのエクスポート
- ワークロードのリプレイのAWRデータのインポート

関連項目:

- DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

リプレイ・データの初期化

取得したワークロードを事前処理し、テスト・システムを適切に準備したら、リプレイ・データを初期化できます。リプレイ・データの初期化では、必要なメタデータが、ワークロードのリプレイで必要な表にロードされます。たとえば、取得した接続文字列が、リプレイ用に再マッピング可能な表にロードされます。

リプレイ・データを初期化するには:

- INITIALIZE_REPLAYプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.INITIALIZE_REPLAY (replay_name => 'dec06_102',
                            replay_dir => 'dec06',
                            plsql_mode => 'top_level');
END;
/
```

この例では、INITIALIZE_REPLAYプロシージャによって事前処理済のワークロード・データがdec06ディレクトリからデータベースにロードされます。

この例のINITIALIZE_REPLAYプロシージャでは、次のパラメータを使用します。

- replay_name: 以前のリプレイの設定およびフィルタを取得するために他のAPIで使用できるリプレイ名を指定する必須パラメータ。
- replay_dir: リプレイする取得済のワークロードを含むディレクトリを指定する必須パラメータ。
- オプションのplsql_modeパラメータで、PL/SQLのリプレイ・モードを指定します。

  plsql_modeパラメータには次の2つの値を設定できます。

  - top_level: 最上位レベルのPL/SQLコールのみ。これがデフォルト値です。
  - extended: PL/SQL内で実行されたSQL、またはPL/SQL内に記録されているSQLがない場合は最上位レベルのPL/SQL。PL/SQL以外のコールは通常の方法でリプレイされます。

ノート:

暗号化されたワークロード取得で INITIALIZE_REPLAY を実行するには、oracle.rat.database_replay.encryption 識別子(大/小文字を区別)を使用してパスワードを設定する必要があります。パスワードはソフトウェア・キーストアに格納されます。ワークロード取得が暗号化されているかどうかは、DBA_WORKLOAD_CAPTURES ビューから確認できます。

関連項目:

- ワークロードの取得の事前処理については、「APIを使用したデータベース・ワークロードの事前処理」を参照してください
- テスト・システムの準備の詳細は、「データベース・ワークロードのリプレイのステップ」を参照してください

接続の再マッピング

リプレイ・データを初期化したら、ユーザー・セッションが適切なデータベースに接続し、リプレイ時に取得される場合と同様に外部との対話を実行できるように、取得済のワークロードで使用されている接続文字列を再マッピングする必要があります。接続マッピングを表示するには、DBA_WORKLOAD_CONNECTION_MAPビューを使用します。

接続を再マッピングするには:

- REMAP_CONNECTIONプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (connection_id => 101,
                            replay_connection => 'dlsun244:3434/bjava21');
END;
/
```

この例では、接続ID 101に対応する接続で、replay_connectionパラメータで定義された新しい接続文字列が使用されます。

この例のREMAP_CONNECTIONプロシージャでは、次のパラメータを使用します。

- connection_id: リプレイ・データの初期化時に生成され、取得したワークロードの接続に対応している必須

パラメータ。

- replay_connection: ワークロードのリプレイ時に使用される新しい接続文字列を指定する必須パラメータ。

関連項目:

「接続の再マッピング」

ユーザーの再マッピング

接続文字列の再マッピングに加え、ワークロード取得で取得したユーザーのかわりに、新しいスキーマやユーザーを使用することも可能です。取得済のユーザーを参照するには、DBA_WORKLOAD_USER_MAPビューを使用します。

ユーザーを再マッピングするには:

- SET_USER_MAPPINGプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (capture_user => 'PROD',
                            replay_user => 'TEST');
END;
/
```

この例では、取得時に使用されたPRODユーザーは、リプレイ時にTESTユーザーに再マッピングされます。

この例のSET_USER_MAPPINGプロシージャでは、次のパラメータを使用します。

- 必須のcapture_userパラメータには、ワークロードの取得時に取得したユーザー名を指定します。
- 必須のreplay_userパラメータには、取得したユーザーがリプレイ時に再マッピングされるユーザー名を指定します。パラメータがNULLに設定されている場合、マッピングは無効です。

関連項目:

「ユーザーの再マッピング」

ワークロードのリプレイ・オプションの設定

リプレイ・データを初期化し、接続とユーザーを再マッピングしたら、ワークロードのリプレイ用にデータベースを準備する必要があります。

ワークロードのリプレイをリプレイ・システムで準備するには:

- PREPARE_REPLAYプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY (synchronization => 'OBJECT_ID',
                            capture_sts => TRUE,
                            sts_cap_interval => 300);
END;
/
```

この例では、PREPARE_REPLAYプロシージャによって、事前に初期化されたリプレイが準備されます。ワークロードのリプレ

イと並行して、SQLチューニング・セットも取得されます。

PREPARE_REPLAYプロシージャでは、次のパラメータを使用します。

- synchronization必須パラメータでは、ワークロードのリプレイ時に使用される同期のタイプを指定します。

  このパラメータをTIMEに設定すると、リプレイでは取得と同じ実時間が使用されます。すべてのデータベース・セッション・ログイン時間は、取得どおり正確にリプレイされます。同様に、データベース・セッション内のトランザクション間のすべてのタイミングは、取得どおり維持およびリプレイされます。この同期方法により、ほとんどのワークロードで適切なリプレイが実行されます。

  このパラメータをSCN (デフォルト値)に設定すると、取得したワークロード内のCOMMITの順序がリプレイ時に確認され、すべてのリプレイ・アクションがすべての依存COMMITアクションの完了後にのみ実行されます。この同期方法では、ほとんどのワークロードで大幅な遅延が生じる可能性があります。この場合は、synchronizationパラメータとしてTIMEを使用することをお薦めします。

  このパラメータをOBJECT_IDに設定すると、関連するすべてのCOMMITアクションが完了するまで、すべてのリプレイ・アクションは実行されません。関連するCOMMITアクションは、次の条件を満たしている必要があります。

  - ワークロードの取得内の指定したアクションの前に発行されている。
  - 指定したアクションが暗黙的または明示的に参照しているデータベース・オブジェクトが1つ以上修正されている。

  このパラメータをOBJECT_IDに設定すると、ワークロードの取得時に同じデータベース・オブジェクトを参照しないCOMMITアクションに関して、ワークロードのリプレイ時の同時実行性が向上します。

- connect_time_scale: 指定した値で、ワークロードの取得を開始した時点からセッションが接続する時点までの経過時間をスケール変更するパラメータで、%値として解釈されます。このパラメータは、リプレイ中に同時ユーザー数を増加または削減する場合に使用します。デフォルト値は100です。

- think_time_scale: 同一セッションからの連続する2つのユーザー・コール間の経過時間をスケール変更するパラメータで、%値として解釈されます。このパラメータを0(ゼロ)に設定すると、リプレイ時にユーザー・コールは可能なかぎり高速でデータベースに送信されます。デフォルト値は100です。

- think_time_auto_correct: リプレイ時にユーザー・コールの完了にかかる時間が取得時より長い場合に、コール間の思考時間を(think_time_scaleパラメータに基づいて)変更するパラメータ。このパラメータは、TRUEまたはFALSEに設定できます。このパラメータをTRUEに設定すると、ワークロードのリプレイがワークロードの取得より長くかかっている場合に、思考時間が短縮されます。デフォルト値はTRUEです。

- scale_up_multiplier: リプレイ時にワークロードをn倍に増加させる値を定義するパラメータ。取得された各セッションが、このパラメータで指定された値を掛けた数だけ、同時にリプレイされます。ただし、同一リプレイ・セッションの各セット内で1つのセッションのみが問合せと更新の両方を実行します。残りのセッションは問合せのみを実行します。

- capture_sts: ワークロードのリプレイと並行してSQLチューニング・セットを取得するかどうかを指定するパラメータ。このパラメータをTRUEに設定した場合、ワークロードのリプレイ中に1つのSQLチューニング・セットを取得することで、SQL文を再実行しなくても、SQLパフォーマンス・アナライザを使用してそのSQLチューニング・セットを別のSQLチューニング・セットと比較できます。この方法では、データベース・リプレイの実行中に、SQLパフォーマンス・アナライザ・レポートを取得して、変更の前と後のSQLパフォーマンスを比較することができます。また、「ワークロードのリプレイのAWRデータのエクスポート」で説明されているように、EXPORT_AWRプロシージャを使用して、結果のSQLチューニング・セットをそのAWRデータとともにエクスポートできます。

この機能は、Oracle RACではサポートされていません。DBMS_WORKLOAD_REPLAYを使用して定義したワークロード・リプレイ・フィルタは、SQLチューニング・セットの取得には適用されません。このパラメータのデフォルト値はFALSEです。

- sts_cap_interval: カーソル・キャッシュからのSQLチューニング・セット取得の継続時間(秒)を指定するパラメータ。デフォルト値は300です。このパラメータの値をデフォルト値未満に設定すると、一部のワークロードで追加オーバーヘッドが生じる可能性があるため、そのような設定は推奨されません。

これらのパラメータの設定の詳細は、「リプレイ・オプションの指定」を参照してください。

関連項目:

「データベース・ワークロードのリプレイのステップ」

ワークロード・リプレイ・フィルタおよびリプレイ・フィルタ・セットの定義

この項では、ワークロード・リプレイ・フィルタの追加および削除方法と、リプレイ・フィルタ・セットの作成および使用方法について説明します。

この項では、次の項目について説明します。

- ワークロード・リプレイ・フィルタの追加
- ワークロード・リプレイ・フィルタの削除
- リプレイ・フィルタ・セットの作成
- リプレイ・フィルタ・セットの使用

関連項目:

「ワークロード・リプレイ時のフィルタの使用」

ワークロード・リプレイ・フィルタの追加

この項では、リプレイ・フィルタ・セットで使用する新しいフィルタを追加する方法を説明します。

新しいフィルタを追加するには:

- ADD_FILTERプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.ADD_FILTER (
                            fname => 'user_ichan',
                            fattribute => 'USER',
                            fvalue => 'ICHAN');
END;
/
```

この例では、ADD_FILTERプロシージャは、user_ichanというフィルタを追加します。このフィルタは、ユーザー名ICHANに属するすべてのセッションを除外するために使用できます。

この例のADD_FILTERプロシージャでは、次のパラメータを使用します。

- fname: 追加するフィルタの名前を指定する必須パラメータ。
- fattribute: フィルタを適用する属性を指定する必須パラメータ。有効値は、PROGRAM、MODULE、ACTION、SERVICE、USER、およびCONNECTION_STRINGです。CONNECTION_STRING属性として、リプレイ時に使用される有効な取得した接続文字列を指定する必要があります。
- fvalue: フィルタを適用する属性に対応する値を指定する必須パラメータ。一部の属性(モジュールやアクションなど)では、%などのワイルドカードを使用できます。

すべてのワークロード・リプレイ・フィルタを追加した後に、ワークロードのリプレイ時に使用できるリプレイ・フィルタ・セットを作成できます。

ワークロード・リプレイ・フィルタの削除

この項では、ワークロード・リプレイ・フィルタを削除する方法を説明します。

ワークロード・リプレイ・フィルタを削除するには:

- DELETE_FILTERプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.DELETE_FILTER (fname => 'user_ichan');
END;
/
```

この例では、DELETE_FILTERプロシージャはuser_ichanというフィルタを削除しています。

DELETE_FILTERプロシージャでは、削除するフィルタの名前を指定する必須パラメータfnameを使用します。

リプレイ・フィルタ・セットの作成

ワークロード・リプレイ・フィルタを追加した後に、ワークロード・リプレイで使用するリプレイ・フィルタのセットを作成することができます。リプレイ・フィルタ・セットを作成する際、前回のリプレイ・フィルタ・セットの作成以降に追加されたすべてのワークロード・リプレイ・フィルタが使用されます。

リプレイ・フィルタ・セットを作成するには:

- CREATE_FILTER_SETプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.CREATE_FILTER_SET (
                            replay_dir => 'apr09',
                            filter_set => 'replayfilters',
                            default_action => 'INCLUDE');
END;
/
```

この例のCREATE_FILTER_SETプロシージャは、replayfiltersという名前のリプレイ・フィルタ・セットを作成します。このリプレイ・フィルタ・セットは、リプレイ・フィルタで定義されたワークロードの部分は除いて、apr09ディレクトリに保存されているリプレイに対して取得されたすべてのコールをリプレイします。

この例のCREATE_FILTER_SETプロシージャでは、次のパラメータを使用します。

- replay_dir:フィルタ対象のリプレイが格納されているディレクトリを指定するパラメータ。

- filter_set: 作成するフィルタ・セットの名前を指定するパラメータ。
- default_action: 取得された各データベース・コールを再生するかどうか、およびワークロード・リプレイ・フィルタが包含フィルタと除外フィルタのどちらとみなされるかを決定するパラメータ。

  このパラメータをINCLUDEに設定すると、リプレイ・フィルタで定義されたワークロードの部分を除いて、取得されたすべてのデータベース・コールがリプレイされます。この場合、すべてのリプレイ・フィルタが除外フィルタとして処理され、これらのフィルタはリプレイされないワークロードの部分を定義することになります。これはデフォルトの動作です。

  このパラメータをEXCLUDEに設定すると、リプレイ・フィルタで定義されたワークロードの部分を除いて、取得されたすべてのデータベース・コールがリプレイされません。この場合、すべてのリプレイ・フィルタが包含フィルタとして処理され、これらのフィルタはリプレイされるワークロードの部分を定義することになります。

リプレイ・フィルタ・セットの使用

リプレイ・フィルタ・セットを作成してリプレイが初期化されると、リプレイ・フィルタ・セットを使用して、replay_dirディレクトリ内のリプレイをフィルタすることができます。

リプレイ・フィルタ・セットを使用するには:

- USE_FILTER_SETプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.USE_FILTER_SET (filter_set => 'replayfilters');
END;
/
```

この例では、USE_FILTER_SETプロシージャはreplayfiltersという名前のフィルタ・セットを使用します。

この例のUSE_FILTER_SETプロシージャは、必須パラメータfilter_setを使用します。このパラメータは、リプレイで使用するフィルタ・セットの名前を指定します。

リプレイのタイムアウト・アクションの設定

この項では、ワークロード・リプレイのタイムアウト・アクションを設定する方法について説明します。リプレイのタイムアウト・アクションを設定すると、リプレイ時の極端に遅いユーザー・コールや、リプレイがハングする原因となるユーザー・コールを中断できます。リプレイのタイムアウト・アクションの設定が必要になるのは、たとえば、データベースのアップグレードの後で、最適でない実行計画によって生じるメモリー集中型問合せを中断するような場合です。

リプレイのタイムアウト・アクションが有効な場合、リプレイ・タイムアウト・アクションで指定された条件を超えてユーザー・コールが遅延すると、ORA-15569エラーでユーザー・コールが終了します。中断されたコールとそのエラーは、エラーの相違として報告されます。

リプレイ・タイムアウトを設定するには:

- SET_REPLAY_TIMEOUTプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.SET_REPLAY_TIMEOUT (
                           enabled => TRUE,
                           min_delay => 20,
                           max_delay => 60,
                           delay_factor => 10);
```

```
END;
/
```

この例のSET_REPLAY_TIMEOUTプロシージャで定義されるリプレイ・タイムアウト・アクションでは、リプレイ時の遅延が60分を超えるか、リプレイ時の遅延が20分を超えかつその経過時間が取得経過時間の10倍より大きいと、ユーザー・コールが中断されます。

この例のSET_REPLAY_TIMEOUTプロシージャでは、次のパラメータを使用します。

- enabled: リプレイのタイムアウト・アクションが有効か無効かを指定するパラメータ。デフォルト値はTRUEです。
- min_delay: コールの遅延の下限を分単位で定義するパラメータ。リプレイのタイムアウト・アクションは、遅延がこの値を超えた場合にのみ実行されます。デフォルト値は10です。
- max_delay: コールの遅延の上限を分単位で定義するパラメータ。遅延がこの値を超えると、リプレイのタイムアウト・アクションが実行され、エラーが発行されます。デフォルト値は120です。
- delay_factor: min_delayとmax_delayの間のコール遅延について、その係数を定義するパラメータ。現在のリプレイの経過時間が、取得の経過時間とこの値を掛けた値より大きい場合、リプレイのタイムアウト・アクションがエラーを発行します。デフォルト値は8です。

リプレイ・タイムアウト・アクションを取得するには:

- GET_REPLAY_TIMEOUTプロシージャを使用します。

```
DECLARE
  enabled        BOOLEAN;
  min_delay      NUMBER;
  max_delay      NUMBER;
  delay_factor   NUMBER;
BEGIN
  DBMS_WORKLOAD_REPLAY.GET_REPLAY_TIMEOUT(enabled, min_delay, max_delay,
                            delay_factor);
END;
/
```

この例のGET_REPLAY_TIMEOUTプロシージャは、次のパラメータを返します。

- enabled: リプレイのタイムアウト・アクションが有効か無効かを返すパラメータ。
- min_delay: コールの遅延の下限を分単位で返すパラメータ。
- max_delay: コールの遅延の上限を分単位で返すパラメータ。
- delay_factor: 遅延係数を返すパラメータ。

ワークロードのリプレイの開始

ワーク・リプレイを開始する前に実行するタスクがあります。たとえば:

- 取得されたワークロードの事前処理。詳細は、「APIを使用したデータベース・ワークロードの事前処理」を参照してください
- リプレイ・データの初期化。詳細は、「リプレイ・データの初期化」を参照してください
- リプレイ・オプションの指定。詳細は、「ワークロードのリプレイ・オプションの設定」を参照してください
- リプレイ・クライアントの起動。詳細は、「リプレイ・クライアントの起動」を参照してください

ノート:

ワークロード・リプレイを開始すると、新しいリプレイ・クライアントはデータベースに接続できなくなります。START_REPLAY プロシージャの実行前に起動されたリプレイ・クライアントのみ、取得したワークロードのリプレイに使用されます。

ワークロード・リプレイを開始するには:

- START_REPLAYプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.START_REPLAY ();
END;
/
```

ノート:

Oracle Database リリース 19c 以降では、DBA_RAT_CAPTURE_SCHEMA_INFO ビューに、SQL 文の login_schema と current_schema の情報が示されます。拡張モードでのリプレイ中に、「ORA-00942: 表またはビューが存在しません」エラーが発生した場合は、DBA_RAT_CAPTURE_SCHEMA_INFO ビューと DBA_WORKLOAD_CAPTURE_SQLTEXT ビューを使用してエラーの原因になった表を調べます。その後で、該当する表に必要な権限をエラーが発生したユーザーに付与して、リプレイを再開してみます。通常は、これで問題が解決します。

ワークロード・リプレイの一時停止

ワークロード・リプレイを一時停止すると、それ以降にリプレイ・クライアントによって発行されるすべてのユーザー・コールは、ワークロード・リプレイが再開されるか取り消されるまで停止されます。すでに進行中のユーザー・コールは完了できます。このオプションを使用すると、リプレイを一時的に停止して変更を行い、その変更がリプレイの残りの部分に与える影響を確認することができます。

ワークロード・リプレイを一時停止するには:

- PAUSE_REPLAYプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PAUSE_REPLAY ();
END;
/
```

ワークロード・リプレイの再開

この項では、一時停止されたワークロード・リプレイを再開する方法を説明します。

ワークロード・リプレイを再開するには:

- RESUME_REPLAYプロシージャを使用します。

```
BEGIN
```

```
  DBMS_WORKLOAD_REPLAY.RESUME_REPLAY ();
END;
/
```

ワークロード・リプレイの取消し

この項では、ワークロード・リプレイを取り消す方法を説明します。

ワークロード・リプレイを取り消すには:

- CANCEL_REPLAYプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.CANCEL_REPLAY ();
END;
/
```

ワークロード・リプレイに関する情報の取得

リプレイ・ディレクトリ・オブジェクト内のワークロード取得に関する情報と、そのディレクトリからのワークロード・リプレイ試行の履歴に関する情報をすべて取得できます。デフォルトでは、ワークロード・リプレイ相違データはロードされませんが、このデータを選択的にロードする選択ができます。

ワークロード・リプレイに関する情報を取得するには:

- DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFOファンクションをコールします。

  GET_REPLAY_INFOファンクションはまず、ワークロードの取得に関する情報が含まれる行をDBA_WORKLOAD_CAPTURESビューにインポートします。デフォルトでは、これまでDBA_WORKLOAD_REPLAYSビューにロードされていないリプレイの情報のみがインポートされます。このファンクションは、DBA_WORKLOAD_REPLAYSビューのCAPTURE_ID列に関連付けられて、取得した情報にアクセスできる、キャプチャ・ディレクトリのcap_idを戻します(統合キャプチャ・ディレクトリの場合、cap_idは0を戻します)。

  GET_REPLAY_INFOファンクションでは、次のパラメータを使用します。

  - replay_dir: ワークロード・リプレイのディレクトリ・オブジェクト名を指定する必須パラメータ。
  - load_divergence: 相違データがロードされるかどうかを指定するオプション・パラメータ。このパラメータのデフォルト値はFALSEです。リプレイ・ディレクトリから取得した各リプレイ試行の行をDBA_WORKLOAD_REPLAY_DIVERGENCEビューにインポートして、相違データをロードするには、このパラメータをTRUEに設定します。あるいは、リプレイ情報が取得された後で、「ワークロード・リプレイの相違データのロード」で説明されているように、LOAD_DIVERGENCEプロシージャを使用して、ディレクトリ・オブジェクト内の1つのリプレイまたはすべてのリプレイの相違データを選択的にロードすることができます。

例12-1 ワークロード・リプレイに関する情報の取得

次の例に、ワークロード取得に関する情報と、jul14という名前のリプレイ・ディレクトリ・オブジェクトのワークロード・リプレイ試行の履歴に関する情報を取得する方法と、情報が取得されたことを確認する方法を示します。

```
DECLARE
  cap_id          NUMBER;
BEGIN
  cap_id := DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO(replay_dir => 'jul14');
```

```
  SELECT capture_id
    FROM dba_workload_replays
   WHERE capture_id = cap_id;
END;
/
```

ワークロード・リプレイの相違データのロード

ワークロード・リプレイの相違データのロードは、リプレイ・ディレクトリから取得した各リプレイ試行の行をDBA_WORKLOAD_REPLAY_DIVERGENCEビューにインポートして、リプレイ試行中の相違のあるコールとエラーに関する情報を表示します。指定したディレクトリ・オブジェクト内の1つのワークロード・リプレイまたはすべてのワークロード・リプレイのいずれの相違データをロードするかを選択できます。

ワークロード・リプレイの相違データをロードするには:

1. 次のパラメータのいずれかを使用して、WORKLOAD_REPLAY.LOAD_DIVERGENCEプロシージャをコールします。
   - replay_id: 相違データをロードするワークロード・リプレイのIDを指定するパラメータ。このパラメータは、1つのワークロード・リプレイの相違データをロードする場合にのみ使用します。
   - replay_dir: ディレクトリ・オブジェクトの名前を指定するパラメータ(値の大/小文字は区別されます)。このパラメータは、指定したディレクトリ・オブジェクト内のすべてのワークロード・リプレイの相違データをロードする場合に使用します。
2. 相違データのロード・ステータスを確認するには、DBA_WORKLOAD_REPLAYSビューのDIVERGENCE_LOAD_STATUS列を問い合せます。

   TRUEの値は、相違データがロードされることを示し、FALSEの値はロードされないことを示します。

例12-2 1つのワークロード・リプレイの相違データのロード

次の例に、replay_idの値が12のワークロード・リプレイの相違データを取得する方法と、相違データがロードされたことを確認する方法を示します。

```
DECLARE
  rep_id         NUMBER;
BEGIN
  rep_id := DBMS_WORKLOAD_REPLAY.LOAD_DIVERGENCE (replay_id => 12);
  SELECT divergence_load_status
    FROM dba_workload_replays
   WHERE capture_id = rep_id;
END;
/
```

ワークロード・リプレイに関する情報の削除

指定したディレクトリ・オブジェクト内の1つのワークロード・リプレイまたはすべてのワークロード・リプレイのいずれかについて取得した情報を削除できます。「ワークロード・リプレイに関する情報の取得」で説明されているように、削除された情報は、GET_REPLAY_INFOファンクションを使用して取得できます。

ワークロード・リプレイに関する情報を削除するには:

1. replay_idパラメータを使用して、DBMS_WORKLOAD_REPLAY.DELETE_REPLAY_INFOプロシージャをコールします。

- replay_id: リプレイ情報を削除するワークロード・リプレイのIDを指定するパラメータ。このパラメータは、1つのワークロード・リプレイの情報を削除する場合に使用します。

2. デフォルトでは、ワークロード・リプレイに関する情報を削除しても、そのデータはディスクからは削除されません。

例12-3 ワークロード・リプレイに関する情報の削除

次の例に、ワークロード取得に関する情報と、IDが2のワークロード・リプレイに対するワークロード・リプレイ試行の履歴に関する情報を削除する方法を示します。リプレイ・データはディスクから削除されないため、「ワークロード・リプレイに関する情報の取得」で説明されているように、GET_REPLAY_INFOファンクションのコールにより取得できます。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.DELETE_REPLAY_INFO (replay_id => 2);
END;
/
```

ワークロードのリプレイのAWRデータのエクスポート

AWRデータをエクスポートすると、ワークロードの詳細な分析が可能になります。このデータは、2つのワークロードの取得(またはリプレイ)に対してAWR期間比較レポートを実行する場合には必須です。

AWRデータをエクスポートするには:

- EXPORT_AWRプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.EXPORT_AWR (replay_id => 1);
END;
/
```

この例では、リプレイIDが1のワークロード・リプレイに対応するAWRスナップショットがエクスポートされ、また、ワークロードのリプレイ中に取得されたSQLチューニング・セットもエクスポートされます。

EXPORT_AWRプロシージャでは、AWRスナップショットをエクスポートするリプレイのIDを指定する必須パラメータreplay_idを使用します。

ノート:

このプロシージャは、対応するワークロードのリプレイが現在のデータベースで実行され、元のリプレイ期間に対応する AWR スナップショットがまだ使用可能である場合にのみ、機能します。

ワークロードのリプレイのAWRデータのインポート

リプレイ・システムからAWRデータをエクスポートすると、AWRデータを別のシステムにインポートできます。AWRデータをインポートすると、ワークロードの詳細な分析が可能になります。このデータは、2つのワークロードの取得(またはリプレイ)に対してAWR期間比較レポートを実行する場合には必須です。

AWRデータをインポートするには:

- IMPORT_AWRファンクションを使用します。

```
CREATE USER capture_awr
```

```
SELECT DBMS_WORKLOAD_REPLAY.IMPORT_AWR (replay_id => 1,
                                        staging_schema => 'capture_awr')
  FROM DUAL;
```

この例では、取得IDが1のワークロード・リプレイに対応するAWRスナップショットがcapture_awrという名前のステージング・スキーマを使用してインポートされます。

この例のIMPORT_AWRプロシージャでは、次のパラメータを使用します。

- 必須パラメータreplay_idでは、AWRスナップショットをインポートするリプレイのIDを指定します。
- 必須パラメータstaging_schemaでは、リプレイ・ディレクトリからSYS AWRスキーマにAWRスナップショットをインポートする際に、ステージング領域として使用される現在のデータベースの既存のスキーマ名を指定します。

ノート:

staging_schema パラメータで指定したスキーマに AWR 表と同じ名前の表が含まれる場合、このファンクションは失敗します。

APIを使用したワークロードのリプレイの監視

この項では、APIおよびビューを使用したワークロード・リプレイの監視方法について説明します。また、「Enterprise Managerを使用したワークロードのリプレイの監視」で説明されているように、Oracle Enterprise Managerを使用してワークロードのリプレイを監視できます。

この項では、次の項目について説明します。

- 違いのあるコールに関する情報の取得
- ビューを使用したワークロードのリプレイの監視

違いのあるコールに関する情報の取得

リプレイ時に、リプレイ・システムと取得システムの間のエラーおよびデータに相違があった場合は、違いのあるコールとして記録されます。

違いのあるコールに関する情報(SQL識別子、SQLテキスト、バインド値など)を取得するには、次のパラメータを使用してGET_DIVERGING_STATEMENTファンクションをコールします。

- replay_idパラメータを違いのあるコールを含むリプレイのIDに設定します。
- stream_idパラメータを違いのあるコールのストリームIDに設定します。
- call_counterパラメータを違いのあるコールのコール・カウンタに設定します。

違いのあるコールに関するこれらの情報を表示するには、DBA_WORKLOAD_REPLAY_DIVERGENCEビューを使用します。次の例は、ファンクション・コールを示しています。

```
DECLARE
  r                 CLOB;
  ls_stream_id      NUMBER;
  ls_call_counter   NUMBER;
  ls_sql_cd         VARCHAR2(20);
```

```
  ls_sql_err           VARCHAR2(512);
  CURSOR c IS
  SELECT stream_id, call_counter
  FROM DBA_WORKLOAD_REPLAY_DIVERGENCE
  WHERE replay_id = 72;
BEGIN
  OPEN c;
  LOOP
  FETCH c INTO ls_stream_id, ls_call_counter;
  EXIT when c%notfound;
  DBMS_OUTPUT.PUT_LINE (ls_stream_id||''||ls_call_counter);
  r:=DBMS_WORKLOAD_REPLAY.GET_DIVERGING_STATEMENT(replay_id => 72,
  stream_id => ls_stream_id, call_counter => ls_call_counter);
  DBMS_OUTPUT.PUT_LINE (r);
  END LOOP;
END;
/
```

関連項目:

- DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

ビューを使用したワークロードのリプレイの監視

この項では、ワークロードのリプレイを監視するために表示できるビューの概要を示します。これらのビューにアクセスするには、DBA権限が必要です。

- DBA_WORKLOAD_CAPTURESビュー: 現在のデータベースで取得されたワークロードの取得をすべて示します。
- DBA_WORKLOAD_FILTERSビュー: 現在のデータベースに定義されたワークロードの取得に使用されるワークロード・フィルタをすべて示します。
- DBA_WORKLOAD_REPLAYSビュー: 現在のデータベースでリプレイされたワークロードのリプレイをすべて示します。
- DBA_WORKLOAD_REPLAY_DIVERGENCEビュー: 違いのあるコールに関する情報(リプレイID、ストリームID、コール・カウンタなど)を表示できます。
- DBA_WORKLOAD_DIV_SUMMARYビューには、DBA_WORKLOAD_REPLAY_DIVERGENCEビューのリプレイの相違情報のサマリーが表示されます。
- DBA_WORKLOAD_REPLAY_FILTER_SETビュー: 現在のデータベースで定義されたワークロード・リプレイに使用されるワークロード・フィルタをすべて示します。
- DBA_WORKLOAD_CONNECTION_MAPビュー: ワークロードのリプレイの接続マッピング情報を示します。
- V$WORKLOAD_REPLAY_THREADビュー: リプレイ・クライアントからのすべてのセッションに関する情報を示します。

関連項目:

- これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください

13 取得およびリプレイ済ワークロードの分析

この章では、様々なデータベース・リプレイ・レポートを使用して、取得およびリプレイしたワークロードを分析する方法を説明します。この章の構成は、次のとおりです。

- ワークロードの取得レポートの使用
- ワークロードのリプレイ・レポートの使用
- リプレイの期間比較レポートの使用
- SQLパフォーマンス・アナライザ・レポートの使用

> ノート:
>
> リプレイの分析が完了したら、ワークロード・ディレクトリ・オブジェクトを別の物理的な場所にバックアップしてから、そのデータベースをワークロード取得時点の元の状態にリストアし、ワークロードのリプレイを繰り返して、システムに対する他の変更をテストすることができます。

ワークロードの取得レポートの使用

ワークロードの取得レポートには、取得されたワークロード統計、取得された上位セッション・アクティビティ、および取得プロセス中に使用されたすべてのワークロード・フィルタが含まれます。

次の項では、ワークロードの取得レポートの生成および使用方法について説明します。

- Enterprise Managerを使用したワークロードの取得レポートへのアクセス
- APIを使用したワークロードの取得レポートの生成
- ワークロードの取得レポートの確認

Enterprise Managerを使用したワークロードの取得レポートへのアクセス

この項では、Oracle Enterprise Managerを使用してワークロードの取得レポートを生成する方法について説明します。

ワークロードの取得レポートを生成するための主ツールは、Oracle Enterprise Managerです。なんらかの理由でOracle Enterprise Managerが使用できない場合は、「APIを使用したワークロードの取得レポートの生成」で説明されているように、APIを使用してワークロードの取得レポートを生成できます。

Enterprise Managerを使用し、ワークロードの取得レポートにアクセスするには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。

2. ステータスが「完了」以外の取得のデータベース・リプレイ・ページにある「取得済ワークロード」タブで、「取得」表から必

要な取得の名前をクリックします。

データベース・リプレイ・ページの「サマリー」タブが表示されます。

3. ワークロードの取得レポートおよびワークロードの取得ASH分析レポートに対するコントロールにアクセスするには、「レポート」タブをクリックします。
    - ワークロードの取得レポートには、取得されたデータ・コンポーネントと取得されなかったデータ・コンポーネントのタイプを示す、詳細な出力が含まれます。
    - 取得ASH分析レポートには、データベース時間を最も消費しているセッションが表示されます。このレポートには積上げグラフが表示され、イベント、アクティビティ・クラス、モジュール/アクション、セッション、インスタンスIDおよびPL/SQLファンクションなどの複数のディメンションのアクティブ・セッション・アクティビティを簡単に可視化できます。

      レポートの「その他のアクティビティ」リストで選択されている場合は、そのアクティビティが取得されていないことを意味します。
4. レポートにアクセスした後、「保存」をクリックして保存できます。

関連項目:

- ワークロードの取得レポートの解釈方法の詳細は、「ワークロードの取得レポートの確認」を参照してください
- ASH(アクティブ・セッション履歴)の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください

APIを使用したワークロードの取得レポートの生成

DBMS_WORKLOAD_CAPTUREパッケージを使用してワークロードの取得レポートを生成できます。また、「Enterprise Managerを使用したワークロードの取得レポートへのアクセス」で説明されているように、Oracle Enterprise Managerを使用してワークロードの取得レポートを生成することも可能です。

最新のワークロードの取得レポートを生成するには:

1. DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFOプロシージャを使用します。

   GET_CAPTURE_INFOプロシージャは、ワークロードの取得に関するすべての情報を収集し、そのワークロードの取得に対応するcap_idを戻します。このファンクションでは、必須パラメータdirを使用します。このパラメータは、ワークロードの取得のディレクトリ・オブジェクト名を指定します。
2. DBMS_WORKLOAD_CAPTURE.REPORTファンクションをコールします。

   REPORTファンクションは、GET_CAPTURE_INFOプロシージャによって戻されたcap_idを使用してレポートを生成します。このファンクションでは、次のパラメータを使用します。
    - capture_id: レポートが生成されるワークロードの取得を含むディレクトリを指定する必須パラメータ。このディレクトリは、ワークロードの取得を含むホスト・システム内の有効なディレクトリである必要があります。このパラメータの値は、GET_CAPTURE_INFOプロシージャによって戻されるcap_idに一致する必要があります。
    - format: レポートの形式を指定する必須パラメータ。有効値は、DBMS_WORKLOAD_CAPTURE.TYPE_TEXTおよびDBMS_WORKLOAD_REPLAY.TYPE_HTMLです。

この例では、GET_CAPTURE_INFOプロシージャがjul14ディレクトリ内のワークロードの取得に関するすべての情報を収集し、その

ワークロードの取得に対応するcap_idを戻します。次にREPORTファンクションは、GET_CAPTURE_INFOプロシージャによって戻されたcap_idを使用してテキスト・レポートを生成します。

```
DECLARE
  cap_id          NUMBER;
  cap_rpt         CLOB;
BEGIN
  cap_id  := DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO(dir => 'jul14');
  cap_rpt := DBMS_WORKLOAD_CAPTURE.REPORT(capture_id => cap_id,
                            format => DBMS_WORKLOAD_CAPTURE.TYPE_TEXT);
END;
/
```

関連項目:

- ワークロードの取得レポートの解釈方法の詳細は、「ワークロードの取得レポートの確認」を参照してください
- DBMS_WORKLOAD_CAPTUREパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

ワークロードの取得レポートの確認

ワークロードの取得レポートには、ワークロードの取得の有効性を評価するために使用できる様々なタイプの情報が含まれます。このレポートに示される情報から、取得したワークロードについて、次の項目を確認できます。

- リプレイする実際のワークロードを表しているかどうか
- 除外するワークロードが含まれていないかどうか
- リプレイできるかどうか

ワークロードの取得レポートに含まれる情報は、次のカテゴリに分類できます。

- ワークロードの取得に関する詳細情報(ワークロードの取得の名前、定義済のフィルタ、日付、時刻、取得のSCNなど)
- ワークロードの取得に関する全般的な統計(取得済の合計DB時間、取得済のログインとトランザクションの数など)と、合計システム・アクティビティに対する割合
- 取得されたワークロードのプロファイル
- バージョン制限のために取得されなかったワークロードのプロファイル
- 定義済のフィルタによって除外された未取得のワークロードのプロファイル
- バックグラウンド・プロセスまたはスケジュール済のジョブで構成された未取得のワークロードのプロファイル

ワークロードのリプレイ・レポートの使用

ワークロード・リプレイ・レポートには、取得システムとリプレイ・システム間のパフォーマンスの違いの測定に使用できる情報が含まれます。

次の項では、ワークロードのリプレイ・レポートの生成および確認方法について説明します。

- Enterprise Managerを使用したワークロード・リプレイ・レポートへのアクセス
- APIを使用したワークロードのリプレイ・レポートの生成
- ワークロードのリプレイ・レポートの確認

Enterprise Managerを使用したワークロード・リプレイ・レポートへのアクセス

この項では、Oracle Enterprise Managerを使用してワークロードのリプレイ・レポートを生成する方法について説明します。

ワークロードのリプレイ・レポートを生成するための主ツールは、Oracle Enterprise Managerです。なんらかの理由でOracle Enterprise Managerが使用できない場合は、「APIを使用したワークロードのリプレイ・レポートの生成」で説明されているように、APIを使用してワークロードのリプレイ・レポートを生成できます

Enterprise Managerを使用したワークロード・リプレイ・レポートにアクセスするには:

1. Enterprise Manager Cloud Controlコンソールの「エンタープライズ」メニューで、「クオリティ管理」、「データベース・リプレイ」の順に選択します。

   「データベース・ログイン」ページが表示されたら、管理者権限のあるユーザーとしてログインします。

   「データベース・リプレイ」ページが表示されます。
2. 「リプレイ・タスク」タブをクリックして、レポートにアクセスするリプレイを選択します。
3. 「レポート」タブをクリックして、個別のレポートにアクセスします。

   完了したワークロードのリプレイに対して表示できるレポートには、次の数種類があります。

   - データベース・リプレイ

     このレポートは、リプレイの相違およびワークロード・プロファイルを含む表形式の完全なリプレイ統計を表示する場合に使用します。

DB Replay Report for task1

DB Name	DB Id	Release	RAC	Replay Name	Replay Status
SIDB	1079228280	12.1.0.1.0	NO	task1	COMPLETED

Replay Information

Information	Replay	Capture
Name	task1	sidb_cap1
Status	COMPLETED	COMPLETED
Database Name	SIDB	SIDB
Database Version	12.1.0.1.0	12.1.0.1.0
Start Time	28-11-12 06:37:11	28-11-12 05:40:47
End Time	28-11-12 06:47:11	28-11-12 05:50:47
Duration	10 minutes 0 seconds	10 minutes 0 seconds
Directory Object	SIDB_REP1	SIDB_REP1
Directory Path	/scratch/wload/sidb_rep1	/scratch/wload/sidb_rep1
AWR DB Id	1079228280	
AWR Begin Snap Id	22	
AWR End Snap Id	23	
Replay Schedule Name		

Replay Options

Option Name	Value
Synchronization	SCN
Connect Time	100%
Think Time	100%
Think Time Auto Correct	TRUE
Number of WRC Clients	1 (1 Completed, 0 Running)

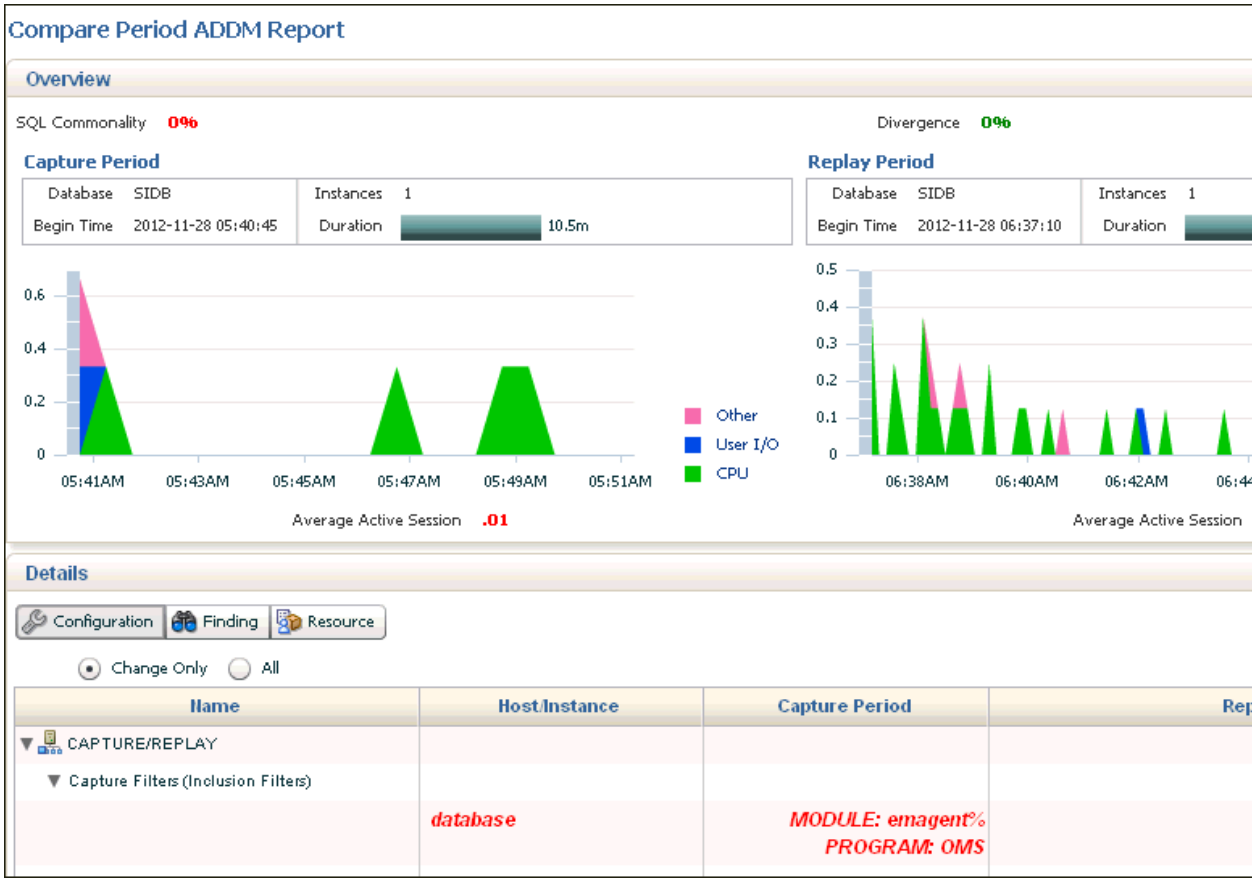
Replay Statistics

Statistic	Replay	Capture
DB Time	3.314 seconds	1.481 seconds
Average Active Sessions	.01	0
User calls	788	788

Replay Divergence Summary

- 期間比較ADDM

このレポートは、1つのワークロード・リプレイとその取得、または同じ取得の別のリプレイとの高レベルの比較を実行する場合に使用します。このレポートでは、5分以上のデータベース時間を含むワークロード・リプレイのみを比較できます。



レポートの次のセクションを調べ、2つの期間の間のパフォーマンス変化とその変化の原因を把握します。

- 概要

  レポートのこの部分に、両方の期間に共通するSQL文の平均的なリソース消費に基づいて基本期間および比較期間を比較できるSQLの共通性が表示されます。

  100%の共通性とは、両方の期間でのワークロードの署名が同一であることを示します。リプレイ中のワークロードが同じ(リプレイ・フィルタを使用していないと想定)であるので、このような場合に100%の共通性が期待されます。0%の値は、特定のワークロード・ディメンションについて2つの期間に共通するものがないことを示します。

  共通性は入力タイプ(つまり実行されているSQL)と、実行中のSQL文の負荷に基づいています。つまり、ある期間でのみ実行されるが、時間がかからないSQL文は共通性に影響を与えません。そのため、一部のSQL文が2つの期間のいずれかでのみ実行されている場合でも、それらのSQL文が多くのリソースを消費していない条件で2つのワークロードの100%の共通性を確保できます。

- 構成

  表示される情報には、インスタンス、ホストおよびデータベース別に分類された各種パラメータのベース期間と比較期間の値が含まれます。

- 結果

  結果にパフォーマンスの向上が示され、システム変更による主なパフォーマンスの差異を識別できる可能性があります。原因を把握してこれを解決すると、マイナスの結果が解消されます。

  基本期間および比較期間に表示される値は、データベース時間に関するパフォーマンスを表します。

  「影響の変更」の値は、ある期間から別の期間のパフォーマンスの変化の規模の測定値を表します。

これは、それぞれの期間で消費したデータベースの合計時間によって測定した問題またはアイテムに適用できます。絶対値は降順でソートされます。

値がプラスの場合は増加、マイナスの場合は減少を示します。たとえば、影響の変化が200%の場合は、期間2が期間1より3倍低速であることを意味します。

ADDMおよびSQLチューニング・アドバイザなどのパフォーマンス・チューニング・ツールを実行して、比較期間の問題を修正し、全体的なシステム・パフォーマンスを向上させることができます。

- リソース

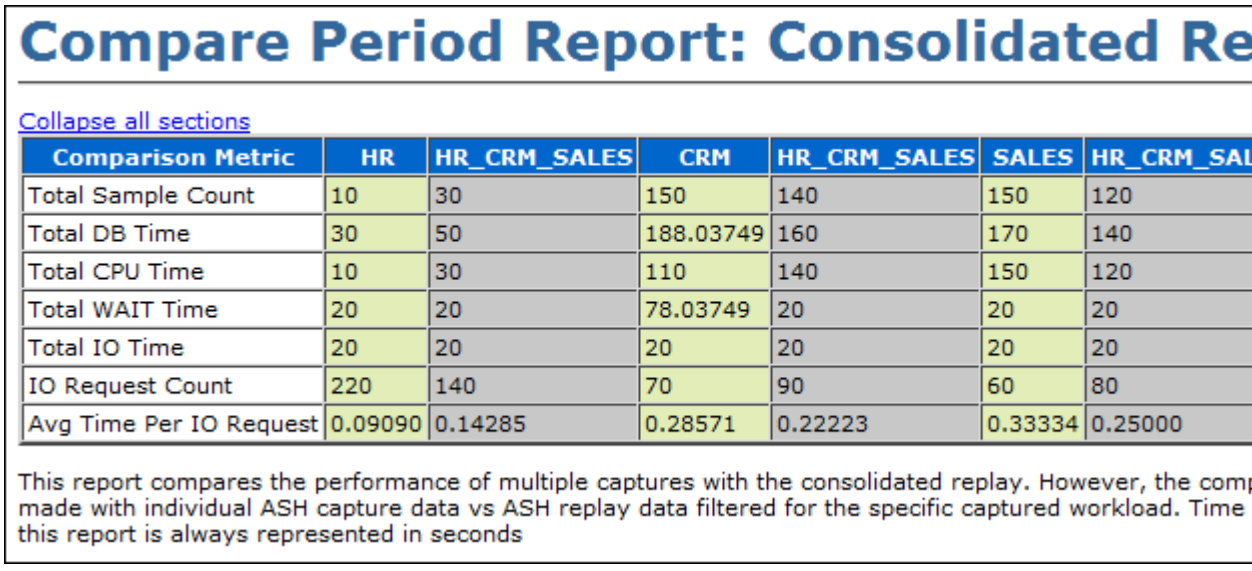
  表示される情報は、両方の期間のデータベース時間分割のサマリーと、CPU、メモリー、I/Oおよび相互接続のリソース使用率を示します(Oracle RACのみ)。

- SQLパフォーマンス・アナライザ

  このレポートは、ワークロード取得のSQLチューニング・セットを、ワークロード・リプレイの別のSQLチューニング・セットと、または2つのワークロード・リプレイの2つのSQLチューニング・セットと比較する場合に使用します。SQLチューニング・セットをデータベース・リプレイと比較すると、SQL文ごとにすべての実行計画が検討され、表示されるのに対し、SQLパフォーマンス・アナライザのテスト実行では、SQLを試行するごとに1SQL文につき1つの実行計画しか生成されないため、前者の方がより多くの情報が得られます。

- リプレイの期間比較

  このレポートは、1つのワークロード・リプレイのAWRデータをその取得、または同じ取得の別のリプレイと比較する場合に使用します。このレポートを実行する前に、取得またはリプレイしたワークロードのAWRデータを事前にエクスポートしておく必要があります。



Compare Period Report: Consolidated Re

Collapse all sections

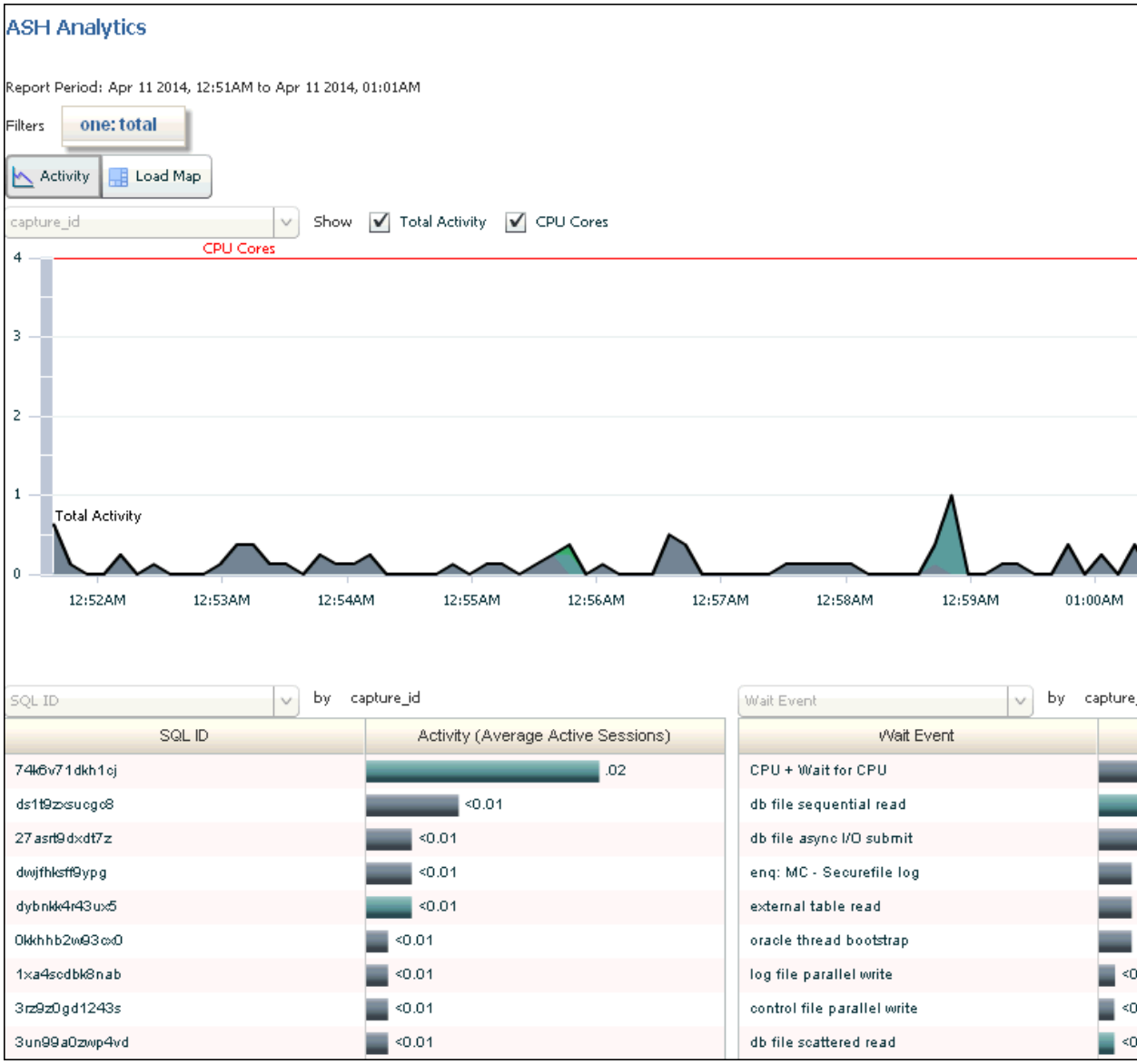
Comparison Metric	HR	HR_CRM_SALES	CRM	HR_CRM_SALES	SALES	HR_CRM_SAL
Total Sample Count	10	30	150	140	150	120
Total DB Time	30	50	188.03749	160	170	140
Total CPU Time	10	30	110	140	150	120
Total WAIT Time	20	20	78.03749	20	20	20
Total IO Time	20	20	20	20	20	20
IO Request Count	220	140	70	90	60	80
Avg Time Per IO Request	0.09090	0.14285	0.28571	0.22223	0.33334	0.25000

This report compares the performance of multiple captures with the consolidated replay. However, the comp
made with individual ASH capture data vs ASH replay data filtered for the specific captured workload. Time
this report is always represented in seconds

このレポートの使用方法の詳細は、「リプレイの期間比較レポートの確認」を参照してください。

- リプレイASH分析

  リプレイASH分析レポートには、ドロップダウン・メニューで選択したカテゴリにリプレイされたワークロードの指定した期間のアクティブ・セッション履歴(ASH)情報が含まれます。このレポートを実行する前に、取得またはリプレイしたワークロードから事前にAWRデータをエクスポートしておく必要があります。



チャートには、待機クラスのワークロード・アクティビティ・ブレークダウン値が表示され、システムに悪影響を与えているトップ・アクティビティ・セッションの詳細な統計が示されます。

システム・アクティビティのグラフィカル・ビューを表示するには、オプションでロード・マップを使用できます。ロード・マップは、選択した期間内でアクティビティの変更を時系列で確認しない場合に1次元または多次元レイアウトのアクティビティを表示する際に役立ちます。

関連項目:

リプレイの期間比較レポートの解釈方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

APIを使用したワークロードのリプレイ・レポートの生成

DBMS_WORKLOAD_REPLAYパッケージを使用してワークロードのリプレイ・レポートを生成できます。また、「Enterprise Managerを使用したワークロード・リプレイ・レポートへのアクセス」で説明されているように、Oracle Enterprise Managerを使用してワークロード・リプレイ・レポートを生成することも可能です。

APIを使用してワークロードの取得に対する最新のワークロードのリプレイ・レポートを生成するには:

1. 「ワークロード・リプレイに関する情報の取得」で説明されているように、ワークロードの取得、およびDBMS_WORKLOAD_REPLAY.GET_REPLAY_INFOファンクションをコールすることによる、リプレイ・ディレクトリ・オブジェクトからのワークロード・リプレイ試行の履歴に関する情報を取得します。

   GET_REPLAY_INFOファンクションは1つのキャプチャ・ディレクトリのcap_idを戻します(統合キャプチャ・ディレクトリの場合、戻されるcap_idは0です)。

2. GET_REPLAY_INFOファンクションにより戻されるcap_idを使用して、問合せを実行し、ワークロードの最新のリプレイの適切なrep_idを戻します。

3. DBMS_WORKLOAD_REPLAY.REPORTファンクションをコールします。

   REPORTファンクションは、SELECT文によって戻されたrep_idを使用してレポートを生成します。

   REPORTファンクションでは、次のパラメータを使用します。

   - replay_id: レポートが生成されるワークロードのリプレイを含むディレクトリを指定する必須パラメータ。このディレクトリは、ワークロードのリプレイを含むホスト・システム内の有効なディレクトリである必要があります。このパラメータの値は、前の問合せで戻されたrep_idと一致する必要があります。
   - format: レポートの形式を指定する必須パラメータ。有効な値は、DBMS_WORKLOAD_REPLAY.TYPE_TEXT、DBMS_WORKLOAD_REPLAY.TYPE_HTMLおよびDBMS_WORKLOAD_REPLAY.TYPE_XMLです。

この例では、GET_REPLAY_INFOファンクションが、ワークロードの取得に関するすべての情報と、jul14リプレイ・ディレクトリ・オブジェクトからのすべてのワークロード・リプレイ試行の履歴に関する情報を収集します。このファンクションは、DBA_WORKLOAD_REPLAYSビューのCAPTURE_ID列に関連付けられて、取得した情報にアクセスできる、キャプチャ・ディレクトリのcap_idを戻します。SELECT文は最新のワークロードのリプレイの適切なrep_idを戻します。次に、REPORTファンクションは、SELECT文によって戻されたrep_idを使用してHTMLレポートを生成します。

```
DECLARE
  cap_id          NUMBER;
  rep_id          NUMBER;
  rep_rpt         CLOB;
BEGIN
  cap_id := DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO(replay_dir => 'jul14');
  /* Get the latest replay for that capture */
  SELECT max(id)
  INTO   rep_id
  FROM   dba_workload_replays
  WHERE  capture_id = cap_id;

  rep_rpt := DBMS_WORKLOAD_REPLAY.REPORT(replay_id => rep_id,
                             format => DBMS_WORKLOAD_REPLAY.TYPE_HTML);
END;
/
```

関連項目:

- ワークロードのリプレイ・レポートの解釈方法の詳細は、「ワークロードのリプレイ・レポートの確認」を参照してください
- DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

ワークロードのリプレイ・レポートの確認

テスト・システムでワークロードをリプレイすると、取得した内容とリプレイされる内容に多少の相違が発生する場合があります。リプレイの相違が発生する要因は多数ありますが、それらはワークロードのリプレイ・レポートを使用して分析できます。ワークロードのリプレイ・レポートに含まれている情報は、パフォーマンスおよびリプレイの相違で構成されています。

パフォーマンスの相違は、データベースのパフォーマンス全体に影響を与える新しいアルゴリズムがリプレイ・システムに導入された場合に発生する可能性があります。たとえば、より新しいバージョンのOracle Databaseでワークロードがリプレイされると、新しいアルゴリズムによって特定のリクエストがより高速に実行される可能性があり、この高速の実行が相違として現れます。この場合は、望ましい相違です。

データの相違は、DMLまたはSQL問合せの結果が、ワークロードですでに取得された結果と一致しない場合に発生します。たとえば、リプレイ時にSQL文によって取得時より少ない数の行が戻されることがあります。

エラーの相違は、リプレイされたデータベース・コールで次のような状況が発生した場合に現れます。

- 取得されていない新しいエラーが発生した場合
- 取得されたエラーが発生しなかった場合
- 取得されたエラーとは異なるエラーが発生した場合

ワークロードのリプレイ・レポートに含まれる情報は、次のカテゴリに分類されます。

- ワークロードのリプレイおよびワークロードの取得に関する詳細情報(ジョブ名、ステータス、データベース情報、各プロセスの継続時間と時刻、ディレクトリ・オブジェクト、ディレクトリ・パスなど)
- ワークロードのリプレイに関して選択されたリプレイ・オプションおよび起動されたリプレイ・クライアントの数
- ワークロードのリプレイおよびワークロードの取得に関する全般的な統計(取得済およびリプレイ済の合計DB時間、取得済およびリプレイ済のログインとトランザクションの数など)と、合計システム・アクティビティに対する割合
- リプレイされたワークロードのプロファイル
- リプレイの相違
- エラーの相違
- DMLおよびSQL問合せのデータの相違

Oracle Databaseリリース19c以降、ワークロード・リプレイ・レポートには、遅いワークロード・リプレイの診断に必要な情報が示されます。このワークロード・リプレイ・レポートに含まれる情報には、次の追加セクションがあります。

- リプレイ・セッション
- 同期化
- 追跡されたコミット
- セッションの失敗

リプレイ・セッション

リプレイ・セッションのセクションには、進行中および完了済のリプレイ・セッションに関する統計情報(上位イベント、リプレイ速度が下位のセッション、リプレイ速度が上位のセッションなど)が含まれています。進行中のリプレイ・セッションに関連する統計情報は、リプレイが進行中の場合にのみ示されます。この情報を使用すると、進行中および完了済リプレイ・セッションの上位イベント、

取得よりも遅いリプレイ・セッション、取得とリプレイのセッション経過時間の比較を確認できます。

同期化

ワークロードのリプレイ時には、取得したSQL文の実行順序が維持されます。リプレイ中にSQL文の実行の遅延またはブロックが発生すると、ワークロード・リプレイが遅くなります。

同期化のセクションには、同期化されたSQL文のいずれかの実行をブロックしているセッションに関する情報が示されます。この情報は、SQL文の実行がブロックされている場合にのみ表示されます。このセクションの情報を使用すると、ワークロードのリプレイがブロックされている理由やワークロードの取得よりも遅くなっている理由を確認できます。このセクションには、上位イベントと、同期されたSQL文を実行しているセッションに関連する上位イベントとともに上位のSQL文も示されます。

追跡されたコミット

ワークロードの取得中に実行されたコミットとその実行間で経過した時間は、データベース・リプレイによって追跡されます。遅いコミットを検出して、取得済ワークロードのうちワークロード・リプレイを遅くしている部分を識別できます。

追跡されたコミットのセクションには、取得したコミットの数とコミットの実行時刻に関する情報が含まれています。この情報を使用して、ワークロード・リプレイの動作がワークロードの取得よりも遅くなっているかどうかを調べます。

セッションの失敗

ユーザー・セッションのワークロード・リプレイが失敗すると、そのセッションの残りのコールのリプレイはスキップされます。セッションの失敗のセクションには、セッションが失敗したときに実行されていたSQL文、ファイルID、失敗したセッションのコール・カウンタ、および実行されなかったコールの合計数に関する情報が示されます。

リプレイの期間比較レポートの使用

リプレイの期間比較レポートは様々な目的に使用できます。たとえば、リプレイの期間比較レポートを使用すると、次のパフォーマンスを比較できます。

- ワークロード・リプレイとそのワークロード取得
- ワークロード・リプレイと、同じワークロード取得の別のリプレイ
- 複数のワークロード取得と統合リプレイ

次の項では、リプレイの期間比較レポートの生成および確認方法について説明します。

- APIを使用したリプレイの期間比較レポートの生成
- リプレイの期間比較レポートの確認

関連項目:

- データベース統合リプレイの詳細は、「データベース統合リプレイの使用」を参照してください

APIを使用したリプレイの期間比較レポートの生成

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、リプレイの期間比較レポートを生成する方法について説明します。

このレポートでは、データベース時間が5分以上のワークロード・リプレイのみを比較できます。

リプレイの期間比較レポートを生成するには、DBMS_WORKLOAD_REPLAY.COMPARE_PERIOD_REPORTプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.COMPARE_PERIOD_REPORT (
                              replay_id1 => 12,
                              replay_id2 => 17,
                              format => DBMS_WORKLOAD_CAPTURE.TYPE_HTML,
                              result => :report_bind);
END;
/
```

この例のCOMPARE_PERIOD_REPORTプロシージャは、リプレイID 12のワークロード・リプレイをID 17の別のリプレイと比較する、HTML形式のリプレイの期間比較レポートを生成します。

この例のCOMPARE_PERIOD_REPORTプロシージャでは、次のパラメータを使用します。

- replay_id1: レポートの生成対象となる変更後のワークロード・リプレイの数値IDを指定するパラメータ。このパラメータは必須です。
- replay_id2: レポートの生成対象となる変更前のワークロード・リプレイの数値IDを指定するパラメータ。このパラメータを指定しない場合、ワークロードの取得との比較が実行されます。
- format: レポートの形式を指定するパラメータ。有効な値は、DBMS_WORKLOAD_CAPTURE.TYPE_HTML(HTMLの場合)と、DBMS_WORKLOAD_CAPTURE.TYPE_XML(XMLの場合)などです。このパラメータは必須です。
- result: レポートの出力を指定するパラメータ。

関連項目:

- リプレイの期間比較レポートの解釈方法の詳細は、「リプレイの期間比較レポートの確認」を参照してください
- DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

リプレイの期間比較レポートの確認

リプレイの期間比較レポートを確認することで、リプレイの相違が発生しているか、および大幅なパフォーマンスの変化があったかを判断できます。

行った比較の種類に応じて、次の3つのいずれかのリプレイの期間比較レポートが生成されます。

- 取得とリプレイ

  このレポート・タイプでは、ワークロード・リプレイのパフォーマンスを取得済ワークロード比較します。

- リプレイとリプレイ

  このレポート・タイプでは、同一のワークロード取得の2つのワークロード・リプレイのパフォーマンスを比較します。

- 統合リプレイ

  このレポート・タイプでは、複数のワークロード取得のパフォーマンスを統合リプレイと比較します。このレポート・タイプには、

ASH Data Comparisonセクションしかありません。このレポート・タイプの詳細は、「データベース統合リプレイのレポート作成および分析」を参照してください。

どのタイプのリプレイの期間比較レポートにも、比較対象の2回の実行の最も重要な変更についての情報が含まれます。この情報を使用すると、実行する適切なアクションを決定できます。たとえば、新しい同時実行の問題が検出された場合、自動データベース診断モニター(ADDM)レポートを確認して問題を診断します。SQLのパフォーマンスに関する新しい問題が検出された場合、SQLチューニング・アドバイザを実行し問題を修正します。

リプレイの期間比較レポート内の情報は、次のセクションに分類できます。

- 一般情報
- リプレイの相違
- メイン・パフォーマンス統計
- 上位SQL/コール
- ハードウェア使用率の比較
- ADDMの比較
- ASHデータ比較

一般情報

このセクションには、レポートで比較される2つの実行に関するメタデータが含まれています。2つの実行間でのinit.oraパラメータの変更もここに示されます。テストしたシステムの変更が実行されたかどうか確認するには、このセクションを参照してください。

リプレイの相違

このセクションには、最初の実行と比較した2番目の実行の相違分析が示されます。分析に大幅な相違がある場合、相違の完全なレポートを確認します。

メイン・パフォーマンス統計

このセクションには、2つの実行間の高度なパフォーマンス統計比較(データベース時間の変化など)が示されます。比較でデータベース時間に大きな違いがない場合、2回の実行間のパフォーマンスは通常類似しています。データベース時間に大幅な違いがある場合、統計を確認し、この大きな違いの原因となっているコンポーネント(CPUまたはユーザーI/O)を特定します。

上位SQL/コール

このセクションでは、個々のSQL文の1回の実行と次の実行におけるパフォーマンス変化を比較します。SQL文は、データベース時間の合計変化の順に並べられます。データベース時間が最も低下した上位のSQL文は、SQLチューニングを行う最有力の対象となります。

ハードウェア使用率の比較

このセクションでは、2回の実行におけるCPUとI/O使用量を比較します。すべてのインスタンスのCPU数値が合計され、インスタンス間のCPUの使用量の平均が算出されます。

データ・ファイルおよび一時ファイルのI/O統計が示されます。1ブロックの読取り時間が高い値である場合(10ミリ秒を大幅に超える場合)、システムがI/Oバウンドであることが示唆されます。この場合、読み書きの合計時間を確認し、遅延の原因がI/Oリクエストが多いためまたはI/Oスループットが悪いためでないか判断します。

ADDMの比較

このセクションでは、2回の実行のADDM分析の比較を影響の絶対差分順に示しています。2回の実行のADDM結果を比較し、1回の実行のみに存在していた可能性のある問題を特定できます。データベースのパフォーマンスを向上させるためにシステム変更をテストした場合、期待していた向上がADDMの結果にあったかを確認できます。

関連項目:

- ADDM分析の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください

ASHデータ比較

このセクションでは、2回の実行のASHデータを比較します。表には、比較期間の開始時間および終了時間が示されます。これらの時間は、比較している2回の実行の取得およびリプレイ時間とは一致しない場合があります。かわりに、これらの時間はASHサンプルが取得された時間を示します。

ASH Data Comparisonセクションには、次のサブセクションが含まれます。

- サマリーの比較
- 上位SQL
- 長時間実行されているSQL
- 共通のSQL
- 上位オブジェクト

関連項目:

- ASHの詳細はOracle Databaseパフォーマンス・チューニング・ガイドを参照

サマリーの比較

このセクションには、データベース時間および待機時間の内訳に基づく2回の実行のアクティビティの要約があります。たとえば:

- データベース時間の内訳は、データベース時間に対するCPU使用量、待機時間、I/Oリクエストの内訳を示します。

  図13-1に、サンプル・レポートのデータベース時間の内訳サブセクションを示します。

  図13-1 データベース時間の内訳

(-) DB Time Distribution

Compare Attr	Time Period1	Time Period2
Total Sample Count	2050	7490
Total DB Time	2519.88801	10164.33483
Total CPU Time	2260	9130
Total WAIT Time	259.88801	1034.33483
Total IO Time	10	770
IO Request Count	2924	329819
Avg Time Per IO Request	00000000.00342	00000000.00233

- 待機時間の内訳は、待機イベントに対する合計待機時間の内訳を示します。2回の実行の上位の待機イベント・クラス、イベント名、イベント数が示されます。

図13-2に、サンプル・レポートの待機時間の内訳サブセクションを示します。

図13-2 待機時間の内訳

(-) Wait Time Distribution

Time Period (1)

Event(1)	Wait time(1)	Event Count(1)
log file switch (checkpoint incomplete),Configuration	239.88801	12
direct path write temp,User I/O	10	2924
log buffer space,Configuration	10	6

Time Period (2)

Event(2)	Wait time(2)	Event Count(2)
direct path read,User I/O	590	185443
log file switch (checkpoint incomplete),Configuration	114.33483	20
direct path read temp,User I/O	80	132686
free buffer waits,Configuration	70	435
direct path write temp,User I/O	50	73
db file sequential read,User I/O	50	11617
log file switch completion,Configuration	30	44
log buffer space,Configuration	30	34
latch: row cache objects,Concurrency	10	28
undo segment extension,Configuration	10	4

上位SQL

このセクションでは、2回の実行でデータベース時間、CPU時間、待機時間の合計が上位のSQL文を示します。

長時間実行されているSQL

このセクションでは、2回の実行において実行時間が長い上位のSQL文を示します。長時間実行されるSQL文ごとに、最大、最小および平均応答時間などの問合せの詳細が記載されています。

共通のSQL

このセクションでは、2回の実行に共通するSQL文を抽出し、平均応答時間およびデータベース時間の合計の変化における上位共通のSQL文を示します。

上位オブジェクト

このセクションには、2回の実行で待機時間の合計が上位のオブジェクトの詳細を示します。

図13-3に、サンプル・レポートの「上位オブジェクト」セクションを示します。

図13-3 上位オブジェクト

(-) Top Objects By Total Wait time

Time Period (1)

Object Id(1)	Object Name(1)	Wait Time(1)	Owner(1)
89843	CR2	249.88801	SYS

Time Period (2)

Object Id(2)	Object Name(2)	Wait Time(2)	Owner(2)
89843	CR2	700	SYS
89844	CR3	100	SYS
89842	CR1	50	SYS
89842	CR1	50	SYS

SQLパフォーマンス・アナライザ・レポートの使用

SQLパフォーマンス・アナライザ・レポートは、ワークロード・リプレイのSQLチューニング・セットを、ワークロード取得の別のSQLチューニング・セットと、または2つのワークロード取得の2つのSQLチューニング・セットと比較する場合に使用します。

SQLパフォーマンス・アナライザのテスト実行では、各SQL試行のSQL文ごとに1つの実行計画のみを生成するのに対し、SQLチューニング・セットをデータベース・リプレイと比較すると、各SQL文のすべての実行計画が考慮されて示されるため、SQLパフォーマンス・アナライザのテスト実行よりも詳しい情報が得られます。

APIを使用したSQLパフォーマンス・アナライザ・レポートの生成

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用してSQLパフォーマンス・アナライザ・レポートを生成する方法について説明します。

SQLパフォーマンス・アナライザ・レポートを生成するには、DBMS_WORKLOAD_REPLAY.COMPARE_SQLSET_REPORTプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.COMPARE_SQLSET_REPORT (
                              replay_id1 => 12,
                              replay_id2 => 17,
                              format => DBMS_WORKLOAD_CAPTURE.TYPE_HTML,
                              result => :report_bind);
END;
/
```

この例のCOMPARE_SQLSET_REPORTプロシージャは、リプレイIDが12のワークロード・リプレイ時に取得されたSQLチューニング・セットを、リプレイIDが17のワークロード・リプレイ時に取得されたSQLチューニング・セットと比較する、HTML形式のSQLパフォーマンス・アナライザ・レポートを生成します。

この例のCOMPARE_SQLSET_REPORTプロシージャでは、次のパラメータを使用します。

- replay_id1: レポートの生成対象となる変更後のワークロード・リプレイの数値IDを指定するパラメータ。このパラメータは必須です。
- replay_id2: レポートの生成対象となる変更後のワークロード・リプレイの数値IDを指定するパラメータ。このパラメー

タを指定しない場合、ワークロードの取得との比較が実行されます。

- format: レポートの形式を指定するパラメータ。有効な値は、DBMS_WORKLOAD_CAPTURE.TYPE_HTML(HTMLの場合)、DBMS_WORKLOAD_CAPTURE.TYPE_XML(XMLの場合)およびDBMS_WORKLOAD_CAPTURE.TYPE_TEXT(テキストの場合)です。このパラメータは必須です。
- result: レポートの出力を指定するパラメータ。

関連項目:

- SQLパフォーマンス・アナライザ・レポートの解釈方法の詳細は、「コマンドラインを使用したSQLパフォーマンス・アナライザ・レポートの確認」を参照してください
- DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

14 ワークロード・インテリジェンスの使用

ワークロード・インテリジェンスでは、取得したワークロードに格納されているデータを分析し、ワークロードの有意なパターンを特定します。この章では、次の内容で、ワークロード・インテリジェンスの使用方法を説明します。

- ワークロード・インテリジェンスの概要
- ワークロード・インテリジェンスを使用した取得済ワークロードの分析
- 例: ワークロード・インテリジェンスの結果

ワークロード・インテリジェンスの概要

ワークロードを取得すると、取得したワークロードに関する情報を含む取得ファイルと呼ばれる多数のバイナリ・ファイルが生成されます。取得ファイルに保存されるこの情報を使用すると、取得したワークロードを後でリプレイする際に現実的に再生できます。取得済のワークロードに記録されているそれぞれのクライアント・リクエストから、SQLテキスト、バインド値、トランザクション情報、タイミング・データ、アクセスしたオブジェクトの識別子およびリクエストに関するその他の情報が取得されます。

ワークロード・インテリジェンスでは、取得ファイル内に保存されたこの情報を次などその他の用途に使用できます。

- ワークロードの分析およびモデル化
- ワークロードの有意なパターンおよび傾向の検出
- ワークロードの取得時に本番システムで実行されていたものの視覚化

この項では、次の内容で、ワークロード・インテリジェンスについて説明します。

- ワークロード・インテリジェンスについて
- ワークロード・インテリジェンスの使用事例
- ワークロード・インテリジェンスの使用要件

関連項目:

- ワークロードの取得の詳細は、「ワークロードの取得」を参照してください
- ワークロードのリプレイの詳細は、「ワークロードのリプレイ」を参照してください

ワークロード・インテリジェンスについて

ワークロード・インテリジェンスとは、取得したワークロードに格納されているデータを分析するJavaプログラムのスイートです。これらのJavaプログラムは、ワークロードの取得時に記録されたデータに作用して、ワークロードを示すモデルを作成します。これらのモデルは、ワークロードの一部として実行されるテンプレートの有意なパターンを特定する助けとなります。

テンプレートとは、読取り専用のSQL文または1つ以上のSQL文で構成される全トランザクションです。2つのSQL文(またはトランザクション)が非常に類似している場合、同じテンプレートによって表されています。

ワークロード・インテリジェンスでは、テンプレートのパターンと対応するSQL文を調べ、取得したワークロードを視覚的に把握しやすくします。パターンの実行回数やそのパターンの実行で消費されたデータベース時間など重要な統計をパターンごとに確認できます。

ワークロード・インテリジェンスの使用事例

ワークロード・インテリジェンスを使用して、取得済ワークロードで有意なパターンを検出できます。

本番システムで実行されるSQL文は通常ユーザーが入力するものではなく、データベース・サーバーに接続されているアプリケーション・サーバーで実行されている1つ以上のアプリケーションから入力されます。アプリケーションには、通常このようなSQL文は有限数あります。特定の文のすべての実行で様々なバインド値が使用された場合でも、そのSQLテキストは基本的に同じです。

ユーザーのアプリケーションへの入力に応じ、データベースに対して、アプリケーション・コードで定義された特定の順序で発行される1つ以上のSQL文を含むコード・パスが実行されます。ユーザーが頻繁に行うアクションは、常に実行されるアプリケーションのコード・パスに対応します。このような頻繁に実行されるコード・パスによって、データベースによって特定の順序で実行されるSQL文のパターンが頻繁に生成されます。ワークロード・インテリジェンスは、取得済ワークロードを分析して、このようなパターンを検出し、関連する実行統計と関連付けます。つまり、ワークロード・インテリジェンスでは、取得ファイルに保存されているこの情報を使用し、ワークロードの取得時に本番システムで実行されていたアプリケーションの有意なコード・パスによって生成されるパターンを検出します。ワークロード・インテリジェンスは、これをアプリケーションの情報をまったく使用せず行います。

ワークロード・インテリジェンスを使用し、有意なパターンを検出することにより、次が可能となります。

- ワークロードの取得時、データベースで実行されていたものを視覚化しやすくなります。
- 最適化に使用できる情報をさらに得ることができます。
- SQL文は分離されず、まとめられているので、コンテキストが把握しやすくなります。

ワークロード・インテリジェンスの使用要件

ワークロード・インテリジェンスは取得ファイルに保存された情報を使用するもので、ワークロード・リプレイを使用したワークロードの実行は必要としません。また、ワークロード・インテリジェンスでは、ユーザー・スキーマ、ユーザー・データまたは本番システムへの接続を必要としません。ワークロード・インテリジェンスの実行は大量のリソースを消費する可能性があるため、本番システムでオーバーヘッドを回避するには、特に取得済ワークロードが大きい場合、テスト・システムに取得ファイルをコピーして、ワークロード・インテリジェンスを使用することをお薦めします。

ワークロード・インテリジェンスを構成するJavaプログラムを起動するために必要なJavaクラスは、$ORACLE_HOME/rdbms/jlib/dbrintelligence.jarに含まれています。クラスパスには、$ORACLE_HOME/rdbms/jlib/dbrparser.jarと$ORACLE_HOME/jdbc/lib/ojdbc6.jarの2つのjarファイルも含まれている必要があります。

ワークロード・インテリジェンスでは、内部でいくつかのSYS表とビューも使用します。

ワークロード・インテリジェンスを使用した取得済ワークロードの分析

この項では、ワークロード・インテリジェンスを使用した取得済ワークロードの分析のためのステップについて説明します。たとえば:

ワークロード・インテリジェンスを使用し、取得済ワークロードを分析するには:

1. 「ワークロード・インテリジェンスのデータベース・ユーザーの作成」で説明されているように、ワークロード・インテリジェンスを使用する適切な権限を持つデータベース・ユーザーを作成します。
2. 「ワークロード・インテリジェンスのジョブの作成」で説明されているように、LoadInfo Javaプログラムを実行し、新しい

ワークロード・インテリジェンス・ジョブを作成します。

3. 「ワークロード・モデルの生成」で説明されているように、BuildModel Javaプログラムを実行し、ワークロードのモデルを生成します。
4. 「ワークロード内のパターンの識別」で説明されているように、FindPatterns Javaプログラムを実行し、ワークロードで発生するパターンをテンプレートから識別します。
5. 「ワークロード・インテリジェンス・レポートの生成」で説明されているように、GenerateReport Javaプログラムを実行し、結果を表示するレポートを生成します。

ワークロード・インテリジェンスのデータベース・ユーザーの作成

ワークロード・インテリジェンスの使用を開始する前に、適切な権限を持つデータベース・ユーザーを作成する必要があります。

例14-1に、ワークロード・インテリジェンスを使用可能なデータベース・ユーザーを作成する方法を示します。

例14-1 ワークロード・インテリジェンスのデータベース・ユーザーの作成

```
create user workintusr identified by password;
grant create session to workintusr;
grant select,insert,alter on WI$_JOB to workintusr;
grant insert,alter on WI$_TEMPLATE to workintusr;
grant insert,alter on WI$_STATEMENT to workintusr;
grant insert,alter on WI$_OBJECT to workintusr;
grant insert,alter on WI$_CAPTURE_FILE to workintusr;
grant select,insert,alter on WI$_EXECUTION_ORDER to workintusr;
grant select,insert,update,delete,alter on WI$_FREQUENT_PATTERN to workintusr;
grant select,insert,delete,alter on WI$_FREQUENT_PATTERN_ITEM to workintusr;
grant select,insert,delete,alter on WI$_FREQUENT_PATTERN_METADATA to workintusr;
grant select on WI$_JOB_ID to workintusr;
grant execute on DBMS_WORKLOAD_REPLAY to workintusr;
```

ワークロード・インテリジェンスのジョブの作成

ワークロード・インテリジェンスのジョブを作成するには、LoadInfoプログラムを使用します。LoadInfoは、ワークロード・インテリジェンスのアルゴリズムを適用する新しいタスクを作成するJavaプログラムです。このプログラムは、取得ディレクトリにあるプログラムを解析し、ワークロード・インテリジェンスの実行に必要な関連情報を内部表に格納します。

LoadInfoプログラムでは、次の構文を使用します。

```
java oracle.dbreplay.workload.intelligence.LoadInfo -cstr connection_string -user username -job
job_name -cdir capture_directory
java oracle.dbreplay.workload.intelligence.LoadInfo -version
java oracle.dbreplay.workload.intelligence.LoadInfo -usage
```

LoadInfoプログラムでは、次のオプションをサポートします。

- -cstr

  ワークロード・インテリジェンスが実行に必要な情報と中間結果を格納するデータベースへのJDBC接続文字列を指定します(たとえば、jdbc:oracle:thin@hostname:portnum:ORACLE_SID)。

- -user

  データベース・ユーザー名を指定します。ユーザーには、ワークロード・インテリジェンスを使用する特定の権限が必要で

す。

適切な権限を持つデータベース・ユーザーの作成方法の詳細は、「ワークロード・インテリジェンスのデータベース・ユーザーの作成」を参照してください。

- -job

  ワークロード・インテリジェンスのジョブを一意に識別する名前を指定します。

- -cdir

  ワークロード・インテリジェンスで分析する取得ディレクトリのオペレーティング・システム・パスを指定します。

- -version

  LoadInfoプログラムのバージョン情報を示します。

- -usage

  LoadInfoプログラムのコマンドライン・オプション情報を示します。

例14-2に、LoadInfoプログラムを使用し、wijobsalesというワークロード・インテリジェンスのジョブを作成する方法を示します。

例14-2 ワークロード・インテリジェンスのジョブの作成

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.LoadInfo -job wijobsales -cdir
/test/captures/sales -cstr jdbc:oracle:thin:@myhost:1521:orcl -user workintusr
```

ワークロード・モデルの生成

ワークロード・モデルを生成するには、BuildModelプログラムを使用します。BuildModelは、取得済ワークロード(このデータは、LoadInfoプログラムで生成される必要があります)からデータを読み取り、ワークロードを表すモデルを生成するJavaプログラムです。このモデルは、ワークロードで発生した頻繁に出現するテンプレート・パターンを識別するために使用できます。

BuildModelプログラムでは、次の構文を使用します。

```
java oracle.dbreplay.workload.intelligence.BuildModel -cstr connection_string -user username -job
job_name
java oracle.dbreplay.workload.intelligence.BuildModel -version
java oracle.dbreplay.workload.intelligence.BuildModel -usage
```

BuildModelプログラムでは、次のオプションをサポートします。

- -cstr

  ワークロード・インテリジェンスが実行に必要な情報と中間結果を格納するデータベースへのJDBC接続文字列を指定します(たとえば、jdbc:oracle:thin@hostname:portnum:ORACLE_SID)。

- -user

  データベース・ユーザー名を指定します。ユーザーには、ワークロード・インテリジェンスを使用する特定の権限が必要です。

  適切な権限を持つデータベース・ユーザーの作成方法の詳細は、「ワークロード・インテリジェンスのデータベース・ユーザーの作成」を参照してください。

- -job

ワークロード・インテリジェンスのジョブを一意に識別する名前を指定します。

- -version

  BuildModelプログラムのバージョン情報を示します。

- -usage

  BuildModelプログラムのコマンドライン・オプション情報を示します。

例14-3に、BuildModelプログラムを使用し、ワークロード・モデルを生成する方法を示します。

例14-3 ワークロード・モデルの生成

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.BuildModel -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr
```

ワークロード内のパターンの識別

ワークロード内のパターンを識別するには、FindPatternsプログラムを使用します。FindPatternsは、取得済ワークロード(このデータは、LoadInfoプログラムで生成される必要があります)およびこれに対応するワークロード・モデル(このワークロード・モデルは、BuildModelプログラムで生成される必要があります)からデータを読み取り、ワークロードで発生した頻繁に出現するテンプレート・パターンを識別するJavaプログラムです。

FindPatternsプログラムでは、次の構文を使用します。

```
java oracle.dbreplay.workload.intelligence.FindPatterns -cstr connection_string
-user username -job job_name -t threshold
java oracle.dbreplay.workload.intelligence.FindPatterns -version
java oracle.dbreplay.workload.intelligence.FindPatterns -usage
```

FindPatternsプログラムでは、次のオプションをサポートします。

- -cstr

  ワークロード・インテリジェンスが実行に必要な情報と中間結果を格納するデータベースへのJDBC接続文字列を指定します(たとえば、jdbc:oracle:thin@hostname:portnum:ORACLE_SID)。

- -user

  データベース・ユーザー名を指定します。ユーザーには、ワークロード・インテリジェンスを使用する特定の権限が必要です。

  適切な権限を持つデータベース・ユーザーの作成方法の詳細は、「ワークロード・インテリジェンスのデータベース・ユーザーの作成」を参照してください。

- -job

  ワークロード・インテリジェンスのジョブを一意に識別する名前を指定します。

- -t

  あるテンプレートから次のテンプレートへの移行が同じパターン内であるか、2つのパターンの境界にあるかを定義する際のしきい値とする確率を指定します。有効な値は、[0.0, 1.0]の範囲の実数です。これは任意指定の値ですが、デフォルト値は0.5です。

- -version

  FindPatternsプログラムのバージョン情報を示します。

- -usage

  FindPatternsプログラムのコマンドライン・オプション情報を示します。

例14-4に、FindPatternsプログラムを使用し、ワークロード内のよく出現するパターンを識別する方法を示します。

例14-4 ワークロード内のパターンの識別

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.FindPatterns -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr -t 0.2
```

ワークロード・インテリジェンス・レポートの生成

ワークロード・インテリジェンスのレポートを生成するには、GenerateReportプログラムを使用します。GenerateReportは、ワークロード・インテリジェンスの結果を示すレポートを生成するJavaプログラムです。ワークロード・インテリジェンスのレポートは、ワークロード内のパターンを示すHTMLページです。

GenerateReportプログラムでは、次の構文を使用します。

```
java oracle.dbreplay.workload.intelligence.GenerateReport -cstr connection_string
-user username -job job_name -top top_patterns -out filename
java oracle.dbreplay.workload.intelligence.GenerateReport -version
java oracle.dbreplay.workload.intelligence.GenerateReport -usage
```

GenerateReportプログラムでは、次のオプションをサポートします。

- -cstr

  ワークロード・インテリジェンスが実行に必要な情報と中間結果を格納するデータベースへのJDBC接続文字列を指定します(たとえば、jdbc:oracle:thin@hostname:portnum:ORACLE_SID)。

- -user

  データベース・ユーザー名を指定します。ユーザーには、ワークロード・インテリジェンスを使用する特定の権限が必要です。

  適切な権限を持つデータベース・ユーザーの作成方法の詳細は、「ワークロード・インテリジェンスのデータベース・ユーザーの作成」を参照してください。

- -job

  ワークロード・インテリジェンスのジョブを一意に識別する名前を指定します。

- -top

  レポートに示すパターン数を示す数値を指定します。パターンは、様々な基準で(実行回数、データベース時間および長さ)でソートされており、定義した上位の結果数のみが表示されます。これは任意指定の値ですが、デフォルト値は10です。

- -out

  レポートが格納されるファイルの名前(HTML形式)を指定します。この値の指定は任意ですが、デフォルト値には、-

jobオプションで指定したジョブ名が使用されます。

- -version

  GenerateReportプログラムのバージョン情報を示します。

- -usage

  GenerateReportプログラムのコマンドライン・オプション情報を示します。

例14-5に、GenerateReportプログラムを使用し、ワークロード・インテリジェンスのレポートを生成する方法を示します。

例14-5 ワークロード・インテリジェンス・レポートの生成

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.GenerateReport -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr -top 5 -out wijobsales.html
```

例: ワークロード・インテリジェンスの結果

この項では、ワークロード・インテリジェンスがSwingbench(Oracle Databaseのストレス・テストに使用されるベンチマーク)で生成された取得ワークロードに対して使用されているシナリオを想定しています。

ワークロード・インテリジェンスで検出された最も有意なパターンは、次の6つのテンプレートのものです。

```
SELECT product_id, product_name, product_description, category_id, weight_class,
       supplier_id, product_status, list_price, min_price, catalog_url
  FROM product_information
 WHERE product_id = :1;

SELECT p.product_id, product_name, product_description, category_id, weight_class,
       supplier_id, product_status, list_price, min_price, catalog_url,
       quantity_on_hand, warehouse_id
  FROM product_information p, inventories i
 WHERE i.product_id = :1 and i.product_id = p.product_id;

INSERT INTO order_items (order_id, line_item_id, product_id, unit_price, quantity)
     VALUES (:1, :2, :3, :4, :5);

UPDATE orders
   SET order_mode = :1, order_status = :2, order_total = :3
 WHERE order_id = :4;

SELECT /*+ use_nl */ o.order_id, line_item_id, product_id, unit_price, quantity,
       order_mode, order_status, order_total, sales_rep_id, promotion_id,
       c.customer_id, cust_first_name, cust_last_name, credit_limit, cust_email
  FROM orders o, order_items oi, customers c
 WHERE o.order_id = oi.order_id
   AND o.customer_id = c.customer_id
   AND o.order_id = :1;

UPDATE inventories
   SET quantity_on_hand = quantity_on_hand - :1
 WHERE product_id = :2
   AND warehouse_id = :3;
```

このパターンは、製品の注文でよく行われるユーザー・アクションに対応します。この例で特定されたパターンは222,261回(または実行の合計数の約8.21%)実行され、DB時間は58,533.70秒(または合計DB時間の約11.21%)消費されました。

この例で、ワークロード・インテリジェンスでは、次の4つのテンプレートの別の有意なパターンも検出しています。

```
SELECT customer_seq.nextval
  FROM dual;

INSERT INTO customers (customer_id, cust_first_name, cust_last_name, nls_language,
                       nls_territory, credit_limit, cust_email, account_mgr_id)
     VALUES (:1, :2, :3, :4, :5, :6, :7, :8);

INSERT INTO logon
     VALUES (:1, :2);

SELECT customer_id, cust_first_name, cust_last_name, nls_language, nls_territory,
       credit_limit, cust_email, account_mgr_id
  FROM customers
 WHERE customer_id = :1;
```

このパターンは、新しい顧客アカウントの作成とその後のシステムへのログインに対応します。この例で特定されたパターンは90,699回(または実行の合計数の約3.35%)実行され、DB時間は17,484.97秒(または合計DB時間の約3.35%)消費されました。

15 データベース統合リプレイの使用

データベース・リプレイでは、本番システムのワークロードを取得し、テスト・システムでそれをリプレイすることができます。これは、本番システムに変更の影響を与えずにテストできるので、新しいデータベース・テクノロジを評価したり採用するときに非常に便利です。ただし、テスト対象の新しいシステムのパフォーマンスが既存のシステムよりも大幅に高い場合、データベース・リプレイでは、新しいシステムでさらにどの程度ワークロードを追加処理できるか正確に予測できない場合があります。

たとえば、複数の本番システムを1つのOracle Exadataマシンに統合する場合、1つの既存のシステムから取得したワークロードをOracle Exadataマシンでリプレイすると、新しいシステムの方が強力であるため、リソースの使用率(ホストCPUやI/Oなど)が低くなる場合があります。このような場合には、1つのシステムの1つのワークロードではなく、すべての既存のシステムのワークロードを統合したものを新しいシステムがどのように処理するか評価するのが有用です。

データベース統合リプレイでは、1つ以上のシステムから取得した複数のワークロードを統合し、1つのテスト・システムで同時にリプレイできます。この例では、データベース統合リプレイを使用し、データベースの統合がどのように本番システムに影響を与えるか、また1つのOracle Exadataマシンで統合データベースのワークロードを処理できるか評価できます。

この章では、次の内容で、データベース統合リプレイの使用方法について説明します。

データベース統合リプレイの使用事例

データベース統合リプレイでは、1つ以上のシステムから取得した複数のワークロードを同時にリプレイできます。統合リプレイでリプレイが開始されると、統合されたすべてのワークロード取得のリプレイが開始されます。

データベース統合リプレイのいくつかの典型的な使用事例は次のとおりです。

- プラガブル・データベースを使用したデータベースの統合
- ストレス・テスト
- スケールアップ・テスト

これらのいずれの使用事例も、この章で説明する手順を使用して実行できます。さらに、データベース統合リプレイの使用時には、様々なワークロードのスケールアップ・テクニックを採用できます。詳細は、「ワークロード・スケールアップの使用」を参照してください。

プラガブル・データベースを使用したデータベースの統合

データベース統合リプレイの1つの用途は、統合したデータベースのワークロードをシステムが処理できるか評価することです。

たとえば、CRM、ERPおよびSCMアプリケーションのデータベースをプラガブル・データベース(PDB)に移行してデータベースを統合するとします。データベース統合リプレイを使用すると、3つのアプリケーションから取得したワークロードを統合し、PDBで同時にリプレイできます。

関連項目:

この使用事例については、「例: APIを使用した統合済ワークロードのリプレイ」を参照してください

ストレス・テスト

データベース統合リプレイは、ストレス・テストまたはキャパシティ・プランニングにも使用できます。

たとえば、連休の時期に営業アプリケーションのワークロードが倍になると予想されるとします。この場合、データベース統合リプレイを使用し、ワークロードを倍にし統合したワークロードをリプレイして、システムに対する追加の負荷をテストできます。

関連項目:

この使用事例については、「タイム・シフトの使用」を参照してください

スケールアップ・テスト

データベース統合リプレイの3つ目の用途は、スケールアップ・テストです。

たとえば、財務のアプリケーションと注文のアプリケーションの取得済ワークロードをシステムで同時に処理できるかテストしたいとします。この場合、データベース統合リプレイを使用し、ワークロードを統合しそれらを同時にリプレイして、スケールアップしたワークロードのシステムに対する追加の負荷の影響をテストできます。

関連項目:

- 「スキーマの再マッピングの使用」
- 「ワークロードの縮小の使用」

データベース統合リプレイの使用ステップ

この項では、統合済ワークロード・リプレイを使用する場合に含まれるステップを説明します。次の項目が含まれます。

- データベース統合リプレイ用のデータベース・ワークロードの取得
- データベース統合リプレイ用のテスト・システムの設定
- データベース統合リプレイ用のデータベース・ワークロードの前処理
- データベース統合リプレイ用のデータベース・ワークロードのリプレイ
- データベース統合リプレイのレポート作成および分析

データベース統合リプレイ用のデータベース・ワークロードの取得

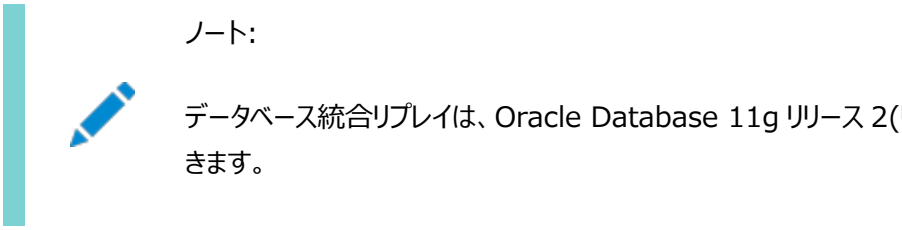
データベース統合リプレイ用にデータベース・ワークロードを取得するには、特別なステップは必要ありません。データベース・ワークロードを取得するステップは、データベース・リプレイで単一のワークロードを取得するステップとまったく同じです。詳細は、「データベース・ワークロードの取得」を参照してください。

この項では、データベース統合リプレイに固有なワークロードの取得に関して、次の内容で説明します。

- サポートされているタイプのワークロード取得
- 取得サブセット

サポートされているタイプのワークロード取得

データベース統合リプレイでは、1つ以上のオペレーティング・システム上のOracle Database 9iリリース2(リリース9.2.0.8.0)以上で実行される、1つ以上のシステムから取得された複数のワークロードがサポートされています。たとえば、HP-UX上で実行されるOracle Database 9iリリース2(リリース9.2.0.8.0)を実行するシステムから取得したワークロードと、AIX上で実行されるOracle Database 10gリリース2(リリース10.2.0.4.0)を実行する別のシステムから取得したワークロードを使用できます。



ノート:

データベース統合リプレイは、Oracle Database 11g リリース 2(リリース 11.2.0.2.0)以上のみで使用できます。

取得サブセット

データベース統合リプレイでは、取得した既存のワークロードを、新しい小さな取得サブセットに変換することができます。その後、「データベース統合リプレイの使用事例」で説明されているように、別の使用事例で使用できる取得サブセットから、新しいワークロード取得を生成できます。

取得サブセットとは、時間範囲を適用して既存の1つのワークロード取得から定義された1つのワークロード取得の部分です。時間範囲は、ワークロード取得の開始からのオブセットとして指定します。指定した時間範囲に取得されたすべてのユーザー・ワークロードは、定義した取得サブセットに含まれます。

たとえば、午前2時から午後8時までのワークロードを取得し、ワークロードのピークが午前10時から午後4時だと特定されたとします。このとき、ワークロードの開始から8時間後(または午前10時)に開始し、ワークロードの開始から14時間後(または午後4時)に終了する時間範囲をワークロードのピークとして適用して、取得サブセットを定義できます。

ただし、指定した時間範囲に記録されたユーザー・ワークロードのみを取得サブセットに含める場合、指定した時間範囲よりも前に発生したユーザー・ログインは記録されないことになります。リプレイでこれらのユーザー・ログインが必要な場合、その取得サブセットはリプレイできません。たとえば、取得サブセットに指定した時間範囲が午前10時00分から午後4時であり、ユーザー・セッションが午前9時30分に開始され午前10時30分に終了した場合、午前9時30分のユーザー・ログインがワークロードに含まれないと、リプレイは失敗します。また、ユーザー・セッションが午後3時30分に開始し午後4時30分までに完了しない場合、指定した時間範囲には一部のみしか記録されず、ユーザー・コールが不完全になります。

データベース統合リプレイでは、指定した時間範囲の開始時間のみが原因でユーザー・コールが不完全になる場合、これを含めるようにしてこの問題を解決しています。ワークロード取得が縮小されている場合、同じ不完全なユーザー・コールが2度含ま

れるのを避けるために、終了時間が原因でユーザー・コールが不完全となるものはデフォルトでは含めないようになっています。したがって、基本的に取得サブセットとは、正常なリプレイに必要な、指定した時間範囲に記録されたユーザー・コールの最少数で、必要なユーザー・ログイン、ALTER SESSION文、および開始時間が原因で不完全となってしまうユーザー・コールはこれに含まれます。

関連項目:

「APIを使用した取得サブセットの生成」

データベース統合リプレイ用のテスト・システムの設定

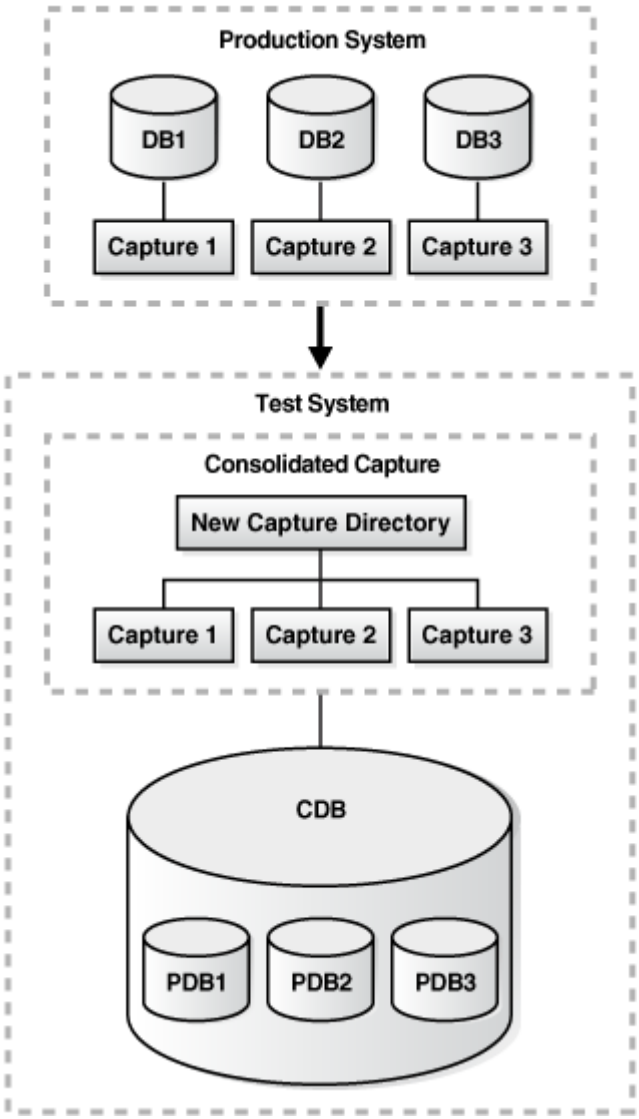
データベース統合リプレイ用のテスト・システムの設定は、データベース・リプレイ用のテスト・システムの設定と同様です。ただし、データベース統合リプレイ用にリプレイ・データベースを設定する場合には、他に考慮する必要のある事項があります。データベース・リプレイ用のテスト・システムの設定の詳細は、「データベース・ワークロードのリプレイのステップ」を参照してください。

リプレイ中の相違を最小限に抑えるには、テスト・システムには同じアプリケーション・データが必要であり、アプリケーション・データの状態は、ワークロードをそれぞれ取得開始したときの取得システムのデータの状態と論理的に同じにする必要があります。ただし、取得が統合されている場合、様々な本番システムのワークロード取得が複数含まれている可能性があるため、テスト・システムをすべての取得に対応するよう設定する必要があります。この場合、各データベースが取得の開始時に取得システムと同等のデータを持つように、マルチテナント・アーキテクチャを使用して複数データベースを統合することをお薦めします。

データベース統合リプレイを行うには、関係しているすべてのワークロード取得を、テスト・システムの新しい取得ディレクトリの下に配置する必要があります。ワークロード取得をすべて新しい取得ディレクトリにコピーするか、元のワークロード取得をポイントするシンボリック・リンクを作成します。ワークロード取得を統合する前に、新しい取得ディレクトリに関係するすべての取得を格納する十分な空き領域があることを確認します。

図15-1に、3つのワークロード取得を統合するテスト・システムと新しい取得ディレクトリを設定する方法を示します。

図15-1 データベース統合リプレイ用のテスト・システムの設定



関連項目:

- 「APIを使用した統合リプレイ・ディレクトリの設定」
- マルチテナント・アーキテクチャの詳細は、『Oracle Database概要』を参照してください。

データベース統合リプレイ用のデータベース・ワークロードの前処理

データベース統合リプレイ用のデータベース・ワークロードの事前処理は、データベース・リプレイ用のデータベース・ワークロードの事前処理と同様です。データベース・リプレイ用のデータベース・ワークロードの事前処理の詳細は、「データベース・ワークロードの事前処理」を参照してください。

データベース統合リプレイを行うには、取得した各ワークロードをその固有のディレクトリに前処理します。前処理を行う際には、別のワークロード取得を1つのディレクトリにまとめないようにする必要があります。取得済ワークロードの前処理は、ワークロードをリプレイするテスト・システムのOracle Databaseと同じバージョンのデータベースを使用して実行する必要があります。

データベース統合リプレイ用のデータベース・ワークロードのリプレイ

データベース統合リプレイで統合されたワークロードのリプレイは、データベース・リプレイで1つのデータベース・ワークロードをリプレイすることとは大きく異なります。

この項では、データベース統合リプレイに固有なワークロードのリプレイに関して、次の内容で説明します。

- リプレイ・スケジュールの定義
- データベース統合リプレイ用の接続の再マッピング
- データベース統合リプレイ用のユーザーの再マッピング
- データベース統合リプレイの準備
- 個々のワークロードのリプレイ

リプレイ・スケジュールの定義

リプレイ・スケジュールでは、1つまたは複数のワークロード取得を統合リプレイに追加し、取得のリプレイ開始順序を指定します。リプレイ・スケジュールは、統合リプレイを初期化する前に作成する必要があります。1回の統合リプレイには、複数のリプレイ・スケジュールを定義できます。リプレイの初期化時、既存のリプレイ・スケジュールの中から任意のものを選択できます。

リプレイ・スケジュールでは、2種類の操作が実行されます。

- ワークロード取得の追加
- スケジュールの順序の追加

関連項目:

「APIを使用したリプレイ・スケジュールの定義」

ワークロード取得の追加

リプレイ・スケジュールでは、最初に関係するワークロード取得がリプレイに追加されます。

リプレイ・スケジュールにワークロード取得が追加されると、そのワークロード取得を識別する一意の番号が返されます。ワークロード取得には、追加のたびに別の番号が割り当てられるので、複数回リプレイ・スケジュールに追加することが可能です。リプレイ・スケジュールでは、追加のたびに取得をコピーしてディスク領域を無駄にしないために、都度同じ取得ディレクトリをポイントします。

関連項目:

「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」

スケジュールの順序の追加

次いで、リプレイ・スケジュールには、関係するワークロード取得のリプレイ時開始順序が追加されます。

スケジュール順序では、リプレイ・スケジュールに追加された2つのワークロード取得の開始順序が定義されます。リプレイ・スケジュールには、複数のスケジュール順序を追加できます。たとえば、リプレイ・スケジュールにワークロード取得が3つ追加されているとします。追加した1つのスケジュール順序では、取得2は取得1が完了してから開始するよう指定します。別のスケジュール順序では、取得3は取得1が完了してから開始するよう指定します。この場合、取得2も取得3も取得1が完了するのを待機してから開始する必要があります。

リプレイ・スケジュールには、必ずしもスケジュール順序を含める必要はありません。この場合、リプレイ・スケジュール内の関係す

るすべてのワークロード取得は、統合リプレイの開始時に同時にリプレイが開始されます。

関連項目:

「APIを使用したリプレイ・スケジュールへのスケジュール順序の追加」

データベース統合リプレイ用の接続の再マッピング

データベース・リプレイを使用して1つのデータベース・ワークロードをリプレイする場合と同様に、本番システムに接続するために使用した取得した接続文字列を使用してリプレイ・システムに再マッピングする必要があります。詳細は、「接続の再マッピング」を参照してください。

データベース統合リプレイを行うには、複数のワークロード取得の取得済接続文字列をリプレイ時に別の接続文字列に再マッピングする必要があります。

関連項目:

「APIを使用した接続の再マッピング」

データベース統合リプレイ用のユーザーの再マッピング

データベース・リプレイを使用して1つのデータベース・ワークロードをリプレイする場合と同様に、本番システムに接続するために使用するデータベースのユーザー名およびスキーマはリプレイ時に再マッピングできます。詳細は、「ユーザーの再マッピング」を参照してください。

データベース統合リプレイの場合には、リプレイ時に、複数のワークロード取得からの取得済ユーザーを別のユーザーまたはスキーマに再マッピングするよう選択できます。

関連項目:

「APIを使用したユーザーの再マッピング」

データベース統合リプレイの準備

データベース・リプレイを使用した1つのデータベース・ワークロードのリプレイの場合と同様に、リプレイ・オプションはリプレイの準備時に定義します。詳細は、「リプレイ・オプションの指定」を参照してください。

データベース統合リプレイでは、統合リプレイ内のすべての関係しているワークロード取得は、リプレイの準備時に定義された同じリプレイ・オプションを使用してリプレイされます。

関連項目:

「APIを使用したデータベース統合リプレイの準備」

個々のワークロードのリプレイ

統合ワークロードをリプレイする前に、関係するワークロードを個々にリプレイすることをお薦めします。詳細は、「データベース・ワークロードのリプレイ」を参照してください。

個々にリプレイすることによって各ワークロード取得のベースライン・パフォーマンスを定めて、統合リプレイのパフォーマンスの分析に使用できます。

データベース統合リプレイのレポート作成および分析

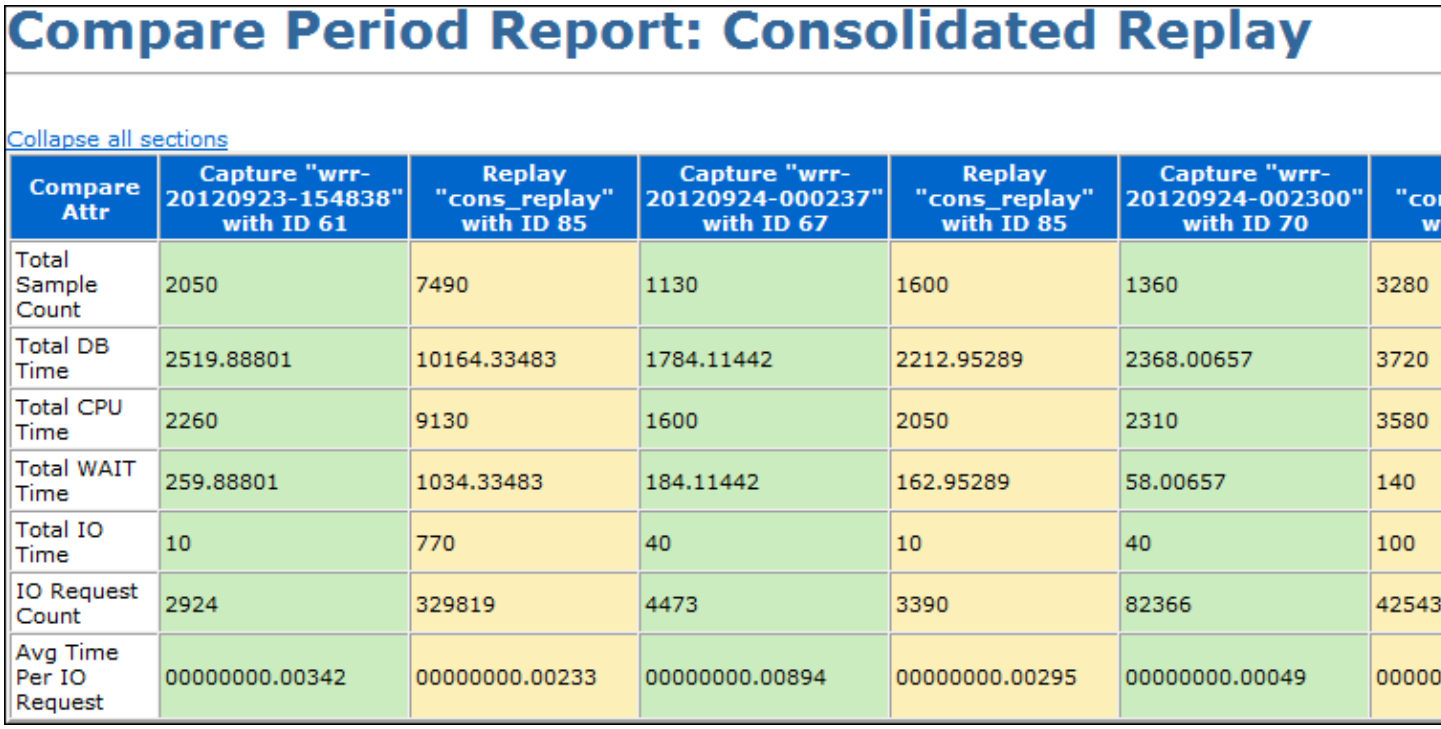
データベース統合リプレイのレポート作成および分析は、リプレイの期間比較レポートを使用して実行されます。詳細は、「リプレイの期間比較レポートの使用」を参照してください。

データベース統合リプレイのリプレイの期間比較レポートには、個々のワークロード取得のアクティブ・セッション履歴(ASH)データがあり、ワークロード取得のASHデータが統合リプレイのフィルタされたASHデータと比較されています。このレポートを使用すると、同じ統合されているワークロード取得のリプレイを比較できます。

データベース統合リプレイのリプレイの期間比較レポートでは、統合リプレイを複数の取得とリプレイの比較として扱います。このレポートのサマリー・セクションには、取得とリプレイの比較を個々にまとめた表があります。このセクションの情報を確認すると、統合リプレイがどのように実行されたかの概要を理解できます。

図15-2に、データベース統合リプレイのサンプル・リプレイ期間比較レポートのサマリー・セクションを示します。

図15-2 期間比較レポート: 統合リプレイ



Compare Period Report: Consolidated Replay

Collapse all sections

Compare Attr	Capture "wrr-20120923-154838" with ID 61	Replay "cons_replay" with ID 85	Capture "wrr-20120924-000237" with ID 67	Replay "cons_replay" with ID 85	Capture "wrr-20120924-002300" with ID 70	"co w
Total Sample Count	2050	7490	1130	1600	1360	3280
Total DB Time	2519.88801	10164.33483	1784.11442	2212.95289	2368.00657	3720
Total CPU Time	2260	9130	1600	2050	2310	3580
Total WAIT Time	259.88801	1034.33483	184.11442	162.95289	58.00657	140
Total IO Time	10	770	40	10	40	100
IO Request Count	2924	329819	4473	3390	82366	42543
Avg Time Per IO Request	00000000.00342	00000000.00233	00000000.00894	00000000.00295	00000000.00049	00000

このレポートの残りのセクションは、リプレイの期間比較レポートのASHデータ比較セクションと似ており、統合リプレイ内のすべての取得とリプレイのレポートを結合したもので構成されています。このセクションの詳細は、「ASHデータ比較」を参照してください。

Enterprise Managerを使用したデータベース統合リプレイの使用

この項では、Enterprise Managerを使用してデータベース統合リプレイを使用する方法について説明します。

Oracle Enterprise Managerは、統合されたデータベース・ワークロードをリプレイする主要ツールです。Oracle Enterprise Managerを使用できない場合は、「APIを使用したデータベース統合リプレイの使用」で説明されているように、APIを使用して統合されたデータベース・ワークロードをリプレイすることもできます。

統合されたデータベース・ワークロードをリプレイするプロセスは、単一のデータベース・ワークロードをリプレイするプロセスとほとんど同じです。違いは、次の項の単一のリプレイの手順で説明されています。

- 「データベース・ワークロードの事前処理」の「データベースのリプレイ・タスクの作成」
- 「データベース・ワークロードの事前処理」の「ワークロードの事前処理とリプレイ・クライアントのデプロイ」
- 「データベース・ワークロードのリプレイ」の「Enterprise Managerを使用したデータベース・ワークロードのリプレイ」

次のリストは、統合されたデータベース・ワークロードのリプレイと単一のデータベース・ワークロードのリプレイの違いのサマリーを示します。

- リプレイ・タスクを作成する際、2つ以上の取得済ワークロードをタスクの作成ページの取得の選択表から選択する必要があります。
- ウィザードの「取得されたワークロードの前処理: ワークロードのコピー」ステップには、「取得名」ドロップダウンの複数の選択肢があるため、ワークロード・ディレクトリの現在の場所の複数の資格証明を入力する必要がある場合があります。
- ウィザードの「取得されたワークロードの前処理: ディレクトリを選択」ステップでは、単一のリプレイで表示されるように「取得サマリー」が表示されません。
- ウィザードの「ワークロード・リプレイ: ワークロードのコピー」ステップには、「取得名」ドロップダウンの複数の選択肢があるため、ワークロード・ディレクトリの現在の場所の複数の資格証明を入力する必要がある場合があります。
- ウィザードの「ワークロード・リプレイ: ディレクトリを選択」ステップでは、単一のリプレイで表示されるように「取得サマリー」が表示されません。
- ウィザードの「ワークロード・リプレイ: 初期化オプション」ステップでは、「ソースを指定」セクションは表示されません。
- ウィザードの「ワークロード・リプレイ: オプションのカスタマイズ」ステップの「接続マッピング」の「取得名」ドロップダウンには、1つ以上の選択肢が表示されるので、取得したワークロードの接続をそれぞれ再マッピングできます。1つの接続記述子またはネット・サービス名を使用することはできません。

APIを使用したデータベース統合リプレイの使用

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して統合済ワークロードを作成およびリプレイする方法を説明します。「Enterprise Managerを使用したデータベース統合リプレイの使用」で説明されているように、Oracle Enterprise Managerを使用して統合されているワークロードを作成およびリプレイすることもできます。

APIを使用して統合されているワークロードを作成およびリプレイするステップは、次などの複数のステップを実行する必要があります。

- APIを使用した取得サブセットの生成
- APIを使用した統合リプレイ・ディレクトリの設定

- APIを使用したリプレイ・スケジュールの定義
- APIを使用したデータベース統合リプレイの実行

関連項目:

DBMS_WORKLOAD_REPLAYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用した取得サブセットの生成

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、既存のワークロード取得から取得サブセットを生成する方法を説明します。取得サブセットの詳細は、「取得サブセット」を参照してください。

既存のワークロード取得から取得サブセットを生成するには:

1. GENERATE_CAPTURE_SUBSETプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET (
   input_capture_dir         IN VARCHAR2,
   output_capture_dir        IN VARCHAR2,
   new_capture_name          IN VARCHAR2,
   begin_time                IN NUMBER,
   begin_include_incomplete  IN BOOLEAN   DEFAULT TRUE,
   end_time                  IN NUMBER,
   end_include_incomplete    IN BOOLEAN   DEFAULT FALSE,
   parallel_level            IN NUMBER    DEFAULT NULL);
```

2. input_capture_dirパラメータを、既存のワークロード取得をポイントするディレクトリ・オブジェクト名に設定します。
3. output_capture_dirパラメータを、新しいワークロード取得を格納する空のディレクトリをポイントするディレクトリ・オブジェクト名に設定します。
4. new_capture_nameパラメータを、生成する新しいワークロード取得の名前に設定します。
5. 任意指定の他のパラメータを適宜設定します。

   これらのパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

この例では、ディレクトリ・オブジェクトrec_dirの既存のワークロード取得からpeak_wkldという取得サブセットをディレクトリ・オブジェクトpeak_capdirに作成する方法を説明します。取得サブセットには、ワークロードの取得の開始から2時間(または7,200秒)から3時間(または10,800秒)のワークロードが含まれています。

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('rec_dir', 'peak_capdir',
     'peak_wkld', 7200, TRUE, 10800, FALSE, 1);
```

関連項目:

GENERATE_CAPTURE_SUBSETプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用した統合リプレイ・ディレクトリの設定

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、テスト・システムに統合リプレイ・ディレクトリを設定する方法を説明します。統合リプレイ・ディレクトリを、統合およびリプレイするワークロード取得を含むテスト・システムのディレクトリに設定します。テスト・システムの設定の詳細は、「データベース統合リプレイ用のテスト・システムの設定」を参照してください。

リプレイ・ディレクトリを設定するには:

1. SET_CONSOLIDATED_DIRECTORYプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.SET_CONSOLIDATED_DIRECTORY (
   replay_dir  IN VARCHAR2);
```

2. replay_dirパラメータをワークロードの統合に使用するワークロード取得を含むオペレーティング・システム・ディレクトリをポイントするディレクトリ・オブジェクト名に設定します。

ヒント:

SET_REPLAY_DIRECTORY プロシージャは非推奨で、SET_CONSOLIDATED_DIRECTORY プロシージャに置き換えられます。

この例では、リプレイ・ディレクトリをrep_dirという名前のディレクトリ・オブジェクトに設定する方法を示します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_CONSOLIDATED_DIRECTORY ('rep_dir');
```

また、DBMS_WORKLOAD_REPLAYパッケージを使用してSET_CONSOLIDATED_DIRECTORYプロシージャによって設定された現在の統合リプレイ・ディレクトリを参照することもできます。

設定されている現在の統合リプレイ・ディレクトリを確認するには:

- GET_REPLAY_DIRECTORYファンクションを使用します。

```
DBMS_WORKLOAD_REPLAY.GET_REPLAY_DIRECTORY RETURN VARCHAR2;
```

  統合リプレイ・ディレクトリが設定されていない場合、ファンクションでNULLが返されます。

関連項目:

- SET_REPLAY_DIRECTORYプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください
- GET_REPLAY_DIRECTORYファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用したリプレイ・スケジュールの定義

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用してリプレイ・スケジュールを定義する方法を説明します。リプレイ・スケジュールの詳細は、「リプレイ・スケジュールの定義」を参照してください。

リプレイ・スケジュールを定義する前に、次の前提条件が満たされていることを確認します。

- すべてのワークロード取得は、「データベース・ワークロードの事前処理」で説明しているように、リプレイ・システムと同じデータベース・バージョンを実行しているシステムで、PROCESS_CAPTUREプロシージャを使用して前処理されています。
- すべての取得ディレクトリは、リプレイ・システムのリプレイ・ディレクトリにコピー済です。
- 「APIを使用した統合リプレイ・ディレクトリの設定」で説明されているように、リプレイ・ディレクトリがSET_REPLAY_DIRECTORYプロシージャを使用して設定済です。
- データベースが、リプレイ・モードの状態ではありません。

リプレイ・スケジュールを定義するには:

1. 「APIを使用したリプレイ・スケジュールの作成」で説明されているように、新しいリプレイ・スケジュールを作成します。
2. 「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」で説明されているように、リプレイ・スケジュールにワークロード取得を追加します。
3. 「APIを使用したリプレイ・スケジュールへのスケジュール順序の追加」で説明されているように、リプレイ・スケジュールにスケジュール順序を追加します。
4. 「APIを使用したリプレイ・スケジュールの保存」で説明されているように、リプレイ・スケジュールを保存します。

APIを使用したリプレイ・スケジュールの作成

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用してリプレイ・スケジュールを作成する方法を説明します。リプレイ・スケジュールの詳細は、「リプレイ・スケジュールの定義」を参照してください。

リプレイ・スケジュールを作成するには:

1. BEGIN_REPLAY_SCHEDULEプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE (
   schedule_name   IN VARCHAR2);
```

2. schedule_nameパラメータをこのリプレイ・スケジュールの名前に設定します。

> ノート:
>
> BEGIN_REPLAY_SCHEDULE プロシージャでは、再利用可能なリプレイ・スケジュールの作成を開始します。リプレイ・スケジュールは 1 度に 1 つのみ定義できます。リプレイ・スケジュールの定義中にこのプロシージャを呼び出すと、エラーが発生します。

この例に、peak_scheduleというリプレイ・スケジュールを作成する方法を説明します。

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('peak_schedule');
```

関連項目:

BEGIN_REPLAY_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用したリプレイ・スケジュールへのワークロード取得の追加

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、リプレイ・スケジュールにワークロード取得を追加および削除する方法を説明します。リプレイ・スケジュールにワークロード取得を追加する方法の詳細は、「ワークロード取得の追加」を参照してください。

リプレイ・スケジュールにワークロード取得を追加する前に、次の前提条件が満たされていることを確認します。

- ワークロード取得を追加するリプレイ・スケジュールが作成済です。

  リプレイ・スケジュールの作成の詳細は、「APIを使用したリプレイ・スケジュールの作成」を参照してください。

リプレイ・スケジュールにワークロード取得を追加するには:

1. ADD_CAPTUREファンクションを使用します。

```
DBMS_WORKLOAD_REPLAY.ADD_CAPTURE (
   capture_dir_name     IN VARCHAR2,
   start_delay_seconds  IN NUMBER  DEFAULT 0,
   stop_replay          IN BOOLEAN DEFAULT FALSE,
   take_begin_snapshot  IN BOOLEAN DEFAULT FALSE,
   take_end_snapshot    IN BOOLEAN DEFAULT FALSE,
   query_only           IN BOOLEAN DEFAULT FALSE)
RETURN NUMBER;

DBMS_WORKLOAD_REPLAY.ADD_CAPTURE (
   capture_dir_name     IN VARCHAR2,
   start_delay_seconds  IN NUMBER,
   stop_replay          IN VARCHAR2,
   take_begin_snapshot  IN VARCHAR2 DEFAULT 'N',
   take_end_snapshot    IN VARCHAR2 DEFAULT 'N',
   query_only           IN VARCHAR2 DEFAULT 'N')
RETURN NUMBER;
```

   このファンクションでは、このリプレイ・スケジュール内でワークロード取得を識別する一意の識別子が返されます。

   参照:

   問合せのみのデータベース・リプレイについては、「問合せのみのデータベース・リプレイについて」を参照してください。

   ノート:

   問合せのみのデータベース・リプレイはテスト環境でのみ使用され、実行されます。

   - 問合せのみのデータベース・リプレイを本番システムで使用しないでください。
   - 問合せのみのデータベース・リプレイ中に相違が起こる可能性があります。

2. capture_dir_nameパラメータを上位レベルのリプレイ・ディレクトリにある取得したワークロードをポイントするディレクトリ・オブジェクト名に設定します。

このディレクトリには、リプレイ・システムと同じデータベース・バージョンを実行しているシステムで前処理された有効なワークロード取得が含まれている必要があります。

3. 任意指定の他のパラメータを適宜設定します。

   これらのパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

次の例に、SELECT文でADD_CAPTUREファンクションを使用して、リプレイ・スケジュールにpeak_wkldというワークロード取得を追加する方法を示します。

```
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('peak_wkld')
  FROM dual;
```

リプレイ・スケジュールからワークロード取得を削除する場合、DBMS_WORKLOAD_REPLAYパッケージを使用することも可能です。

リプレイ・スケジュールからワークロード取得を削除するには:

1. REMOVE_CAPTUREプロシージャを使用します。

   ```
   DBMS_WORKLOAD_REPLAY.REMOVE_CAPTURE (
      schedule_capture_number IN NUMBER);
   ```

2. schedule_capture_numberパラメータを、このリプレイ・スケジュールのワークロード取得を識別する一意の識別子に設定します。

   この一意の識別子は、ワークロード取得がリプレイ・スケジュールに追加された際に、ADD_CAPTUREファンクションによって返された識別子と同じです。

関連項目:

- ADD_CAPTUREファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- REMOVE_CAPTUREプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

APIを使用したリプレイ・スケジュールへのスケジュール順序の追加

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、スケジュール順序をリプレイ・スケジュールから削除および追加する方法を説明します。リプレイ・スケジュールにスケジュール順序を追加する方法の詳細は、「スケジュール順序の追加」を参照してください。

リプレイ・スケジュールにスケジュール順序を追加する前に、次の前提条件が満たされていることを確認します。

- スケジュール順序を追加するリプレイ・スケジュールが作成済です。

  リプレイ・スケジュールの作成の詳細は、「APIを使用したリプレイ・スケジュールの作成」を参照してください。

- そのスケジュール順序に関係するすべてのワークロード取得がリプレイ・スケジュールに追加済です。

  リプレイ・スケジュールへのワークロード取得の追加の詳細は、「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」を参照してください。

ノート:

リプレイ・スケジュールにはスケジュール順序を必ずしも追加する必要はありません。リプレイ・スケジュールにスケジュール順序を追加しない場合、統合リプレイが開始されるとき、リプレイ・スケジュールに追加されたすべてのワークロード取得が同時にリプレイされます。

リプレイ・スケジュールにスケジュール順序を追加するには:

1. ADD_SCHEDULE_ORDERINGファンクションを使用します。

```
DBMS_WORKLOAD_REPLAY.ADD_SCHEDULE_ORDERING (
   schedule_capture_id IN NUMBER,
   waitfor_capture_id IN NUMBER)
RETURN NUMBER;
```

   このファンクションでは、リプレイ・スケジュールに追加された2つのワークロード取得間にスケジュール順序を追加します。スケジュール順序を追加できない場合、ゼロではないエラー・コードが返されます。

2. このスケジュール順序で待機するワークロード取得にschedule_capture_idパラメータを設定します。
3. このスケジュール順序で他のワークロード取得が開始される前に、完了するワークロード取得をwait_for_capture_idパラメータに設定します。

DBMS_WORKLOAD_REPLAYパッケージを使用して、リプレイ・スケジュールからスケジュール順序を削除することも可能です。

リプレイ・スケジュールからスケジュール順序を削除するには:

1. REMOVE_SCHEDULE_ORDERINGプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.REMOVE_SCHEDULE ORDERING (
   schedule_capture_id IN VARCHAR2,
   wait_for_capture_id IN VARCHAR2);
```

2. このスケジュール順序で待機しているワークロード取得にschedule_capture_idパラメータを追加します。
3. このスケジュール順序で他のワークロード取得が開始する前に完了する必要のあるワークロード取得にwait_for_capture_idパラメータを設定します。

スケジュール順序を確認するには:

- DBA_WORKLOAD_SCHEDULE_ORDERINGビューを使用します。

関連項目:

- ADD_SCHEDULE_ORDERINGファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- REMOVE_SCHEDULE_ORDERINGプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBA_WORKLOAD_SCHEDULE_ORDERINGビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

APIを使用したリプレイ・スケジュールの保存

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して定義されたリプレイ・スケジュールを保存する方法を説明します。

リプレイ・スケジュールを保存する前に、次の前提条件が満たされていることを確認します。

- 保存するリプレイ・スケジュールが作成済です。

  リプレイ・スケジュールの作成の詳細は、「APIを使用したリプレイ・スケジュールの作成」を参照してください。

- そのスケジュール順序に関係するすべてのワークロード取得がリプレイ・スケジュールに追加済です。

  リプレイ・スケジュールへのワークロード取得の追加の詳細は、「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」を参照してください。

- 使用するすべてのスケジュール順序がリプレイ・スケジュールに追加済です(このステップは任意指定です)。

  リプレイ・スケジュールへのスケジュール順序の追加の詳細は、「APIを使用したリプレイ・スケジュールへのスケジュール順序の追加」を参照してください。

リプレイ・スケジュールを保存するには:

- END_REPLAY_SCHEDULEプロシージャを使用します。

  ```
  DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
  ```

  この手順でリプレイ・スケジュールの作成が完了します。リプレイ・スケジュールがリプレイ・ディレクトリに関連付けられ保存されます。リプレイ・スケジュールは保存すると、統合リプレイ用に使用できます。

リプレイ・スケジュールを確認するには:

- DBA_WORKLOAD_REPLAY_SCHEDULESビューを使用します。

関連項目:

- END_REPLAY_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBA_WORKLOAD_REPLAY_SCHEDULESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

APIを使用したデータベース統合リプレイの実行

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、データベース統合リプレイを実行する方法を説明します。統合リプレイの詳細は、「データベース統合リプレイ用のデータベース・ワークロードのリプレイ」を参照してください。

データベース統合リプレイを実行する前に、次の前提条件が満たされていることを確認します。

- すべてのワークロード取得は、「データベース・ワークロードの事前処理」で説明しているように、リプレイ・システムと同じデータベース・バージョンを実行しているシステムで、PROCESS_CAPTUREプロシージャを使用して前処理されています。
- すべての取得ディレクトリは、リプレイ・システムのリプレイ・ディレクトリにコピー済です。
- 「APIを使用した統合リプレイ・ディレクトリの設定」で説明されているように、リプレイ・ディレクトリがSET_REPLAY_DIRECTORYプロシージャを使用して設定済です。
- データベースは、すべてのワークロード取得の開始時間にすべての取得システムと同じアプリケーション状態に論理的に

復元されています。

データベース統合リプレイを実行するには:

1. 「APIを使用したデータベース統合リプレイの初期化」で説明されているように、リプレイ・データを初期化します。
2. 「APIを使用した接続の再マッピング」で説明されているように、接続文字列を再マッピングします。
3. 「APIを使用したユーザーの再マッピング」で説明されているように、ユーザーを再マッピングします。

   ユーザーの再マッピングは任意です。
4. 「APIを使用したデータベース統合リプレイの準備」で説明されているように、統合リプレイを準備します。
5. 「リプレイ・クライアントの設定」で説明されているように、リプレイ・クライアントを設定して開始します。
6. 「APIを使用したデータベース統合リプレイの開始」で説明されているように、統合リプレイを開始します。
7. 「データベース統合リプレイのレポート作成および分析」で説明されているように、レポート作成および分析を行います。

APIを使用したデータベース統合リプレイの初期化

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、統合リプレイ用にリプレイ・データを初期化する方法を説明します。

リプレイ・データを初期化すると次の操作が実行されます。

- データベースの状態が統合ワークロードのリプレイ用に初期化モードになります。
- リプレイ・スケジュールに関係するすべてのワークロード取得を含むリプレイ・ディレクトリがポイントされます。
- リプレイに必要なメタデータが表にロードされます。

  たとえば、取得した接続文字列が、リプレイ用に再マッピング可能な表にロードされます。

データベース統合リプレイを初期化するには:

1. INITIALIZE_CONSOLIDATED_REPLAYプロシージャを使用します:

```
DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
   replay_name     IN VARCHAR2,
   schedule_name   IN VARCHAR2,
   plsql_mode      IN VARCHAR2 DEFAULT 'TOP_LEVEL');
```

2. replay_nameパラメータを統合リプレイの名前に設定します。
3. schedule_nameパラメータを使用するリプレイ・スケジュールの名前に設定します。

   schedule_nameパラメータは、「APIを使用したリプレイ・スケジュールの作成」で説明されているように、その作成時に使用されたリプレイ・スケジュールの名前です。

オプションのplsql_modeパラメータで、PL/SQLのリプレイ・モードを指定します。

plsql_modeパラメータには次の2つの値を設定できます。

- top_level: 最上位レベルのPL/SQLコールのみ。これがデフォルト値です。
- extended: PL/SQL内で実行されたSQL、またはPL/SQL内に記録されているSQLがない場合は最上位レベルのPL/SQL。すべての取得は、'extended' PL/SQLモードで実行されている必要があります。PL/SQL以外のコールは通常の方法でリプレイされます。

次の例に、peak_scheduleという名前のリプレイ・スケジュールを使用して、peak_replayという名前の統合リプレイを初期化す

る方法を説明します。

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('peak_replay',
     'peak_schedule');
```

関連項目:

INITIALIZE_CONSOLIDATED_REPLAYプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用した接続の再マッピング

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、統合リプレイ用の接続文字列を再マッピングする方法を説明します。接続の再マッピングの詳細は、「データベース統合リプレイ用の接続の再マッピング」を参照してください。

接続文字列を再マッピングするには:

1. REMAP_CONNECTIONプロシージャを使用します。

   ```
   DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (
      schedule_cap_id   IN NUMBER,
      connection_id     IN NUMBER,
      replay_connection IN VARCHAR2);
   ```

   この手順では、リプレイ・スケジュール内のすべての関係するワークロード取得の取得済の接続を新しい接続文字列に再マッピングします。

2. schedule_capture_idパラメータを、現在のリプレイ・スケジュールに関係するワークロード取得に設定します。

   schedule_capture_idパラメータは、「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」で説明されているように、ワークロード取得をリプレイ・スケジュールに追加した際に返された一意の識別子です。

3. connection_idパラメータを再マッピングする接続に設定します。

   connection_idパラメータは、リプレイ・データの初期化時に生成され、ワークロード取得からの接続に対応しています。

4. replay_connectionパラメータを、リプレイ時に使用される新しい接続文字列に設定します。

関連項目:

REMAP_CONNECTIONプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用したユーザーの再マッピング

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、統合リプレイのためにユーザーを再マッピングする方法を説明します。ユーザーの再マッピングの詳細は、「データベース統合リプレイ用のユーザーの再マッピング」を参照してください。

ユーザーを再マッピングする前に、次の前提条件が満たされていることを確認します。

- 「APIを使用したデータベース統合リプレイの初期化」で説明されているように、リプレイ・データが初期化済です。

- データベースが、リプレイ・モードの状態ではありません。

ユーザーを再マッピングするには:

1. SET_USER_MAPPINGプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (
   schedule_cap_id IN NUMBER,
   capture_user    IN VARCHAR2,
   replay_user     IN VARCHAR2);
```

2. schedule_capture_idパラメータを、現在のリプレイ・スケジュールに関係するワークロード取得に設定します。

   schedule_capture_idパラメータは、「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」で説明されているように、ワークロード取得をリプレイ・スケジュールに追加した際に返された一意の識別子です。

3. capture_userパラメータを、ワークロードの取得時に取得したユーザーまたはスキーマのユーザー名に設定します。

4. replay_userパラメータを、リプレイ時に取得済ユーザーが再マッピングされている新しいユーザーまたはスキーマのユーザー名に設定します。

   パラメータがNULLに設定されている場合、マッピングは無効です。

この例では、1001のワークロード取得のリプレイ中に、取得中に使用されたPRODユーザーをTESTユーザーに再マッピングする方法を示します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (1001, 'PROD', 'TEST');
```

関連項目:

SET_USER_MAPPINGプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

APIを使用したデータベース統合リプレイの準備

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して統合リプレイを準備する方法を説明します。統合リプレイの準備の詳細は、「データベース統合リプレイの準備」を参照してください。

統合リプレイを準備する前に、次の前提条件が満たされていることを確認します。

- 「APIを使用したデータベース統合リプレイの初期化」で説明されているように、リプレイ・データが初期化済です。
- 「APIを使用した接続の再マッピング」で説明されているように、取得済の接続が再マッピング済です。
- 「APIを使用したユーザーの再マッピング」で説明されているように、ユーザーがマッピング済です。

  ユーザーの再マッピングは任意です。ただし、リプレイ時にユーザーを再マッピングする予定の場合、統合リプレイを準備する前にそれが完了している必要があります。

統合リプレイの準備では、次の操作が実行されます。

- 同期モード、セッションの接続率およびセッションのリクエスト率などのリプレイ・オプションが指定されます。
- データベースがリプレイ・モードになります。
- リプレイ・クライアントの起動が有効になります。

統合リプレイを準備するには:

- PREPARE_CONSOLIDATED_REPLAYプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY (
   synchronization          IN VARCHAR2 DEFAULT 'SCN',
   connect_time_scale       IN NUMBER   DEFAULT 100,
   think_time_scale         IN NUMBER   DEFAULT 100,
   think_time_auto_correct IN BOOLEAN  DEFAULT TRUE,
   capture_sts              IN BOOLEAN  DEFAULT FALSE,
   sts_cap_interval         IN NUMBER   DEFAULT 300);
```

  これらのパラメータおよびその設定方法の詳細は、「リプレイ・オプションの指定」を参照してください。

関連項目:

PREPARE_CONSOLIDATED_REPLAYプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

APIを使用したデータベース統合リプレイの開始

この項では、DBMS_WORKLOAD_REPLAYパッケージを使用して、統合リプレイを開始する方法を説明します。

統合リプレイを開始する前に、次の前提条件が満たされていることを確認します。

- 「APIを使用したデータベース統合リプレイの準備」で説明されているように、統合リプレイが準備済です。
- 十分な数のリプレイ・クライアントが起動済です。

  リプレイ・クライアントの設定および開始の詳細は、「リプレイ・クライアントの設定」を参照してください。

統合リプレイを開始するには:

- START_CONSOLIDATED_REPLAYプロシージャを使用します。

```
DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

関連項目:

START_CONSOLIDATED_REPLAYプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

問合せのみのデータベース・リプレイについて

問合せのみのデータベース・リプレイでは、ワークロード取得の読取り専用問合せのみがリプレイされます。つまり、問合せのみのリプレイでは、リプレイ時にSELECT文のみがサーバーに送られます。問合せのみのリプレイ中はDML文は実行されず、リプレイではユーザー・スキーマやデータが変更されることはありません。

ノート:

問合せのみのデータベース・リプレイが一緒に実行できるのは、データベース統合リプレイのみです。

ノート:

問合せのみのデータベース・リプレイはテスト環境でのみ使用され、実行されます。

- 問合せのみのデータベース・リプレイを本番システムで使用しないでください。
- 問合せのみのデータベース・リプレイ中に相違が起こる可能性があります。

問合せのみのデータベース・リプレイの使用事例

問合せのみのデータベース・リプレイを使用して、データベース・バッファ・キャッシュをウォームアップしたり、パフォーマンスの低下を見つけることができます。たとえば:

- データベース・バッファ・キャッシュをウォームアップするため

  場合によっては、データベース・バッファ・キャッシュがウォームな(データ・ブロックがすでにバッファ・キャッシュ内に存在している)ときにワークロードが取得されます。ただし、テスト・システムでそのワークロードをリプレイする場合、バッファー・キャッシュはウォームではなく、データ・ブロックを最初にディスクからロードする必要があります。これは、リプレイ時間が取得時間よりも長くなる可能性があり、データベース時間が増加します。

  バッファー・キャッシュのウォームアップの必要性を避けるために、問合せのみのリプレイを実行し、その後データベースを再起動したりバッファー・キャッシュをフラッシュしたりせずに読取り/書込みリプレイを実行します。問合せのみのリプレイは読取り専用のため、問合せのみのリプレイの後にデータベースを再起動する必要がないことに注意してください。

- パフォーマンスの低下を見つけるため

  問合せのみのリプレイは、同時実行性を持つワークロードの読取り専用部分からパフォーマンスの低下を見つけるための優れた、簡単な方法です。読取り専用部分には、SELECT文(SELECT. . . FOR UPDATE文ではありません)、DMLおよびDDL以外のPL/SQL、LOB読取りなどがあります。通常、これがワークロード取得の主要部分です。

問合せのみのデータベース・リプレイの実行

問合せのみのデータベース・リプレイを実行できます。

問合せのみのデータベース・リプレイを実行するには、「APIを使用したデータベース統合リプレイの使用」の手順に従います。「APIを使用したリプレイ・スケジュールへのワークロード取得の追加」で説明されているように、ADD_CAPTUREファンクションを使用してリプレイ・スケジュールにワークロードの取得を追加するには、query_onlyパラメータをYに設定します。

例: APIを使用した統合済ワークロードのリプレイ

この項では、別のオペレーション・システムで別のバージョンのOracle Databaseを実行している3つの別の本番システムのワークロードを統合するシナリオを仮定しています。

このシナリオでは、次を想定しています。

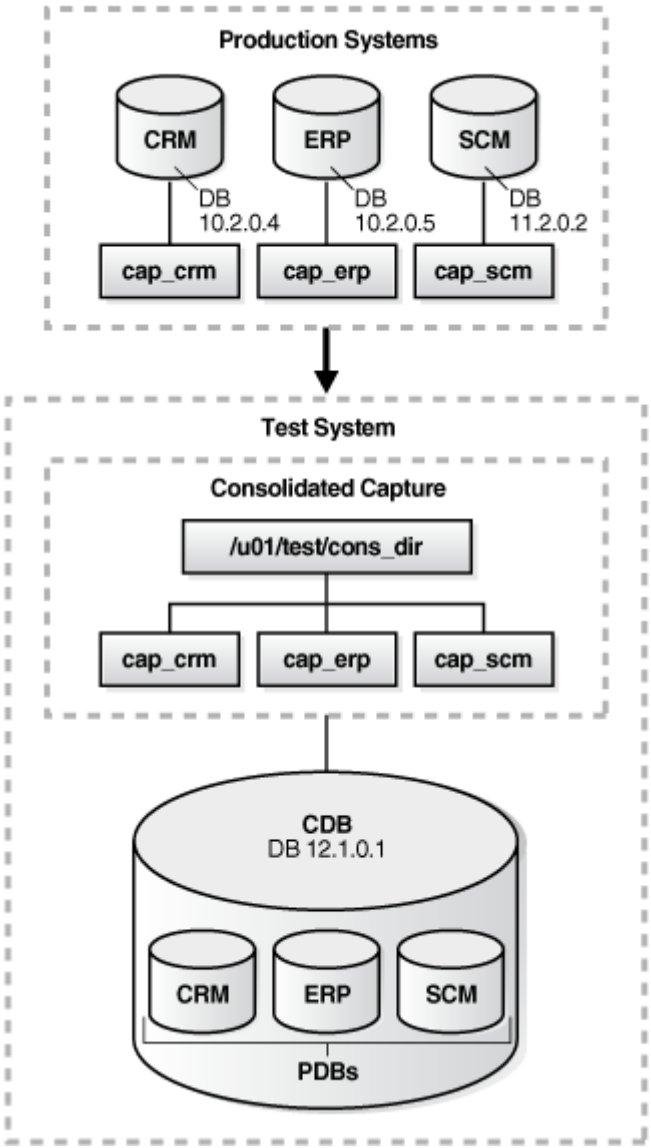
- 統合する最初のワークロードは、SolarisサーバーでOracle Database 10gリリース2(リリース10.2.0.4)を実行し

ているCRMシステムから取得します。

- 統合する2番目のワークロードは、LinuxサーバーでOracle Database 10gリリース2(リリース10.2.0.5)を実行しているERPシステムから取得します。
- 統合する3番目のワークロードは、SolarisサーバーでOracle Database 11gリリース2(リリース11.2.0.2)を実行しているSCMシステムから取得します。
- テスト・システムは、Oracle Database 12c リリース1 (リリース12.1.0.1)を実行するマルチテナント・コンテナ・データベース(CDB)として設定されます。
- CDBには、CRM、ERPおよびSCMで作成された3つのPDBが含まれています。
- CDBに含まれる各PDBは、取得の開始時にCRM、ERPおよびSCMシステムのアプリケーション・データと同じ状態に復元されています。

図15-3は、そのシナリオを示しています。

図15-3 3つのワークロードを統合するシナリオ



このシナリオで、ワークロードを統合し統合ワークロードをリプレイするには:

1. テスト・システムで、個々のワークロード取得を別のディレクトリに前処理します。
   - CRMワークロードには、次を実行します。

a. ディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cap_crm';
```

b. CRMシステムから取得したワークロードがこのディレクトリに格納されていることを確認します。

c. ワークロードを事前処理します。

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CRM');
```

- ERPワークロードには、次を実行します。

a. ディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY erp AS '/u01/test/cap_erp';
```

b. ERPシステムから取得したワークロードがこのディレクトリに格納されていることを確認します。

c. ワークロードを事前処理します。

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('ERP');
```

- SCMワークロードには、次を実行します。

a. ディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY scm AS '/u01/test/cap_scm';
```

b. SCMシステムから取得したワークロードがこのディレクトリに格納されていることを確認します。

c. ワークロードを事前処理します。

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('SCM');
```

2. 事前処理したワークロードを格納するルート・ディレクトリを作成します。

```
mkdir '/u01/test/cons_dir';
CREATE OR REPLACE DIRECTORY cons_workload AS '/u01/test/cons_dir';
```

3. 事前処理した各ワークロード・ディレクトリをルート・ディレクトリにコピーします。

```
cp -r /u01/test/cap_crm /u01/test/cons_dir
cp -r /u01/test/cap_erp /u01/test/cons_dir
cp -r /u01/test/cap_scm /u01/test/cons_dir
```

4. 各ワークロード用に、新しいオペレーティング・システムのディレクトリ・パスを使用して、ディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cons_dir/cap_crm';
CREATE OR REPLACE DIRECTORY erp AS '/u01/test/cons_dir/cap_erp';
CREATE OR REPLACE DIRECTORY scm AS '/u01/test/cons_dir/cap_scm';
```

5. ステップ2で作成したルート・ディレクトリをリプレイ・ディレクトリに設定します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_WORKLOAD');
```

6. リプレイ・スケジュールを作成し、ワークロード取得を追加します。

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('CONS_SCHEDULE');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CRM') FROM dual;
```

```
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('ERP') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('SCM') FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

7. 統合リプレイを初期化します。

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('CONS_REPLAY',
     'CONS_SCHEDULE');
```

8. 接続を再マッピングします。

- DBA_WORKLOAD_CONNECTION_MAPビューに接続マッピング情報を問い合せます。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

- 接続を再マッピングします。

```
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 1, replay_connection => 'CRM');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 1, replay_connection => 'ERP');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 1, replay_connection => 'SCM');
```

  replay_connectionパラメータは、テスト・システムに定義したサービスを示します。

- 接続の再マッピングを確認します。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

9. 統合リプレイを準備します。

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY (
     synchronization => 'OBJECT_ID');
```

10. リプレイ・クライアントを起動します。

- 必要なリプレイ・クライアントの数を見積もります。

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_crm
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_erp
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_scm
```

- 必要なリプレイ・クライアントの数を判断するために出力を追加します。

  統合したワークロードに含まれるワークロード取得ごとに、最低1つのリプレイ・クライアントを開始する必要があります。

- このコマンドを繰り返し、必要な数のリプレイ・クライアントを起動します。

```
wrc username/password mode=replay replaydir=/u01/test/cons_dir
```

  replaydirパラメータは、ワークロード取得が格納されているルート・ディレクトリに設定されています。

11. 統合リプレイを開始します。

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

関連項目:

- CDBの構成の詳細は、『Oracle Database管理者ガイド』を参照してください
- PDBの作成の詳細は、『Oracle Database管理者ガイド』を参照してください

16 ワークロード・スケールアップの使用

この章では、データベース統合リプレイで様々なワークロード・スケールアップ・テクニックを使用する方法を説明します。内容は次のとおりです。

- ワークロード・スケールアップの概要
- タイム・シフトの使用
- ワークロードの縮小の使用
- スキーマの再マッピングの使用

ワークロード・スケールアップの概要

データベース統合リプレイでは、1つ以上のシステムから取得した複数のワークロードを同時にリプレイできます。統合リプレイでリプレイが開始されると、統合されたすべてのワークロード取得のリプレイが開始されます。データベース統合リプレイを使用する際には、使用事例に応じて様々なワークロード・スケールアップ・テクニックを使用できます。

この項では、次のワークロード・スケールアップ・テクニックについて説明します。

- タイム・シフトについて
- ワークロードの縮小について
- スキーマの再マッピングについて

関連項目:

データベース統合リプレイの通常の使用事例の詳細は、「データベース統合リプレイの使用事例」を参照してください

タイム・シフトについて

データベース・リプレイでは、取得したワークロードをリプレイする際にタイム・シフトを実行できます。このテクニックは、既存の取得済ワークロードにワークロードを追加し、一緒にリプレイして、システムに対するストレス・テストを実行したい場合に便利です。

たとえば、営業、CRMおよびDWの3つのアプリケーションから取得した3つのワークロードがあるとします。ストレス・テストを実行するには、これらの取得済ワークロードのピークを合わせ、データベース統合リプレイを使用して一緒にリプレイします。

関連項目:

- タイム・シフトの使用方法の詳細は、「タイム・シフトの使用」を参照してください
- ストレス・テストでのデータベース統合リプレイの使用方法の詳細は、「ストレス・テスト」を参照してください

ワークロードの縮小について

データベース・リプレイでは、既存のワークロード取得を縮小してスケールアップ・テストを実行できます。たとえば、午前2時から午後8時のワークロードを取得したとします。データベース・リプレイを使用し、元のワークロードを3つの取得サブセット(1番目は午

前2時から午後8時、2番目は午前8時から午後2時、3番目は午後2時から午後8時)に縮小します。3つの取得サブセットを同時にリプレイすると、元の取得は縮小してリプレイ時にワークロードを3倍にして、スケールアップ・テストを実行できます。

関連項目:

- ワークロードの縮小の使用方法の詳細は、「ワークロードの縮小の使用」を参照してください
- 取得サブセットの詳細は、「取得サブセット」を参照してください
- スケールアップ・テストでのデータベース統合リプレイの使用方法の詳細は、「スケールアップ・テスト」を参照してください

スキーマの再マッピングについて

データベース・リプレイでは、データベース・スキーマを再マッピングして、スケールアップ・テストを実行できます。このテクニックは、マルチテナント・アプリケーションなどの同じアプリケーションのインスタンスを複数デプロイする際や、既存のアプリケーションに新しいに地理的なエリアを追加するときに便利です。

たとえば、営業のアプリケーションのワークロードが1つあるとします。スケールアップ・テストを実行して存在する場合にホストのボトルネックを特定するには、テスト・システムに営業スキーマの複数のスキーマを設定します。

関連項目:

- スキーマの再マッピングの使用方法の詳細は、「スキーマの再マッピングの使用」を参照してください
- スケールアップ・テストでのデータベース統合リプレイの使用方法の詳細は、「スケールアップ・テスト」を参照してください

タイム・シフトの使用

この項では、タイム・シフトを使用し、3つのアプリケーションから取得したワークロードのピークを合わせて同時にリプレイするシナリオを想定し、データベース統合リプレイでタイム・シフトを使用する方法を説明します。このシナリオでは、タイム・シフトを使用しストレス・テストを実行する方法を説明します。タイム・シフトの詳細は、「タイム・シフトについて」を参照してください。

このシナリオでは、次を想定しています。

- 最初のワークロードは、営業アプリケーションから取得します。
- 2つ目のワークロードは、ピーク時間が営業ワークロードの1時間前であるCRMアプリケーションから取得します。
- 3つ目のワークロードは、ピーク時間が営業ワークロードの30分前であるDWアプリケーションから取得します。
- これらのワークロードのピークを合わせるために、リプレイ時にCRMワークロードに1時間の遅延を、DWワークロードには30分の遅延を追加してタイム・シフトを実行します。

このシナリオでタイム・シフトを実行するには:

1. ストレス・テストを実行するリプレイ・システムで、取得したワークロードが保存されているルート・ディレクトリにディレクトリ・オブジェクトを作成します。

   ```
   CREATE [OR REPLACE] DIRECTORY cons_dir AS '/u01/test/cons_dir';
   ```

2. 取得した個々のワークロードを別のディレクトリに事前処理します。

- 営業ワークロードでは、次の手順を実行します。

  a. ディレクトリ・オブジェクトを作成します。

  ```
  CREATE OR REPLACE DIRECTORY sales AS '/u01/test/cons_dir/cap_sales';
  ```

  b. 営業アプリケーションから取得したワークロードがこのディレクトリに格納されていることを確認します。

  c. ワークロードを事前処理します。

  ```
  EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('SALES');
  ```

- CRMワークロードには、次を実行します。

  a. ディレクトリ・オブジェクトを作成します。

  ```
  CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cons_dir/cap_crm';
  ```

  b. CRMアプリケーションから取得したワークロードがこのディレクトリに格納されていることを確認します。

  c. ワークロードを事前処理します。

  ```
  EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CRM');
  ```

- DWワークロードには、次の手順を実行します。

  a. ディレクトリ・オブジェクトを作成します。

  ```
  CREATE OR REPLACE DIRECTORY DW AS '/u01/test/cons_dir/cap_dw';
  ```

  b. DWアプリケーションから取得したワークロードがこのディレクトリに格納されていることを確認します。

  c. ワークロードを事前処理します。

  ```
  EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('DW');
  ```

3. ルート・ディレクトリをリプレイ・ディレクトリに設定します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
```

4. リプレイ・スケジュールを作成し、ワークロード取得を追加します。

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('align_peaks_schedule');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('SALES') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CRM', 3600) FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('DW', 1800) FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

3,600秒(または1時間)の遅延がCRMワークロードに追加され、1,800秒の遅延(または30分)がDWワークロードに追加されます。

5. 統合リプレイを初期化します。

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('align_peaks_replay',
                          'align_peaks_schedule');
```

6. 接続を再マッピングします。

- DBA_WORKLOAD_CONNECTION_MAPビューに接続マッピング情報を問い合せます。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
```

```
  FROM dba_workload_connection_map;
```

- 接続を再マッピングします。

```
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 1, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 2, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 1, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 2, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 1, replay_connection => 'inst3');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 2, replay_connection => 'inst3');
```

replay_connectionパラメータは、テスト・システムに定義したサービスを示します。

- 接続の再マッピングを確認します。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

7. 統合リプレイを準備します。

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
```

8. リプレイ・クライアントを起動します。

- 必要なリプレイ・クライアントの数を見積もります。

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_sales
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_crm
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_dw
```

- 必要なリプレイ・クライアントの数を判断するために出力を追加します。

  統合したワークロードに含まれるワークロード取得ごとに、最低1つのリプレイ・クライアントを開始する必要があります。

- このコマンドを繰り返し、必要な数のリプレイ・クライアントを起動します。

```
wrc username/password mode=replay replaydir=/u01/test/cons_dir
```

  replaydirパラメータは、ワークロード取得が格納されているルート・ディレクトリに設定されています。

9. 統合リプレイを開始します。

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

ワークロードの縮小の使用

この項では、ワークロードの縮小を使用して取得したワークロードを3倍にするシナリオを想定し、データベース統合リプレイでワークロードの縮小を使用する方法を説明します。このシナリオでは、スケールアップ・テストでワークロードの縮小を使用する方法を説明します。ワークロードの縮小の詳細は、「ワークロードの縮小について」を参照してください。

このシナリオでは、次を想定しています。

- 元のワークロードは午前2時から午後8時まで取得され、3つの取得サブセットに縮小されました。
- 最初の取得のサブセットは、元のワークロードの午前2時から午後8時の部分です。
- 2つ目の取得のサブセットは、元のワークロードの午前8時から午後2時の部分です。
- 3つ目の取得のサブセットは、元のワークロードの午後2時から午後8時の部分です。
- リプレイ時にワークロードを3倍にするために、3つの取得サブセットを同時にリプレイすることでワークロードの縮小が実行されます。

このシナリオでワークロードの縮小を実行するには:

1. スケールアップ・テストを実行する計画のあるリプレイ・システムで、取得したワークロードが保存されているルート・ディレクトリにディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cons_dir AS '/u01/test/cons_dir';
```

2. 元のワークロードが保存されているディレクトリにディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cap_monday AS '/u01/test/cons_dir/cap_monday';
```

3. 取得サブセットを保存する予定のディレクトリにディレクトリ・オブジェクトを作成します。

   a. 最初の取得サブセットのディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cap_mon_2am_8am
                            AS '/u01/test/cons_dir/cap_monday_2am_8am';
```

   b. 2つ目の取得サブセットのディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cap_mon_8am_2pm
                            AS '/u01/test/cons_dir/cap_monday_8am_2pm';
```

   c. 3つ目の取得サブセットのディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cap_mon_2pm_8pm
                            AS '/u01/test/cons_dir/cap_monday_2pm_8pm';
```

4. 取得サブセットを作成します。

   a. 午前2時から午前8時の期間の1つ目の取得サブセットを生成します。

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                            'CAP_MON_2AM_8AM', 'mon_2am_8am_wkld',
                            0, TRUE, 21600, FALSE, 1);
```

   b. 午前8時から午後2時の期間の2つ目の取得サブセットを生成します。

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                            'CAP_MON_8AM_2PM', 'mon_8am_2pm_wkld',
                            21600, TRUE, 43200, FALSE, 1);
```

   c. 午後2時から午後8時の期間の3つ目の取得サブセットを生成します。

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                            'CAP_MON_2PM_8PM', 'mon_2pm_8pm_wkld',
                            43200, TRUE, 0, FALSE, 1);
```

5. 取得サブセットを事前処理します。

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_2AM_8AM');
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_8AM_2PM');
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_2PM_8PM');
```

6. ルート・ディレクトリをリプレイ・ディレクトリに設定します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
```

7. リプレイ・スケジュールを作成し、取得サブセットを追加します。

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('monday_folded_schedule');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_2AM_8AM') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_8AM_2PM') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_2PM_8PM') FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

8. 統合リプレイを初期化します。

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
                          'monday_folded_replay', 'monday_folded_schedule');
```

9. 接続を再マッピングします。

   a. DBA_WORKLOAD_CONNECTION_MAPビューに接続マッピング情報を問い合せます。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

   b. 接続を再マッピングします。

```
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 1, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 2, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 1, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 2, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 1, replay_connection => 'inst3');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 2, replay_connection => 'inst3');
```

   replay_connectionパラメータは、テスト・システムに定義したサービスを示します。

   c. 接続の再マッピングを確認します。

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

10. 統合リプレイを準備します。

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
```

11. リプレイ・クライアントを起動します。

   a. 必要なリプレイ・クライアントの数を見積もります。

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_2am_8am
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_8am_2pm
```

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_2pm_8pm
```

b. 必要なリプレイ・クライアントの数を判断するために出力を追加します。

統合したワークロードに含まれるワークロード取得ごとに、最低1つのリプレイ・クライアントを開始する必要があります。

c. このコマンドを繰り返し、必要な数のリプレイ・クライアントを起動します。

```
wrc username/password mode=replay replaydir=/u01/test/cons_dir
```

replaydirパラメータは、ワークロード取得が格納されているルート・ディレクトリに設定されています。

12. 統合リプレイを開始します。

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

スキーマの再マッピングの使用

この項では、アプリケーションにインスタンスを複数デプロイした場合にホストで発生する可能性のあるボトルネックを特定するためにスキーマの再マッピングを使用するシナリオを想定し、データベース統合リプレイでスキーマの再マッピングを使用する方法を説明します。このシナリオでは、スケールアップ・テストでスキーマの再マッピングを使用する方法を説明します。スキーマの再マッピングの詳細は、「スキーマの再マッピングについて」を参照してください。

このシナリオでは、次を想定しています。

- 営業アプリケーションから取得したワークロードが1つあります。
- 営業スキーマの複数のスキーマをリプレイ・システムに設定するために、取得したワークロードを複数回リプレイ・スケジュールに追加してユーザーを別のスキーマに再マッピングします。

このシナリオでスキーマの再マッピングを実行するには:

1. スケールアップ・テストを実行する計画のあるリプレイ・システムで、取得したワークロードが保存されているルート・ディレクトリにディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cons_dir AS '/u01/test/cons_dir';
```

2. 取得したワークロードが保存されているディレクトリにディレクトリ・オブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY cap_sales AS '/u01/test/cons_dir/cap_sales';
```

営業アプリケーションから取得したワークロードがこのディレクトリに格納されていることを確認します。

3. 取得したワークロードを事前処理します。

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_SALES');
```

4. ルート・ディレクトリをリプレイ・ディレクトリに設定します。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
```

5. リプレイ・スケジュールを作成し、取得したワークロードを複数回追加します。

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('double_sales_schedule');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_SALES') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_SALES') FROM dual;
```

```
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

6. 統合リプレイを初期化します。

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
                          'double_sales_replay', 'double_sales_schedule);
```

7. ユーザーを再マッピングします。

```
EXEC DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (2, 'sales_usr', 'sales_usr_2');
```

8. 統合リプレイを準備します。

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
```

9. リプレイ・クライアントを起動します。

   a. 必要なリプレイ・クライアントの数を見積もります。

   ```
   wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_sales
   ```

   b. 必要なリプレイ・クライアントの数を判断するために出力を追加します。

   統合したワークロードに含まれるワークロード取得ごとに、最低1つのリプレイ・クライアントを開始する必要があります。

   c. このコマンドを繰り返し、必要な数のリプレイ・クライアントを起動します。

   ```
   wrc username/password mode=replay replaydir=/u01/test/cons_dir
   ```

   replaydirパラメータは、ワークロード取得が格納されているルート・ディレクトリに設定されています。

10. 統合リプレイを開始します。

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

第III部 ワークロード分析

ワークロード分析は、本番データベースで実行されたSQL文のほぼリアルタイムの貴重な分析を提供します。

第III部では、ワークロード分析について説明し、次の章が含まれています:

- Enterprise Managerでのワークロード分析へのアクセス
- ワークロード分析の概要
- スケジュール済分析の使用
- 1回かぎりの分析の使用
- ワークロード分析タスクの比較レポートの確認

17 ワークロード分析の使用

ワークロード分析は、低下または改善の理由を特定、定量化および解消するのに役立ちます。

データベース・パフォーマンスが低下する一般的な理由は、問合せ計画の変更、データ量の増加またはデータベースでのアクティビティの増加によって発生する、パフォーマンスが低下したSQL文です。

ワークロード分析では、同じまたは類似の2つの異なる時点から、データベース内の上位問合せの分析が実行されます。パフォーマンスが低下した文は、SQLチューニング・アドバイザまたはSQL計画ベースラインを使用してチューニングできます。

Enterprise Managerでのワークロード分析へのアクセス

Oracle Enterprise Manager Cloud Controlのワークロード分析には、2つの方法を使用してアクセスします。

方法1:

1. 「ターゲット」ドロップダウン・リストをクリックします。
2. 「データベース」を選択します。
3. 「名前」列で、データベース名を選択します。たとえば、rep_databaseです。
4. 「パフォーマンス」ドロップダウン・リストから、「ワークロード分析」を選択します。
5. 「データベース・ログイン」画面で、「指定」または「新規」の資格証明を選択し、「ログイン」をクリックしてワークロード分析にアクセスします。

方法2:

1. 「ターゲット」ドロップダウン・リストをクリックします。
2. 「データベース」を展開し、「データベース・インスタンス」をクリックします。
3. 「ターゲット名」列で、データベース名をクリックします。たとえば、rep_databaseです。
4. 「パフォーマンス」ドロップダウン・リストから、「ワークロード分析」を選択します。
5. 「データベース・ログイン」画面で、「指定」または「新規」の資格証明を選択し、「ログイン」をクリックしてワークロード分析にアクセスします。

ワークロード分析の概要

ワークロード分析では、データベース上位SQL文のほぼリアルタイムの分析が提供され、履歴実行統計を使用して、変更されたパフォーマンスおよび変更されたパフォーマンスの理由が識別されます。

ワークロードとは、データベースまたはPDBで実行するSQL文のセットのことです。フィルタを使用してアプリケーション内の特定のアプリケーションまたはモジュールに制限することも、完全なデータベースまたはPDBにまたがることもできます。統計および実行計画を含むこれらの文は、SQLチューニング・セット(STS)に格納されます。自動ワークロード・リポジトリ(AWR)からSTSを収集する場合、dbms_workload_repository.modify_snapshot_settings(topnsql =>[number])で変更できる上位N個の文に制限されます。

ワークロード分析機能では、本番データベース内の異なる時点の2つのSQLチューニング・セットが比較され、これは変更前後のテスト・データベース内の1つのSQLチューニング・セットのみを分析するSQLパフォーマンス・アナライザと比較されます。特定の基準に基づいて、またはデータベースの上位文に基づいて、2つのSQLチューニング・セットを比較できます。

SQLパフォーマンス・アナライザはデータベース・レベルでのパフォーマンス・データの分析に役立ちますが、ワークロード分析はアプリケーション・レベルでのパフォーマンス・データの分析に役立ちます。

参照ワークロードを使用している場合は、ワークロード分析を使用したパフォーマンス・データの分析を開始する前に、ワークロードのSQLチューニング・セットを作成します。

現在使用可能なワークロード分析オプションは2種類あります。

- スケジュール済分析
- 1回かぎりの分析

スケジュール済分析と1回かぎりの分析の両方に、時間制限なしでデータを表示するオプションがあります。

関連トピック

- DBMS_WORKLOAD_REPOSITORYパッケージ

スケジュール済分析の使用

スケジュール済分析では、データベース管理者が構成したスケジュールに基づいてレポートが生成されます。

- スケジュール済分析について
- スケジュール済分析タスクの作成
- スケジュール済分析タスクの結果の確認
- スケジュール済分析タスクのリスト
- ワークロードおよびメトリックのサマリーの確認

スケジュール済分析について

スケジュール済分析を使用して、毎時、日次、週次、月次などのスケジュールで実行される2つのSQLチューニング・セットを比較するタスクを作成できます。

スケジュール済分析タスクの作成

ワークロード取得の詳細、ワークロード比較の詳細を指定し、タスクの日時をスケジュールして、スケジュール済分析タスクを作成します。

1. Enterprise Managerのデータベース・メイン・ページに移動します。
2. 「パフォーマンス」ドロップダウン・リストから、「ワークロード分析」を選択します。
3. スケジュール済分析ページを選択します。
4. 分析タスクの作成をクリックして、ワークロードのスケジュール済分析タスクを作成します。
5. 次のセクションの情報を入力します:

一般オプション

スケジュール済分析用に作成できる様々なタスクで使用可能なオプションです。

- 名前: スケジュール済タスクの名前を入力します。

- 説明: スケジュール済タスクの簡単な説明を入力します。

ワークロードの取得

「ワークロードの取得」セクションで、SQLチューニング・セットに関する情報を入力し、自動ワークロード・リポジトリ(AWR)スナップショットから取得したSQL文をロードします。

- SQLチューニング・セット名の接頭辞: SQLチューニング・セット名の前に付ける接頭辞を指定します。
- 自動ワークロード・リポジトリ(AWR)スナップショットを使用したSQL文のロード
  - カスタム時間範囲の指定: 時間範囲を指定してAWRスナップショットを取得し、これを新しいSQLチューニング・セットの作成時にSQL文のロードに使用できます。
    - 開始時間: AWRからAWRスナップショットを取得する開始時間を指定します。
    - 終了時間: AWRからAWRスナップショットを取得する終了時間を指定します。
  - 過去に作成されたスナップショットからのクイック選択: 時間数または日数を指定することで、過去に取得されたAWRスナップショットをドロップダウン・リストからすばやく選択できます。
- 取得されたSQL文の合計数: SQL文の取得方法を指定します。
  - すべて取得: すべてのSQL文を取得するには、このオプションを選択します。
  - 上位Nの取得: 上位10個または上位20個のSQL文など、指定した数のSQL文を取得するには、このオプションを選択します。
  - フィルタ・オプション: 演算子を使用して、解析スキーム名、SQLテキスト、SQL IDまたはモジュールのフィルタを追加できます。

ワークロードの比較

「ワークロードの比較」には、次のオプションがあります:

- 後続の比較
  - ローリング参照を使用した比較: 取得したワークロードを以前に取得したワークロードとローリング・ベースで比較できます。例: 前日のワークロードに対する今日のワークロード。
  - 固定参照を使用した比較: 固定参照は、特定の時点に取得される静的SQLチューニング・セットです。たとえば、比較の描画時に、1月1日以降、時間AとBの間で常にこの参照SQLチューニング・セットを使用して比較します。
- オプションの初期参照ワークロード

  他のワークロードを比較するための初期参照ワークロードとして使用できるSQLチューニング・セットを指定します。この値はオプションです。

  - 比較メトリック: 経過時間、CPU時間、バッファ読取り、ディスク読取り、物理I/O、ダイレクト書込みおよびオプティマイザ・コストに基づいてパフォーマンス・メトリックを比較します。
  - しきい値の変更: ここでは最小しきい値が必要です。
  - コンシューマ参照グループ: タスクを特定のリソース・グループに割り当てることができます
  - パージ結果を保持する日数: 優先時間範囲を選択することで、パージされた実行を保持できます
- スケジュール

  タスクの実行時間をスケジュールするには、特定の時間範囲とタイムゾーンを指定します。

  - 開始日: タスクを実行する開始日を指定します。
    - 即時: タスクをすぐに実行するには、このオプションを選択します。
    - 後で: 指定した日付にタスクを実行するには、このオプションを選択します。

- 終了日: タスクを実行する終了日を指定します。
  - なし: 特定の日付にタスクを終了しない場合は、このオプションを選択します。
  - 指定した日付: 特定の日付にタスクを終了するには、このオプションを選択します。
- 繰返し間隔: 毎時、日次、週次、月次などの時間プリファレンスに基づいてスケジュールを繰り返すには、このドロップダウン・オプションを選択します。

スケジュール済分析タスクの結果の確認

「結果」タブを使用して、結果を表示し、データベース・ワークロード用に作成したタスクを分析します。

表17-1 スケジュール済分析タスクの結果

項目	説明
分析	SQL チューニング・セットのパフォーマンス分析および比較レポートを提供します。比較レポートを表示するには、レポート対象の分析タスクを展開し、「比較レポート」をクリックします。スケジュールされたワークロードが低下または改善するたびに、レポートが生成されます。
説明	SQL チューニング・セットで取得されたワークロードの説明。
参照ワークロード	既存または以前の SQL チューニング・セットのワークロード。参照ワークロードをクリックして、SQL チューニング・セットの詳細を取得します。
比較済ワークロード	既存の SQL チューニング・セットのパフォーマンスを別の SQL チューニング・セットと比較します。比較済ワークロードをクリックして、SQL チューニング・セットの詳細を取得します。
最終更新日	タスクが最後に更新された日付。
作成者	タスクを作成したユーザー。たとえば、SYS、SYSTEM または SYMAN です。
メトリック比較	経過時間、CPU 時間、バッファ読取り、ディスク読取り、物理 I/O、ダイレクト書込み、オプティマイザ・コストなどのパフォーマンス・メトリックの比較に基づいて、タスクが「低下」、「改善」または「変更なし」の場合にステータスが表示されます。
欠落した SQL	参照ワークロードの一部であったが、比較済ワークロードに存在しなくなった SQL 文の数。
新規 SQL	比較済ワークロードで初めて出現したが、参照ワークロードに存在しなかった SQL 文の数。
前の結果	参照ワークロードの名前、比較済ワークロード、その他のメトリック比較など、タスクの以前の結果に関するレポートを表示します。

スケジュール済分析タスクのリスト

「分析タスク」タブには、次の列が表示され、作成したすべてのスケジュール済分析タスクがリストされます。

表17-2 分析タスクのリスト

項目	説明
タスク	タスクの名前。
説明	タスクの説明。
作成者	タスクを作成したユーザー。たとえば、SYS、SYSTEM または SYSMAN です。
最終実行日	タスクが最後に実行された日付。
次回実行日	タスクが次に実行されるようにスケジュールされている日付。
状態	タスクのステータス(スケジュールされているかどうか)。スケジュール済ジョブの詳細は、ステータスをクリックします。
有効	タスクが有効かどうか。
前の実行	以前に実行されたタスク。タスク番号をクリックして、作成者、開始日、完了日、経過時間、現在のステータスなどのタスクに関する情報を取得します。
垂直ドット・メニュー	タスクの実行を有効化、無効化、削除または停止するには、垂直ドット・メニューをクリックします。

ワークロードおよびメトリックのサマリーの確認

スケジュール済分析には、作成した分析タスクのワークロードおよびメトリックの簡単なサマリーを示す一連のパネルが用意されています。

- 変更されたワークロード: 変更されたワークロードの数。
- 監視対象のワークロード: 監視対象のワークロードの数。
- 低下したメトリック: パフォーマンスが改善および低下したメトリックは比較に基づいています。1つのSQLチューニング・セットに複数の比較を含めることができます。
- 改善されたメトリック: 改善されたSQLチューニング・セットの数。
- 未変更のメトリック: 変更されていないSQLチューニング・セットの数。

データベース・ページの右上にある「データの表示」ドロップダウン・リストからオプションを選択して、任意の時間または時間制限なしでワークロード分析データを表示することもできます。

1回かぎりの分析の使用

1回かぎりの分析ページを使用して、2つの類似したワークロードのパフォーマンス特性を比較するのに役立つ1回かぎりの分析タスクを作成します。たとえば、データベースのアップグレードまたはパッチ適用の前後にSQLチューニング・セットのパフォーマンスを測定する場合です。

- 1回かぎりの分析について
- 1回かぎりの分析タスクの作成
- 1回かぎりの分析タスクの結果の確認
- 分析およびメトリックのサマリーの確認

1回かぎりの分析について

1回かぎりの分析では、2つのSQLチューニング・セットの比較が実行されます。この分析を実行して、アプリケーションのアップグレードなどの既知の変更後のパフォーマンスを検証できます。

1回かぎりの分析タスクの作成

ワークロード定義および比較の詳細を指定して、ワークロードの1回かぎりの分析用の分析タスクを作成します。

1. Enterprise Managerのデータベース・メイン・ページに移動します。
2. 「パフォーマンス」ドロップダウン・リストから、「ワークロード分析」を選択します。
3. 1回かぎりの分析ページを選択します。
4. 1回かぎりの分析タスクの作成をクリックして、1回かぎりの分析タスクを作成するか、ワークロードの分析タスクをスケジュールします。
5. 1回かぎりの分析タスクの作成ウィンドウで、次のセクションの情報を入力します:

一般オプション

1回かぎりの分析用に作成できる様々なタスクで使用可能なオプションです。

- 名前: スケジュール済タスクの名前を入力します。
- 説明: スケジュール済タスクの簡単な説明を入力します。

ワークロードの比較

様々なSQLチューニング・セットのパフォーマンスを比較するための情報を入力します。

- 参照ワークロード: 参照ポイントとして設定するSQLチューニング・セットを検索して入力します。
- 比較済ワークロード: 既存のSQLチューニング・セットのパフォーマンスを別のSQLチューニング・セットと比較できるように、ワークロードを検索して入力します。
- 比較メトリック: 比較レポートの生成時に使用する値を入力します。複数のメトリックを選択すると、各メトリックによって、経過時間、CPU時間、バッファ読取り、ディスク読取り、物理I/O、ダイレクト書込み、オプティマイザ・コストなどのメトリックに基づいて圧縮レポートが生成されます。
- しきい値の変更: 文のパフォーマンスが低下したとみなされた場合のしきい値を%で入力します。

1回かぎりの分析タスクの結果の確認

1回かぎりの分析ページには、作成したすべての1回かぎりの分析ワークロード・タスクの結果が表示されます。次の情報がリストされます。

表17-3 1回かぎりの分析タスクの結果

項目	説明

項目	説明
分析	SQL チューニング・セットの 1 回かぎりのパフォーマンス分析および比較レポートを提供します。比較レポートを表示するには、レポート対象の分析タスクを展開し、「比較レポート」をクリックします。スケジュールされたワークロードが低下または改善するたびに、レポートが生成されます。
説明	SQL チューニング・セットで取得されたワークロードのタスクの説明。
参照ワークロード	既存または以前の SQL チューニング・セットのワークロード。参照ワークロードをクリックして、SQL チューニング・セットの詳細を取得します。
比較済ワークロード	既存の SQL チューニング・セットのパフォーマンスを別の SQL チューニング・セットと比較します。比較済ワークロードをクリックして、SQL チューニング・セットの詳細を取得します。
最終更新日	タスクが最後に更新された日付。
作成者	タスクを作成したユーザー。たとえば、SYS、SYSTEM または SYMAN です。
削除	1 回かぎりのタスクを削除するオプションです。
メトリック比較	経過時間、CPU 時間、バッファ読取り、ディスク読取り、物理 I/O、ダイレクト書込み、オプティマイザ・コストなどのパフォーマンス・メトリックの比較に基づいて、タスクが「低下」、「改善」または「変更なし」の場合にステータスが表示されます。
欠落した SQL	参照ワークロードの一部であったが、比較済ワークロードに存在しなくなった SQL 文の数。
新規 SQL	比較済ワークロードで初めて出現したが、参照ワークロードに存在しなかった SQL 文の数。
状態	完了または実行中のタスクの現在のステータス。

分析およびメトリックのサマリーの確認

1回かぎりの分析には、作成した分析タスクの簡単な分析およびメトリックを示す一連のパネルが用意されています。

- 差異のある分析: 差異がある1回かぎりの分析タスクの数が表示されます。
- 合計分析: 1回かぎりの分析タスクの合計数が表示されます。
- 低下したメトリック: パフォーマンスが改善および低下したメトリックは比較に基づいています。1つのSQLチューニング・セットに複数の比較を含めることができます。
- 改善されたメトリック: 改善されるSQLチューニング・セットの数。
- 未変更のメトリック: 変更されていないSQLチューニング・セットの数。

データベース・ページの右上にある「データの表示」ドロップダウン・リストからオプションを選択して、最後の24時間、1週間、30

日間またはこれらの組合せのワークロード分析データを表示することもできます。

ワークロード分析タスクの比較レポートの確認

スケジュール済または定時ワークロードに低下または改善があるたびに、SQLチューニング・セットのパフォーマンス分析および比較レポートが生成されます。

このレポートで、比較メトリック、SQL文のパフォーマンス・ブレークダウン、ワークロードの影響を受ける上位SQL文などのデータを確認します。

また、「タスク名」、参照ワークロード、比較済ワークロード、「実行名」、参照ワークロード所有者、比較済ワークロード所有者、参照SQL分析済、比較済SQL分析済などのページの上部にある情報を確認します。

- 比較レポートへのアクセス
- サマリー・レポートの確認
- 例: ワークロード分析レポート

比較レポートへのアクセス

ワークロード分析タスクの比較レポートを表示できます。

1. 「ワークロード分析」ホームページで、レポート対象の分析タスクを展開します。
2. レポート対象の比較メトリックの横にある「比較レポート」リンクをクリックします。
3. ワークロード分析レポート・ページに比較メトリックが表示されます。
4. 「レポートの保存」をクリックして、比較レポートをダウンロードします。

サマリー・レポートの確認

「サマリー」パネルには、「ワークロード分析」ホームページの分析タスクで選択した比較メトリックに対するワークロードの影響のサマリーが表示されます。

サマリーが使用可能な比較メトリックには、次のようなものがあります:

- ダイレクト書込み: ダイレクト書込みにより、セッションではI/O書込みリクエストをキューに入れ、オペレーティング・システムがI/Oを処理している間に処理を続行できます。
- 物理I/O: SQL文の実行時のダイレクト書込みとディスク読取りの合計。
- ディスク読取り: ユーザーがSQL問合せを実行すると、Oracleはまずデータベース・バッファ・キャッシュ(メモリー)からデータの取得を試み、メモリーでデータを使用できない場合にのみディスクから取得を試みます。
- バッファ取得: このSQL文のバッファ読取りの合計数(データベースがブロックにアクセスした回数)です。
- オプティマイザ・コスト: 実行計画のコストを計算します。変更されたコストは、索引および表の計画変更または新しい統計を示します。
- 経過時間: SQL文の経過秒数です。
- CPU時間: SQL文のCPU時間の合計。

例: ワークロード分析レポート

ワークロード分析レポートは、2つのSQLチューニング・セットを比較するレポートで、2つの実行期間の間の異常を識別できるよ

うに差異を強調表示します。

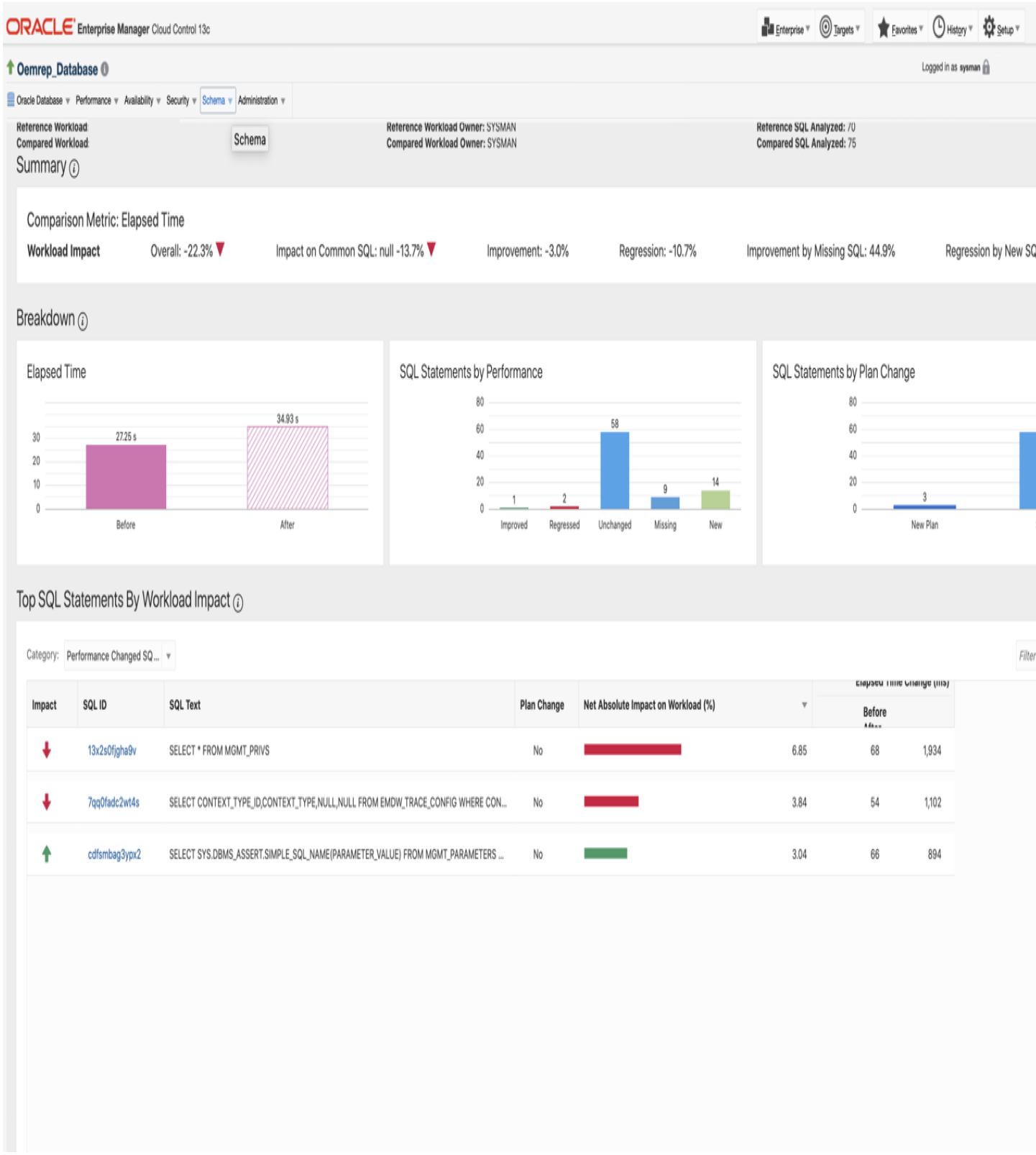
- ワークロード分析レポートの概要
- 「サマリー」セクション
- ブレークダウン
- ワークロードへの影響別上位SQL文
- SQLの詳細

ワークロード分析レポートの概要

2つの異なるSQLチューニング・セットを取得する場合は常に違いがあり、これらの違いが異常かどうかを解釈して理解する必要があります。

たとえば、分析されたワークロードが同じであるとします。先週月曜日の午前9時から午前10時の間にSQLチューニング・セットを取得し、これを月曜日の午前9時から午前10時の間に実行されたワークロードと比較しました。ワークロードは自動ワークロード・リポジトリ(AWR)から取得されたSQLチューニング・セットであるため、カテゴリごとに上位N個の文のみが含まれます。これにより、影響の少ない文の一部が他の文とスワップされる可能性があります。ただし、新しい文が上位のコントリビュータになることはありません。

図17-1 WLAレポートの概要



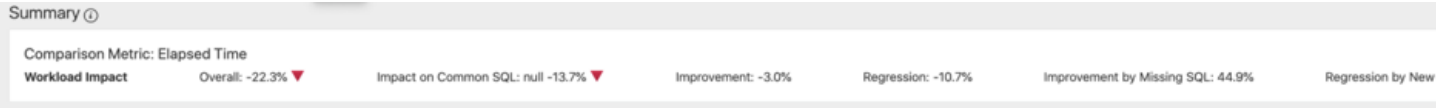
上部セクションには、比較されるワークロードの名前と、各ワークロードで取得されるSQL文の数に関する情報が表示されます。この例では、参照ワークロードに70個の文、比較済ワークロードに75個の文があります。

これは、比較期間のAWRにさらに文があることを示します。これは、参照ワークロードの様々なカテゴリの文が比較済ワークロードよりも調整されているか、AWRスナップショットに多くの文を含めるように変更されていることを意味する場合があります。

「サマリー」セクション

この例では、レポートは経過時間比較メトリックで生成されます。22.3%の全体的な低下を確認できます。

図17-2「サマリー」セクション



低下は様々なカテゴリに分かれています:

共通SQLへの影響: これらの文は両方の期間で実行されます。

改善: 参照ワークロードよりも合計経過時間が短いすべての文の合計改善点。

低下: 参照ワークロードよりも合計経過時間が長いすべての文の合計の低下。

欠落しているSQLによる改善: これは、比較済ワークロードで取得されていないすべての文の改善の合計です。統計が収集されていない場合、その統計は経過時間に影響しません。これらの文は、AWRの上位N個の文として他の文に置き換えることも、アプリケーションのアップグレードや同様のアクティビティによる文に置き換えることもできます。

新規SQLによる回帰: これは、参照ワークロードではなく、比較済ワークロードで取得されたすべての新規文に対する合計の影響です。新しい文の理由は、欠落している文の場合と同じです。

この例では、欠落している文および新規の文が合計ワークロードの約50%を占めています。

ブレークダウン

「ブレークダウン」セクションの情報は、常に「サマリー」セクションのデータと相関しています。経過時間を10%増やす場合は、経過時間の合計などのその他の状況を把握することが重要です。

図17-3「ブレークダウン」セクション

この例では、合計経過時間は約24秒です。したがって、ワークロードが1時間取得された場合に比較を実行する実際の値はありません。ただし、経過時間が約15000秒の場合、10%増加するとアプリケーションのパフォーマンスに影響する可能性があります。

経過時間

「経過時間」タイルには、その期間内に取得されたすべての文の実行の合計経過時間が表示されます。このタイルのラベルは、選択した比較メトリックを反映するように変更されます。

SQL文(パフォーマンス別)

このタイルには、各グループの文の数が表示されます。このデータを「サマリー」セクションで使用可能なデータに関連付けることが

重要です。この例では、欠落している文は文の合計数の12%ですが、合計経過時間のほぼ45%に寄与します。また、新しい文についても同様で、これはすべての文の19%になりますが、経過時間の54%に寄与します。

このユースケースでは、これらは経過時間に対する高いコントリビュータであり、説明なく消えることも表示されることもありません。

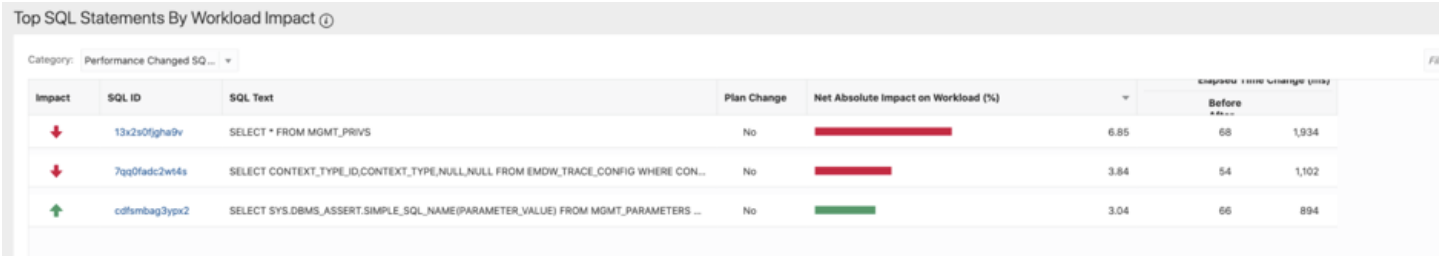
SQL文(計画変更別)

これは、SQL詳細を分析する必要がある重要なセクションです。新しい計画でパフォーマンスの改善が示されている場合は、1つの実行が高速か低速かを判断する必要があります。実行速度が遅い場合は、パフォーマンスの低下が発生しています。ただし、実行回数が少ないため、経過時間が短くなり、さらに調査する必要があります。パフォーマンスの低下を示している場合は、1回の実行の経過時間が長くなっているかどうかを調査する必要があります。ただし、1回の実行の経過時間が短い場合、それ以上の調査は必要ありません。

ワークロードへの影響別上位SQL文

このタイルでは、特定のカテゴリのSQL文のみを表示する様々なフィルタを追加できます。デフォルトでは、このタイルにはパフォーマンス変更済SQLが表示されますが、すべて表示、低下のみ表示、その他いくつかのカテゴリを表示することもできます。

図17-4 ワークロード別上位SQL文

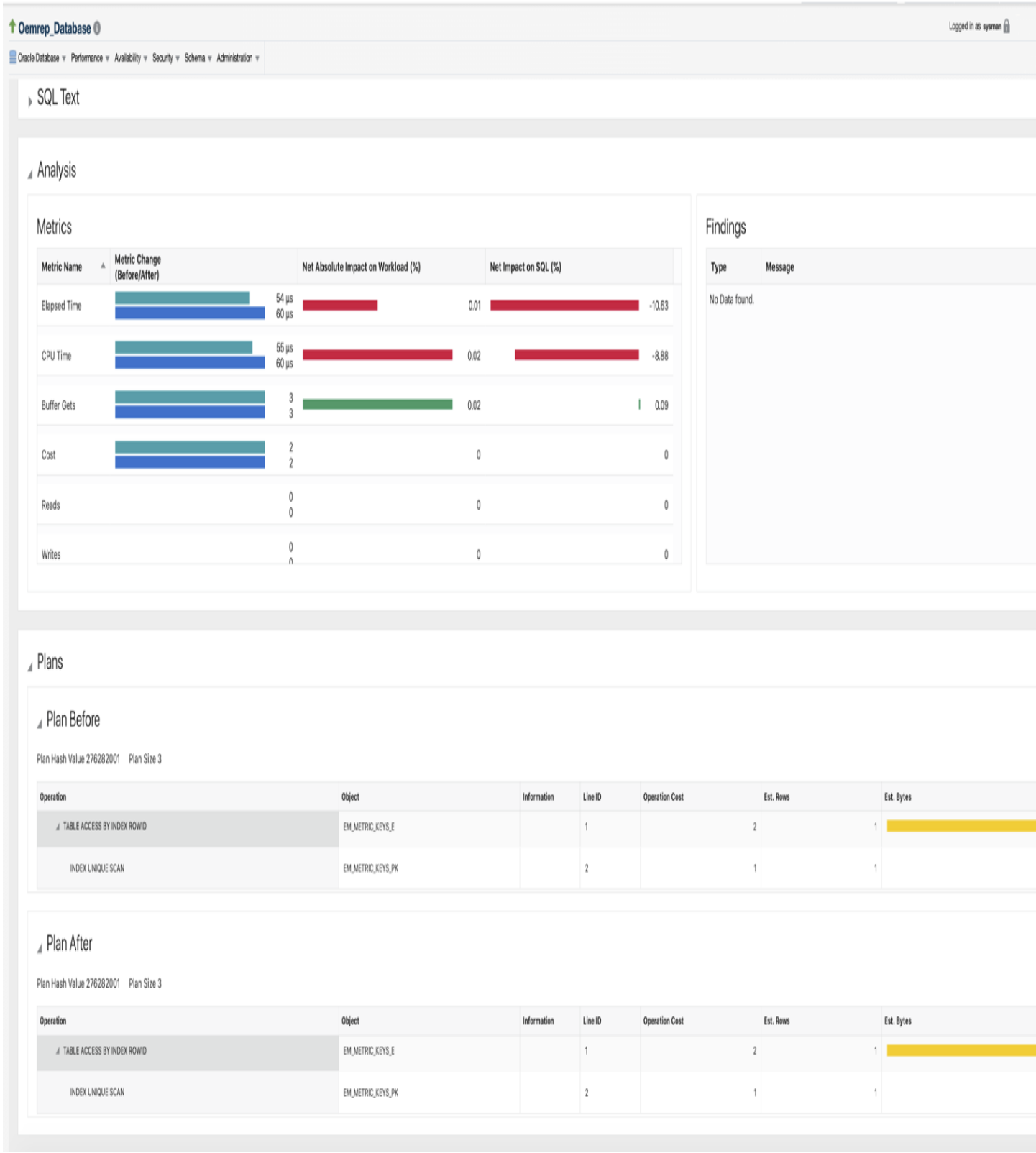


カテゴリを変更するには、ドロップダウン・リストからカテゴリを選択します。この例では、パフォーマンスが変更された3つのSQL文を確認できます。SQL文がパフォーマンス変更としてリストされるため、これが事前定義された影響のしきい値であることに注意してください。デフォルトのしきい値は3%で、ワークロードへの影響またはSQL文への影響のいずれかに適用されます。各SQL文の詳細を表示するには、SQL IDをクリックします。

SQLの詳細

SQLの詳細は、3つの異なるタイルに分割され、「分析」タイルと「計画」タイルが最も重要です。

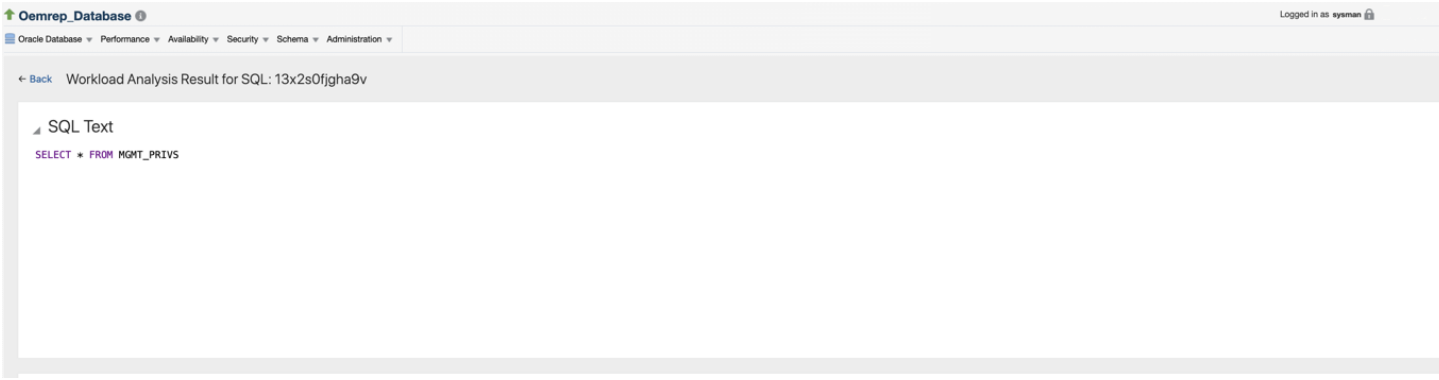
図17-5 SQLの詳細



SQLテキスト

このタイルはデフォルトで縮小されています。完全なSQLテキストを表示するには、これを展開します。

図17-6 SQLテキスト



分析

分析は、「メトリック」と「結果」の2つのタイルに分かれています。

図17-7「分析」の「メトリック」

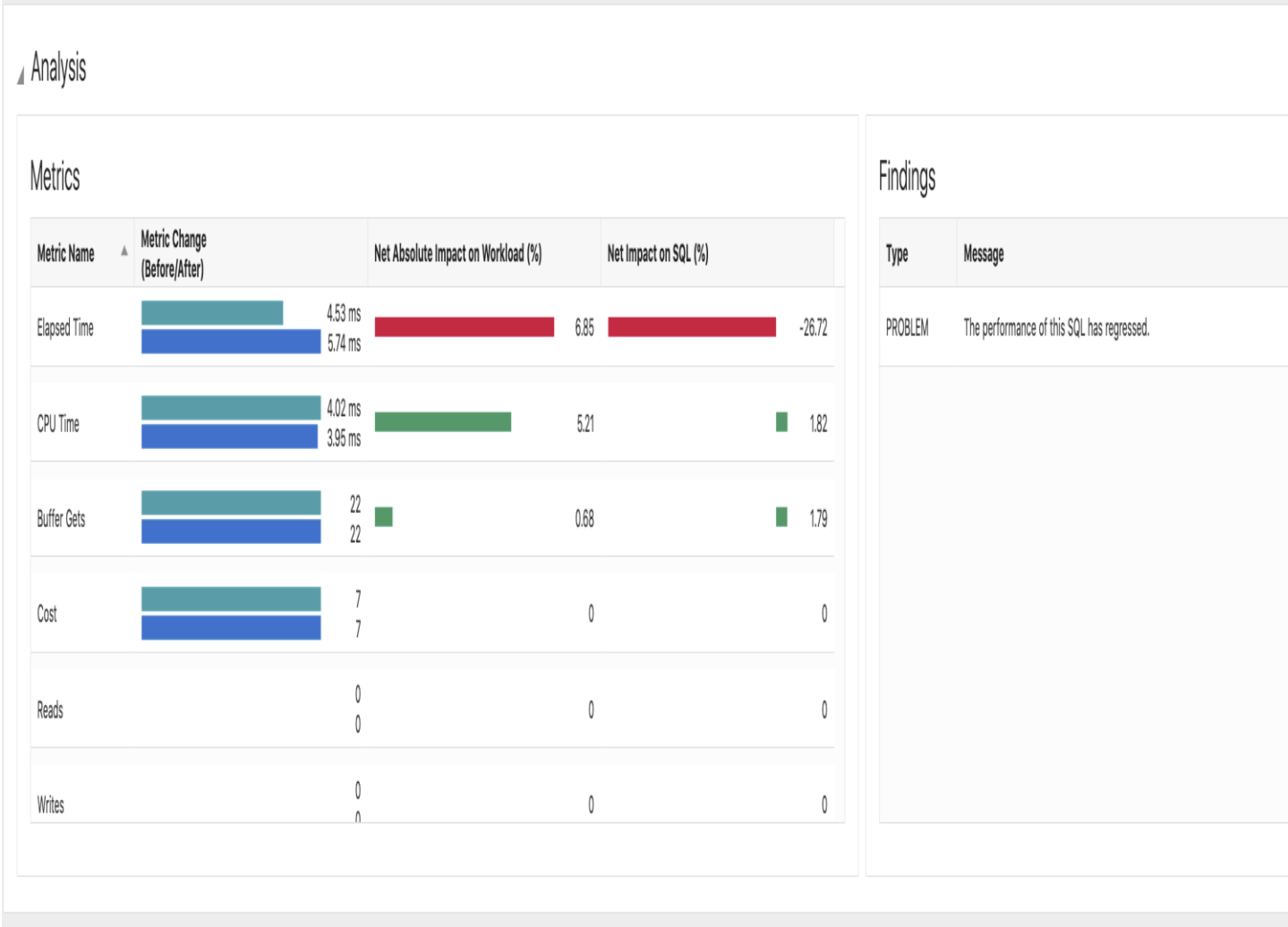
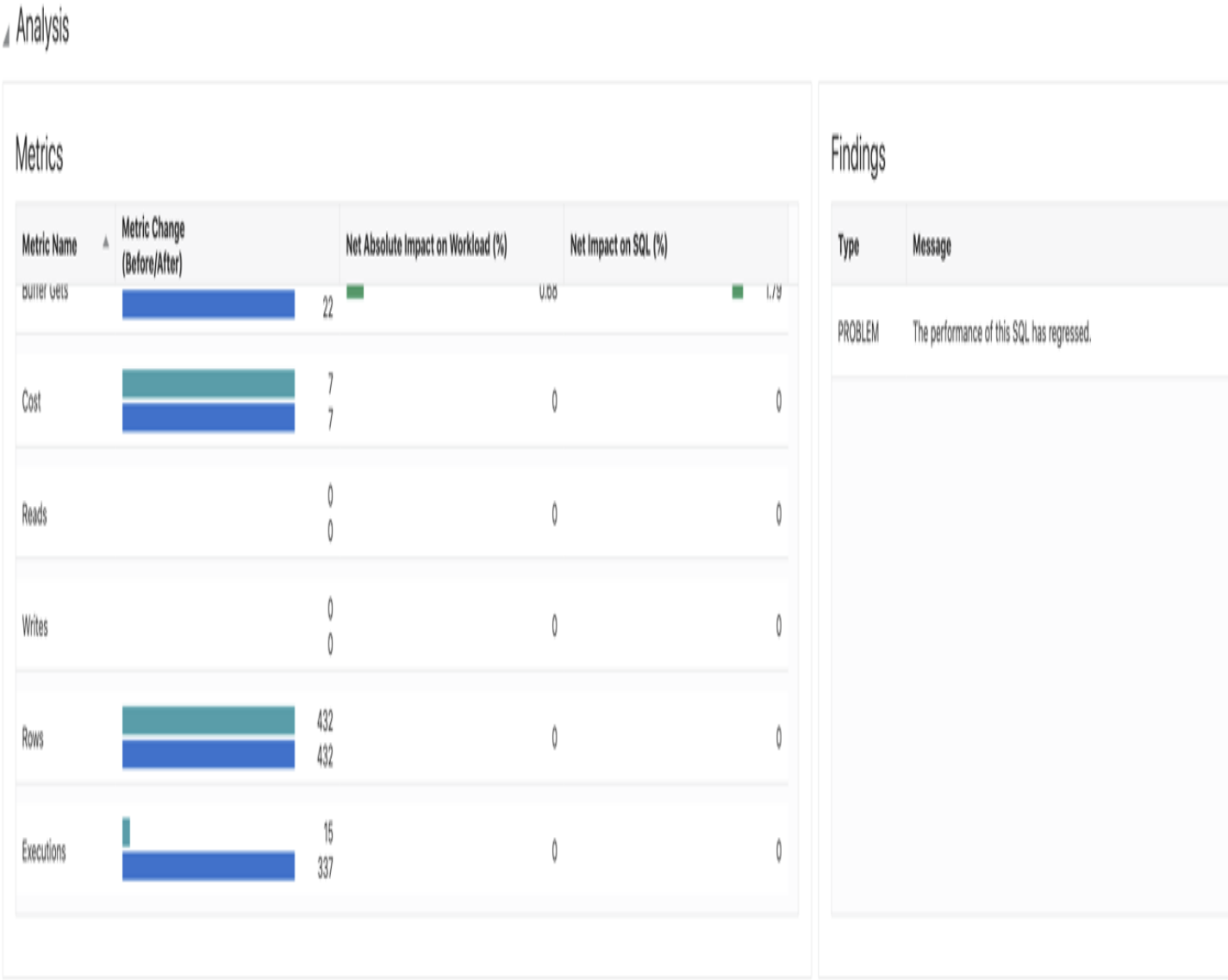
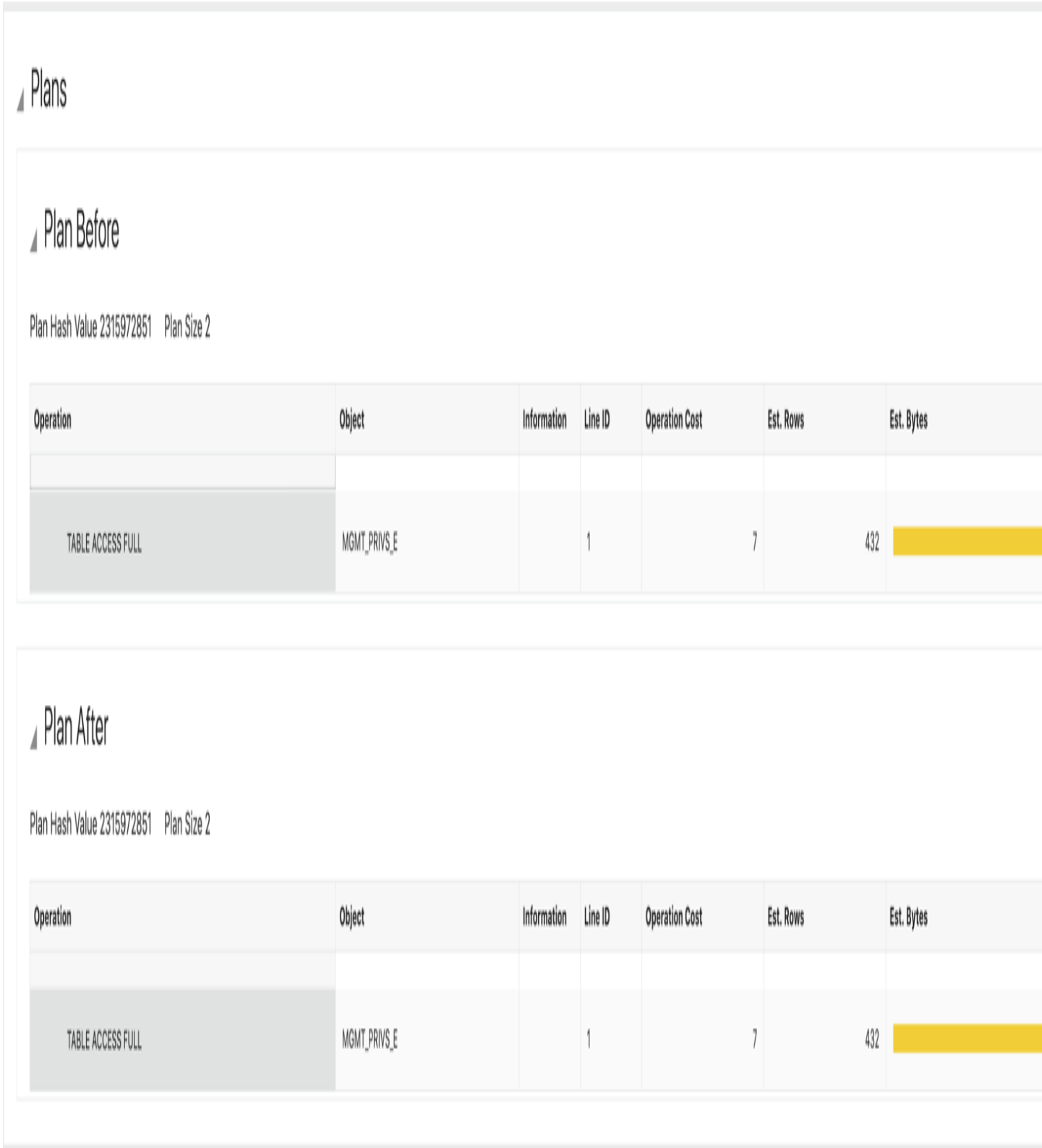


図17-8「分析」の「結果」



「メトリック」タイルには、すべての重要なメトリックの統計が含まれ、すべての差異の概要が表示されます。この例では、経過時間がわずかに変化し、CPU時間が短縮されています。バッファ読取りは同じで、各実行で同じ作業が実行されることを意味します。しかし、主な違いは、実行が16回ではなく、337回であることです。したがって、負荷ははるかに高くなります。これが、経過時間が長い理由です。負荷が高い場合は、待機が発生している可能性があります。CPUがあふれていることを示すI/Oはありません。

図17-9 計画の変更



パフォーマンスの変化のその他の理由は、計画の変更です。問合せに効率的な新しい計画または効率的でない計画があります。ワークロード分析レポートには、すべての共通の文について変更前の計画および変更後の計画が表示されます。

索引

C

D

M

R

S

U

W

- 使用 14.2