# Oracle Enterprise Manager

Oracle SQL Analyze **スタート・ガイド**

**リリース** 1.6.0

1998 **年** 9 **月**

**部品番号** :A61835-1

ORACLE®

Enabling the Information Age™

Oracle SQL Analyze スタート・ガイド、リリース 1.6.0

部品番号 :A61835-1

第 1 版 :1998 年 9 月

原本名 :Oracle Enterprise Manager Getting Started with Oracle SQL Analyze, Release 1.6.0

原本部品番号 :A63795-01

原本著者 :Anatole Wilson

原本協力者 :Priscilla Lee, Lisa Sheehan, Avon Hsu, Faisal Faruqi, Yao Feng, Vipul Shah, Jon Soule

# 目次

## 3 情報の収集と分析

# はじめに

Oracle SQL Analyze は、統合アプリケーション Oracle Enterprise Manager に含まれるソフトウェアの 1 つで、SQL 文の分析およびチューニングを支援します。Oracle Enterprise Manager Tuning Pack の一部である Oracle SQL Analyze は、構造化された、優れたチューニング方法論の要素の多くを補完し、自動化します。

Oracle SQL Analyze では、SQL 文、データベース・オブジェクトおよびパフォーマンス統計についての情報を収集するプロセスが自動化されています。また、ヒントの追加、SQL の構文チェック、結合メソッドの適用などを行うための、自動化された補助手段も提供されています。

このマニュアルでは、Oracle SQL Analyze ソフトウェアについて説明します。このマニュアルでは、Oracle SQL Analyze の使用方法を学習し、その機能のすべてを知ることができます。

この「はじめに」では、次の項目について説明します。



## このマニュアルの目的

このマニュアルでは、Oracle SQL Analyze ソフトウェアについて説明します。このマニュアルでは、Oracle SQL Analyze とそのグラフィカル・インタフェースについて説明し、データベース・チューニング・プロセス全体の一部としての Oracle SQL Analyze の使用方法についても説明します。

このマニュアルは、SQL 文のチューニングのためのガイドを提供しますが、チューニングにおける決定を下す際の明確な根拠を示すものではありません。データベースおよび SQL アプリケーションのチューニングの詳細は、関連資料およびその他の関連資料にリストしたマニュアルを参照してください。

# 対象読者

このマニュアルは、データベース管理者や Oracle ベースのアプリケーション開発者をはじめとして、SQL 文の記述を専門の職務とする、すべてのユーザーを対象としています。

# このマニュアルの構成

このマニュアルは 5 つの章から構成されます。

第 1 章　Oracle SQL Analyze の概要と、チューニング・プロセスにおけるその役割について説明します。SQL Analyze によるチューニング・プロセスの概要についても説明します。

第 2 章　SQL Analyze のインタフェースについて説明します。次に、Oracle SQL Analyze を起動し、チューニングの対象となる文を選択するプロセスを順を追って説明します。

第 3 章　情報を収集して分析し、SQL のパフォーマンスに関する問題の解決策を決定するプロセスを順を追って説明します。

第 4 章　SQL のチューニングの基礎概念を説明し、チューニング・ウィザードとヒント・ウィザードを使ってチューニング・プロセスを自動化する方法を示します。

第 5 章　SQL 文のパフォーマンスが、SQL のチューニング・セッションによって改善されたことを確認する方法について説明します。

# このマニュアルの表記規則

このマニュアルでは、次の表記規則を使用します。

| 表記規則 | 意味 |
|---|---|
| **本文中の太字** | 本文中の太字箇所は、本文、用語集またはその両方で定義されている用語を示します。 |
| => | メニュー項目の選択を示します。たとえば、「ファイル」=>「終了」は、「ファイル」メニューから「終了」を選択することを意味します。 |
| [ ] | 括弧で囲まれたキー名は、ユーザーが押すキーを示します。たとえば [F1] は、「F1」と記されたファンクション・キーを意味します。 |

# 関連資料

このマニュアルに記載されている情報の詳細は、Oracle Server のドキュメント・セットに含まれる、次のマニュアルを参照してください。

- 『Oracle Server 概要 : Vol.1』
- 『Oracle Server 概要 : Vol.2』
- 『Oracle Server リファレンス』
- 『Oracle Server SQL リファレンス : Vol.1』
- 『Oracle Server SQL リファレンス : Vol.2』
- 『Oracle Server チューニング』

### Oracle Enterprise Manager のドキュメント・セット

『Oracle SQL Analyze スタート・ガイド』は、Oracle Enterprise Manager のマニュアルの 1 つです。

#### Oracle Enterprise Manager の基本マニュアル

- 『Oracle Enterprise Manager Readme』オンライン・ドキュメント、ソフトウェアのアップグレード、およびその他の最新情報について説明しています。
- 『Oracle Enterprise Manager インストレーション・ガイド』Oracle Enterprise Manager のインストールについて説明しています。
- 『Oracle Enterprise Manager 管理者ガイド』Oracle Enterprise Manager、Oracle のシステム管理コンソール、共通サービス、および統合されたプラットフォーム・ツールの使用方法を説明しています。
- 『Oracle Enterprise Manager 概説』Oracle Enterprise Manager の概要を説明しています。
- 『Oracle Enterprise Manager 構成ガイド 』Oracle Enterprise Manager の構成方法を説明しています。
- 『Oracle Enterprise Manager アプリケーション開発者ガイド』Oracle Enterprise Manager コンソールの外部インタフェースのプログラミング方法を説明しています。
- 『Oracle Enterprise Manager メッセージ・マニュアル』Oracle Enterprise Manager のエラー・メッセージと、メッセージの診断方法を説明しています。

#### Oracle Enterprise Manager Change Management Pack ドキュメント・セット

- 『Oracle Enterprise Manager Change Management Pack Readme』Change Management Pack のオンライン・ドキュメント、ソフトウェアのアップグレード、およびその他の最新情報について説明しています。
- 『Oracle Change Management Pack スタート・ガイド』Oracle Change Management アプリケーションの概念と機能を説明しています。

Oracle Enterprise Manager Diagnostics Pack **ドキュメント・セット**

- 『Oracle Enterprise Manager Diagnostics Pack Readme』Diagnostics Pack のオンライン・ドキュメント、ソフトウェアのアップグレード、およびその他の最新情報について説明しています。
- 『Oracle Performance Manager および Oracle Capacity Planner スタート・ガイド』Oracle Performance Manager アプリケーションおよび Oracle Capacity Planner アプリケーションについて概念および機能を説明します。
- 『Oracle Enterprise Manager Oracle Trace ユーザーズ・ガイド』Oracle Trace アプリケーションを使って Oracle データベースを獲得し、履歴データを使って Oracle データベースを監視する方法を説明しています。
- 『Oracle Enterprise Manager Oracle Trace 開発者ガイド』アプリケーションで Oracle Trace ルーチンを使う方法を説明しています。
- 『Oracle TopSessions および Oracle Lock Manager スタート・ガイド』Oracle TopSessions および Oracle Lock Manager アプリケーションの概念と機能の概要を説明しています。

Oracle Enterprise Manager Tuning Pack **ドキュメント・セット**

- 『Oracle Enterprise Manager Tuning Pack Readme』Tuning Pack のオンライン・ドキュメント、ソフトウェアのアップグレード、およびその他の最新情報について説明しています。
- 『Oracle Enterprise Manager Oracle Expert ユーザーズ・ガイド』Oracle Expert を使って、初期構成中およびデータベース操作中に、データベース環境のパフォーマンスを最適化する方法を説明しています。
- 『Oracle Tablespace Manager スタート・ガイド』Oracle Tablespace Manager アプリケーションの概念と機能の概要を説明しています。

# その他の関連資料

このガイドで説明した概念について、より詳しく学びたい方は次の書籍を参照してください。

*Advanced Oracle Tuning and Administration*, by Eyal Aronoff, Kevin Loney, and Noorali Sonawalla.Oracle Press series, Osborne McGraw-Hill, 1997.

*Oracle:The Complete Reference*, third edition, by George Koch and Keven Loney.Oracle Press series, Osborne McGraw-Hill, 1997. *(Oracle 7.3)*

*Oracle:The Complete Reference, Electronic Edition*, by George Koch and Keven Loney.Oracle Press series, Osborne McGraw-Hill, 1997. *(Oracle 7.3)*

*Oracle8 Tuning*, by Michael Abbey, Michael J. Corey, Daniel J. Dechichio, and Ian Abramson.Oracle Press series, Osborne McGraw-Hill, 1997.

*Oracle Performance Tuning (Nutshell Handbook)*, second edition, by Mark Gurry and Peter Corrigan.O'Reilly & Associates, 1996.

*Oracle Performance Tuning and Optimization*, by Edward Whalen.Sams Publishing, 1996.

*Oracle SQL High-Performance Tuning*, by Guy Harrison.Prentice Hall Computer Books, 1997.

*Tuning Oracle*, by Michael Abbey, Michael J. Corey, and Daniel Dechichio, Jr. Oracle Press series, Osborne McGraw-Hill, 1995. *(Oracle 7.x)*

# 1
# 概要

この章では、次の項目について説明します。

# 概要

SQL 言語の主な利点の 1 つは、その柔軟性にあります。さまざまな異なるアプローチに対しても、同じ結果が得られます。しかし、それぞれのアプローチによって得られる結果が同じであっても、パフォーマンスの変化は、Oracle **オプティマイザ**によって選択される、データベース環境、索引構造およびアクセス・パスに大きく依存します。

効率的な SQL 文が最高のデータベース・パフォーマンスを維持できるのに対し、非効率的な SQL 文はパフォーマンスを低下させます。多くの場合、SQL 文のチューニングによって、全体的なパフォーマンスを 100% 以上の割合で改善できます。

しかしながら、これまでの間、SQL のチューニングは決して容易な作業ではありませんでした。SQL のチューニングには情報の収集と分析が伴い、また高度な知識と経験も必要です。SQL 文のチューニングには、次の要素が必要です。

- 現在の環境およびデータについての認識
- すべてのスキーマ・オブジェクトについての知識
- Oracle オプティマイザについての理解
- SQL についての詳しい知識

Oracle SQL Analyze は、特定のケースにおける最適なパフォーマンスを得るための、データベース環境およびスキーマ・オブジェクトについての情報の収集、SQL パフォーマンスの分析、オプティマイザのさまざまなアプローチの識別および比較、SQL 文の編集などを自動化するツールを提供します。

## Oracle SQL Analyze の利点

- 最も多くのリソースを消費する SQL 文を識別する、TopSQL 機能を提供。
- 容易に比較を行うための、さまざまな**オプティマイザ**・モード下での SQL の実行と、EXPLAIN PLAN および実行統計の提示。
- EXPLAIN PLAN のウォーク・スルーにより、実行順序と、操作の説明を提供。
- SQL 文の潜在的な結合順序および結合メソッドの分析をウォーク・スルーし、パフォーマンスを改善する代替 SQL を提供。
- 基本的な SQL 設計の「ルール」に SQL 文が違反していないかを自動的にチェックし、それらの違反を修正する代替 SQL を生成。
- SQL パフォーマンスに影響を与える問題を識別、修正する、**オブジェクト詳細**を提示。
- SQL パフォーマンスに直接影響を与える、初期パラメータ設定への容易なアクセスを提供。
- ヒント・ウィザードを使って、SQL 文に**ヒント**を追加。
- 将来の利用のため、SQL 文、実行計画およびパフォーマンス統計をリポジトリに保存。

# チューニング・プロセス全体の一部としての SQL のチューニング

当然のことながら、SQL のチューニングはチューニング・プロセス全体の一部にすぎません。『Oracle Server チューニング』で説明されているように、チューニングを検討する範囲は、SQL の他にもさまざまなものがあります。『Oracle Server チューニング』に規定されたチューニング方法では、次の手順でチューニングすることを提案しています。

1. ビジネス・ルールのチューニング
2. データ設計のチューニング
3. アプリケーション設計のチューニング
4. データベースの論理構造のチューニング
5. **SQL のチューニング**
6. アクセス・パスのチューニング
7. メモリのチューニング
8. I/O および物理構造のチューニング
9. リソース競合のチューニング
10. 基礎となるプラットフォームのチューニング

あるステップで下した決定が、後に続くステップに影響を与えることがあります。たとえば、ステップ 5 で SQL 文の一部を書き直したとします。これらの SQL 文は、ステップ 7 で処理される解析およびキャッシングの問題に重要な影響を与える可能性があります。また、ステップ 8 でチューニングされるディスク I/O は、ステップ 7 でチューニングされるバッファ・キャッシュのサイズに依存します。プロセスのどの時点にあっても、あるステップから、その前のステップのいずれかへとループ・バックする必要があるかもしれません。このチューニング・プロセスについては、『Oracle Server チューニング』で詳しく説明しています。

このマニュアルでは、主に SQL 文のチューニングについて説明します。しかし、後述のように、SQL の効率化には、データベースの論理構造、アクセス・パス、メモリ、I/O および物理構造のすべてが影響しています。

Oracle SQL Analyze は、さまざまな条件およびデータベース環境に対して SQL 文をテストするため、データベース構造についての情報を提供するとともに、いくつかの初期化パラメータを修正可能にすることによって、ユーザーの SQL チューニングを支援します。

## SQL のチューニングに関する問題

アプリケーションにおける SQL のパフォーマンスの低下には、多くの原因が考えられます。

- **SQL は、記述の簡単さに比べ、分析が難しい。**SQL は、学習が比較的容易な言語ですが、その非プロシージャ的な性質は、パフォーマンスに関連する問題をわかりにくくし

てしまう傾向があります。その結果、正しく機能する SQL を記述することに比べて、効率的な SQL を記述することは非常に難しいといえます。

- **データの収集およびパフォーマンスの分析は、困難で時間がかかる。**Oracle SQL Analyze が登場する以前は、ある SQL 文の、索引とビュー、EXPLAIN PLAN および実行計画など、さまざまなデータベース・オブジェクトを記述した統計情報を取得するため、それぞれに異なる SQL スクリプトを実行する必要がありました。そして、データベースに対して実行されるすべての SQL 文から、パフォーマンスを低下させている文を探し出すことは、非常に手間のかかる作業でした。Oracle SQL Analyze は、ユーザーに代わって関連するデータを収集し、SQL 文が消費するリソースによって、パフォーマンスを低下させている文の識別を支援します。
- **プログラマは、オプティマイザが最適な決定をすることを想定。**Oracle オプティマイザは、SQL 文を実行する最も効率的な方法を決定する、Oracle Server の一部です。Oracle Server は、考えられる何通りものアプローチの中から、どの SQL プログラマよりもすばやく決定を下すことができます。しかし、プログラマは、オプティマイザには不足している、アプリケーションの性質または環境についての非常に重要な情報を持っている場合があります。オプティマイザは、優秀な助手ですが、経験を積んだ SQL プログラマのような優れた決定を下すことはできません。

  SQL Analyze は、異なるオプティマイザ・モードと実行計画の間で比較を行うために、ユーザーが環境情報を調整できるようにします。この機能は、SQL 文を実行する最も効率的な方法を決定する作業を支援します。

# SQL の分析およびチューニング方法

SQL 文のチューニングを行う際には、問題の範囲を特定するために、環境データおよびパフォーマンス統計情報を収集し、分析できなければなりません。次の項では、SQL 文のチューニングにあたって、ユーザーが収集できる情報と、Oracle SQL Analyze で利用できるチューニング方法について説明します。

## EXPLAIN PLAN の分析

EXPLAIN PLAN は、SQL 文を実際に実行せずに、SQL 文の実行パスのステップを評価できるようにします。EXPLAIN PLAN によって、次の情報が示されます。

- SQL 文の実行の相対「コスト」（コストベース・オプティマイザを使う場合）
- オプティマイザによって選択された実行パス
- 使われている （または、使われていない）索引の種類
- 使われている結合メソッドの種類
- 結合の実行順序

Oracle SQL Analyze を使って、利用可能なオプティマイザ・モード （次項を参照）のそれぞれに対して、EXPLAIN PLAN の生成とウォーク・スルーを行えます。Oracle SQL Analyze は、EXPLAIN PLAN のグラフィカル・ビューと、結合が実行される様子を詳細に示す**コンパクト・ビュー**を作成します。

EXPLAIN PLAN の生成および「ウォーク・スルー」の詳細は、3-29 ページの「EXPLAIN PLAN のウォークスルー」を参照してください。

## オプティマイザ・モードの制御

Oracle オプティマイザは、SQL 文を実行するための最も効率的な方法を見つけるツールです。オプティマイザには、**「ルール」**、**「コスト（応答時間）」**、**「コスト（スループット）」**および**「選択」**の、4 つの主要な操作モードがあります。モードを選択して、オプティマイザの方針を決めます。

- 「ルール」モードでは、考えられるすべての実行パスが評価され、構文上の規則に基づいて、代替の実行パスが評価されます。
- 「コスト（応答時間）」モードでは、データの第 1 行を最も効率的に取り出す方法で SQL 文が実行されます。
- 「コスト（スループット）」モードでは、指定されたすべての列を最も効率的に処理する方法で SQL 文が実行されます。
- 「選択」モードでは、分析対象が表の場合には「コスト（応答時間）」モードで起動され、そうでない場合は「ルール」モードで起動されます。

これらのモードの詳細は、4-2 ページの「ヒントの理解」を参照してください。データベースの init.ora ファイル中の OPTIMIZER_MODE パラメータを指定することにより、オプティ

マイザのデフォルトの目的を設定できます。また、SQL 文に**ヒント**を追加することにより、ある SQL 文のためにオプティマイザを設定できます。

しかし、SQL 文に対して、どのオプティマイザ・モードが最も有効であるかを知りたい場合もあります。Oracle SQL Analyze では、SQL 文に対して、これらの実行方針がそれぞれテストされ、最適なモードを決定するためのコスト情報およびパフォーマンス統計情報が提供されます。

## ヒントの追加

問合せの内部において、その問合せの処理にコストベース・オプティマイザを使うことを指示するための**ヒント**を指定できます。

ヒントは次の要素に影響を与えます。

- 実行パス

  前述のように、ヒントを使ってオプティマイザ・モードを決定できます。

- データ・アクセス方法

  SQL 文が実行される間、ヒントを使って、オプティマイザが特定の走査方法を使うようにできます。たとえば、ヒントによって、全表走査の代わりに索引走査を使うようオプティマイザに指示できます。

- 結合メソッド

  ヒントによって、オプティマイザが結合列を選択する方法を変えることができます。

- パラレル実行

  ヒントを使って、パラレル操作を拡張することができます。これにより、コストを大きく低減できる場合があります。

  Oracle SQL Analyze は、構文的に正しいヒントを SQL 文に追加するための、ヒント・ウィザードを提供します。

## ルールの適用

特定の構文法がパフォーマンスに悪影響を与える場合もあります。Oracle SQL Analyze では、伝統的な規則の集合に照らして SQL 文を評価し、非効率的なコーディングを識別して、可能であれば代替の文を提示します。SQL Analyze のチューニング・ウィザードを使って、これらの規則を自動的に評価できます。

## 結合メソッドの適用

ほとんどの SQL 問合せは、複数の表からのデータ選択を伴います。これらの操作では、複数の表からのデータが結合され、目的の結果セットが生成されます。使われる結合メソッドの種類および表結合の順序は、索引の存在や、選択プロセスに関連した列のカーディナリティなど、さまざまな要素によって決定されます。

SQL ヒントを使うと、結合の方法論を制御できます。これは、オプティマイザ側からは知ることができない、次のような詳細を開発者が認識している、特定の問合せに対して有効です。

- 問合せのパフォーマンス目標 （スループット vs 応答時間）
- 特定の種類のオブジェクトに対する、旧式または存在しない統計情報
- フィルタ条件に影響を与える可能性があるバインド変数

ヒントを使って結合メソッドおよび特定の問合せ順序を制御することは、複雑かつ危険な操作です。この操作を行う開発者を支援するために、SQL Analyze は、自動化された方法論を提供しています。この方法論は、適切な局面で代替の結合方針を評価するために使われます。Oracle 内部のコストベース・オプティマイザと同様に、SQL Analyze は文の実行コストをさまざまな方法で推定します。Oracle SQL Analyze はオブジェクトの統計情報を推定し、最適な結合順序を完全に評価するために使用できる、バインド変数の標準的な値を収集します。代替の結合順序を使用できる場合、Oracle SQL Analyze は必要なヒントを使って、あるいはオブジェクトの順序を入れ替えて、文を記述し直します。

## オブジェクト詳細の分析

SQL 文のパフォーマンスは、アクセスされるオブジェクトの領域使用状況によっても影響を受けます。表内に連鎖行が存在するなどの要素によって、データ・セットの取り出しに必要な I/O の数が増加します。

# SQL チューニング・プロセス

この項では、Oracle SQL Analyze を使って、問題のある SQL 文を識別し、より効率的な文にチューニングするための方法論を提示します。各ステップに含まれる概念をより深く理解するには、第 4 章の「SQL 文のチューニング」を参照してください。

## ステップ 1: チューニング・セッションの開始

チューニング・セッションを開始する方法は複数あり、チューニング対象の SQL の状態によって異なります。

- データベース上で実行されている、または実行する予定の SQL 文の問題を識別するため、TopSQL を使って文を分析。問題のある、またはチューニングが必要な文を決定した後、ステップ 2 の「情報の収集」に進みます。
- 新しい文を作成するため、新しい SQL 文を入力するか、既存の SQL 文を Oracle SQL Analyze にインポートまたはコピー。
- SQL ファイルを開き、ファイル内部で文を編集。
- 前のチューニング・セッションから、編集中の SQL 文に戻る。
- 前のセッションに戻り、SQL 文の編集を再開。

## ステップ 2: 情報の収集

チューニングする SQL 文を選択した後、文が実行されているデータベース環境や、文のパフォーマンスについてさらに理解する必要があります。

- チューニング環境について詳しく知るため、データベース・パラメータおよび初期化パラメータを参照。
- SQL 文のパフォーマンスについて詳しく知るため、EXPLAIN PLAN を生成。EXPLAIN PLAN をウォーク・スルーすることにより、文がどのように実行されるかがわかります。EXPLAIN PLAN を別の EXPLAIN PLAN と比較することもできます。EXPLAIN PLAN の内部では、次のような情報を収集できます。
  - パフォーマンス統計情報
  - オブジェクト、挿入表、クラスタ、表およびビューについての詳細
  - オプティマイザによって選択された結合順序

## ステップ 3: 文のチューニング

統計のレビュー後、文をチューニングします。次の操作が行えます。

- 文を手動で編集。
- ヒント・ウィザードを使って、次の情報を指定するヒントを追加。
  - SQL 文の最適化アプローチ
  - SQL 文に対してのコストベース・アプローチの目標
  - SQL 文によってアクセスされる表のアクセス・パス
  - 結合文の結合順序

    - 結合文における結合操作。
- チューニング・ウィザードを使って、最適なパフォーマンスを得るためのチューニングを行う。チューニング・ウィザードでは、次の操作を行えます。
    - 文にヒントを追加
    - SQL の構文ガイドライン （ルール）を適用
    - 結合の方法論の分析および最適化
    - チューニング済みの文を元の文と比較して、ユーザーによるチューニングの有効性を測定

## ステップ 4: 結果の検証

情報の収集に使ったのと同じ方法で、SQL 文のパフォーマンスが改善されたかどうかを検証できます。

- 新しい文を実行して、結果を比較
- 新しい EXPLAIN PLAN を生成して、それらを比較
- オブジェクト詳細をレビューして、それらが有効に使われていることを確認

# 2
# チューニング・セッションの開始

この章では、次の項目について説明します。

# チューニング・セッションの開始

チューニング・セッションを開始するには、次のロールおよび権限が必要です。

- SQLADMIN ロールまたは DBA ロールの割り当て（次項で説明）
- "CREATE TABLE" 権限

Oracle SQL Analyze は、Oracle Enterprise Manager ツールバー、またはメニューから起動できます。

メニューから Oracle SQL Analyze を起動するには、**「ツール」**=>「Tuning Pack」=>「Oracle SQL Analyze」を選択します。

> **注意：** Oracle SQL Analyze はリポジトリからデータにアクセスするため、Oracle Enterprise Manager がバックグラウンドで常に動作している必要があります。

## SQLADMIN ロールの割り当て

Oracle SQL Analyze を実行するには、実行するユーザーに SQLADMIN ロールを割り当てる必要があります。

> **注意：** このロールに含まれている許可は、DBA ロールにも含まれています。したがって、DBA ユーザーに SQLADMIN ロールを割り当てる必要はありません。

SQLADMIN ロールを作成するプロセスを自動化するために、VMQROLE.SQL スクリプトが提供されています。このスクリプトは、$ORACLE_HOME\SYSMAN\ADMIN ディレクトリにあります。

1. Oracle Enterprise Manager のプログラム・グループから「SQL Worksheet」アイコンをダブルクリックして、SQL Worksheet を起動します。
2. 「ログイン情報」ダイアログ・ボックスで必要な情報を入力して、SQL Analyze の実行対象データベースに接続します。SYS としてログインします。
3. 「Worksheet」メニューから VMQROLE.SQL スクリプトを実行して、管理対象のデータベースに対する SQLADMIN ロールを作成します。
4. SQL Worksheet 下部のペインで次のように入力して、ユーザーに SQLADMIN ロールを割り当てます。

```
GRANT SQLADMIN to <user>
```

5. SQL Worksheet を終了します。

# チューニング・セッションの作成とセッションでの作業

チューニング・プロセス全体を通じて、いくつかの独立したペインに分けられたメイン・ウィンドウで作業を行います。メイン・ウィンドウに含まれるペインの種類および位置は、実行中の操作によって異なります。たとえば、EXPLAIN PLAN の生成および分析を行っているとき、ウィンドウは「ナビゲータ」ウィンドウ、「SQL テキスト」ウィンドウおよび「詳細」ウィンドウに分けられます。チューニング・ウィザードを実行しているとき、ウィンドウは「ナビゲータ」ウィンドウと「ウィザード」ウィンドウに分けられます。

次の項では、Oracle SQL Analyze のインタフェースを利用して、さまざまな操作を行う方法を説明します。

## Oracle SQL Analyze リポジトリ

Oracle SQL Analyze は、チューニング・セッションの情報を Enterprise Manager リポジトリに格納します。リポジトリに追加される領域は比較的小さなものですが、「**ファイル**」メニューから「**リポジトリに保存**」を選択したときに、次の情報が保存されることを理解しておいてください。格納される情報には、次のようなものが含まれます。

- ナビゲータ・ツリーを再構成するために必要な情報。初期化パラメータ情報や、SQL オブジェクトおよび EXPLAIN PLAN オブジェクトの名前などが含まれます。
- データベース・パラメータ。
- SQL 文。
- すべての EXPLAIN PLAN と関連する統計。

「ナビゲータ」ウィンドウには、Oracle Enterprise Manager によって提供される情報が反映され、検出したノードの一覧が表示されます。

---

**注意：** 切断されたノードに接続されているチューニング・セッションが存在する場合、Oracle SQL Analyze では、接続が既に失われているノードが表示され続ける場合があります。

---

オブジェクトの統計および統計の推定は、リポジトリに保存されません。

### リリース 1.5.5 から 1.6.0 への移行

Oracle SQL Analyze1.5.5 では、SQL のチューニング情報を作業領域に保存するオプションが存在しました。SQL Aalyze の作業領域は、SQL のチューニングに関する情報を格納するために SQL Analyze によって使われる Oracle 表から構成されます。SQL Analyze の作業領域の作成を選択した場合、SQL Analyze データベース・ユーザー接続のスキーマ内にこれらの表が作成されます。作業領域は、SQL Analyze データベース接続のそれぞれに対して作成できるため、SQL Analyze 1.5.5 で複数の作業領域が作成されている場合があります。

リリース 1.6.0 からは、SQL Analyze のリポジトリが Oracle Enterprise Manager のリポジトリと統合されています。この統合により、以前に検出されたノードや優先接続情報リストの設定など、Oracle Enterprise Manager のリポジトリに格納された情報を利用できるようになりました。

バージョン 1.5.5 の Oracle SQL Analyze で作成した作業領域が、Oracle Enterprise Manager1.5.5 のリポジトリの外部に存在する場合、以前の作業を利用するには、SQL Analyze の作業領域スキーマから Enterprise Manager リポジトリのスキーマに表を移動する必要があります。バージョン 1.5.5 で作成した作業領域が複数存在する場合、それらの内 1 つしか移行できないことに注意してください。このような移行作業は、SQL Analyze1.6.0 を使い始める前に済ませておく必要があります。

表を移動するには、Oracle Server の Export ユーティリティまたは Import ユーティリティ、あるいはその両方を使います。この操作の詳細は、『Oracle Server ユーティリティ』およびプラットフォーム固有のマニュアルに説明されています。Oracle Enterprise Manager のリポジトリに移動しなければならない表には、次のものがあります。

- VMQ_SQL_DATABASE
- VMQ_SQL_DATABASE_PARAMS
- VMQ_SQL_SESSION
- VMQ_SQL_SESSION_PARAMS
- VMQ_SQL_OBJECT
- VMQ_SQL_TEXT
- VMQ_SQL_IMPORT_STATS
- VMQ_SQL_PLAN_RULE
- VMQ_SQL_PLAN_COST_FIRST
- VMQ_SQL_PLAN_COST_ALL
- VMQ_SQL_STATS_RULE
- VMQ_SQL_STATS_COST_FIRST
- VMQ_SQL_STATS_COST_ALL

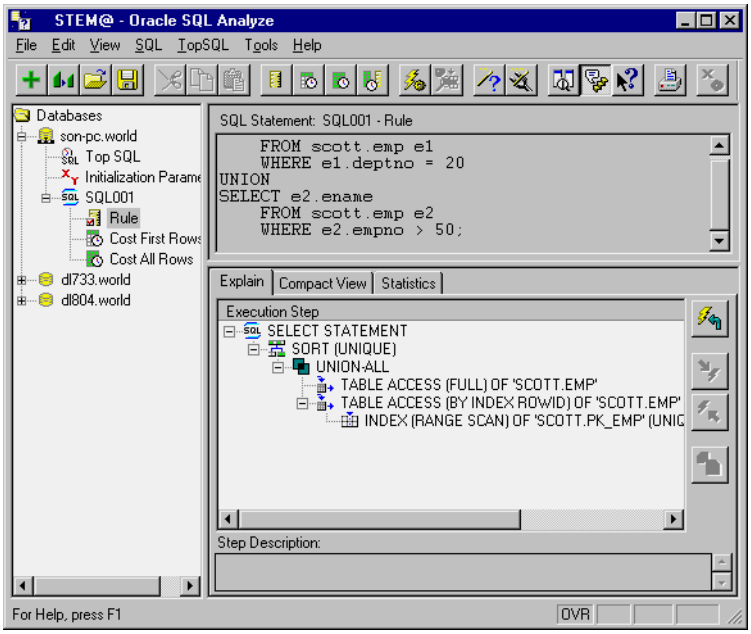
# Oracle SQL Analyze メイン・ウィンドウ

メイン・ウィンドウは、SQL 文の作成およびチューニングを行うための基本作業領域です。このウィンドウには、作業対象のデータベース、セッションおよび SQL 文に加えて、SQL 文の EXPLAIN PLAN が表示されます。

図 2–1 に示すように、メイン・ウィンドウは通常 3 つのペインに分かれています。

- 「ナビゲータ」ウィンドウは、利用可能なノード、オブジェクトおよび SQL 文のツリー状のリストを表示。
- 「SQL テキスト」ウィンドウは、選択された SQL 文のリストを表示。
- 「詳細」ウィンドウは、選択された文についての情報を表示。これには EXPLAIN PLAN、オブジェクト詳細およびパフォーマンス統計が含まれます。

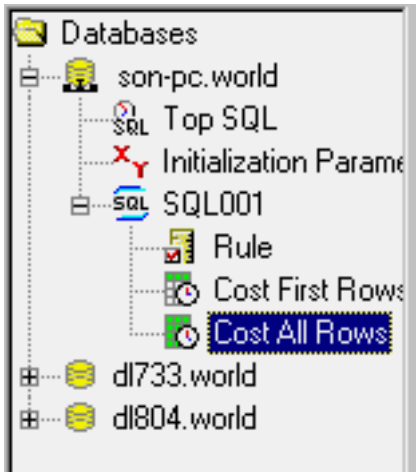
**図 2–1　メイン・ウィンドウ**



# 「ナビゲータ」ウィンドウ

「ナビゲータ」ウィンドウからは、Oracle Enterprise Manager インテリジェント・エージェントによって現在検出されているノードにアクセスできます。「ナビゲータ」ウィンドウは常に表示されており、図 2–2 のように示されます。

**図 2–2 「ナビゲータ」ウィンドウ**




"+" 記号をクリックすると、そのデータベースに関連した、TopSQL オブジェクト、初期化パラメータ・オブジェクト、SQL 文オブジェクトおよび EXPLAIN PLAN オブジェクトが表示されます。

Oracle Enterprise Manager に表示されるものと同様に、データベース・ノードが最上位に表示されます。

---

**注意：** Oracle Enterprise Manager から切断されたノードは、関連付けられた SQL 文オブジェクトが存在する限り、ナビゲーション・ツリーに表示され続けます。切断されたノードを削除するには、接続された SQL 文オブジェクトを削除し、次に Oracle SQL Analyze をいったん終了して、再起動します。

---

オブジェクトの接続階層は、次のようになります。

1. TopSQL オブジェクト
2. 初期化パラメータ・オブジェクト
3. SQL 文オブジェクト
4. EXPLAIN PLAN オブジェクト

TopSQL オブジェクトをクリックすると、TopSQL のフィルタ操作がアクティブになります。この操作により、V$SQLAREA に格納された SQL 文を、それらが消費するリソースに従ってソートできます。

初期化パラメータ・オブジェクトをクリックすると、インスタンス・パラメータが表示されます。このパラメータを編集して、さまざまなデータベース環境をシミュレートできます。

SQL 文オブジェクトには、あるバージョン固有の構文の SQL 文が 1 つ含れています。このオブジェクトをクリックすると、「SQL テキスト」ウィンドウに文が表示されます。

EXPLAIN PLAN オブジェクトには、SQL 文に対して生成された単一の EXPLAIN PLAN が含まれます。Oracle SQL Analyze では、すべての SQL 文に対して、ルール・ベースの EXPLAIN PLAN を生成できます。さらに、ANALYZE コマンドによって分析済みの SQL 文に対して、コスト・ベースの EXPLAIN PLAN を生成できます。

## 「SQL テキスト」ウィンドウ

図 2–3 に示すように、「SQL テキスト」ウィンドウには、現在分析中の SQL 文が表示されます。

このウィンドウでは、文の編集またはデータベース・ビューの検証を実行できます。

**図 2–3 「SQL テキスト」ウィンドウ**

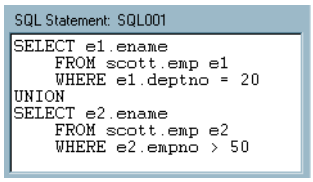



図 2–4 に示すように、TopSQL を使っているとき、リソースの消費順にソートされた、複数の SQL 文が「SQL テキスト」ウィンドウに表示されます。これらの文を「ナビゲータ」ウィンドウにドラッグすることにより、SQL 文オブジェクトを作成できます。

**図 2–4　TopSQL 使用時の「SQL テキスト」ウィンドウ**

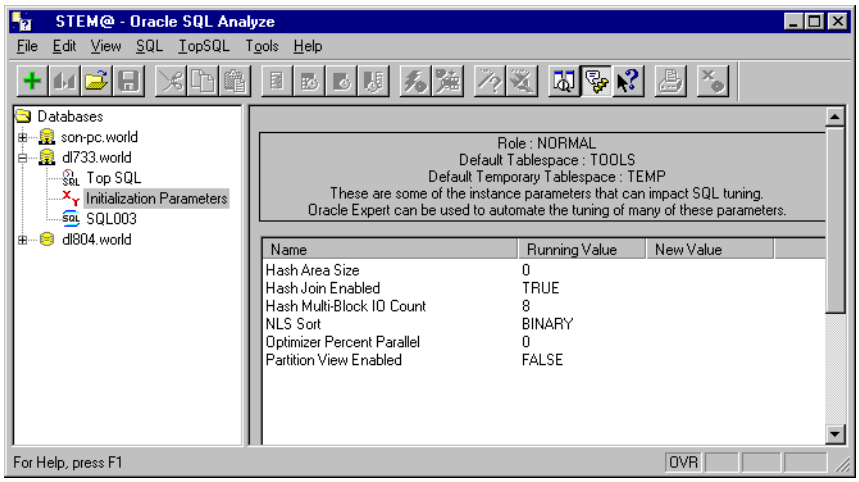
| SQL Text | Disk Reads | Buffer Gets | Executions | Sorts |
|---|---|---|---|---|
| SELECT DATA FROM SMP_BLOB WHERE ID = ... | 1168 | 1199 | 7 | 0 |
| SELECT 'TRUE' FROM SYS.USER_TABLES W... | 98 | 1159 | 98 | 0 |
| SELECT USER FROM DUAL | 38 | 206 | 21 | 0 |
| SELECT 'TRUE' FROM SYS.USER_TABLES W... | 32 | 80 | 7 | 0 |
| Select DISTINCT ji.id, name, type, status, node, d... | 30 | 97 | 7 | 1 |
| DELETE FROM SMP_BLOB WHERE ID = 'ui' AN... | 25 | 74 | 6 | 0 |

# 「詳細」ウィンドウ

「詳細」ウィンドウには、分析の対象物についての情報が表示されます。このウィンドウの外観サイズは、実行されている操作の種類に依存します。

たとえば、テキスト・ウィンドウの文をベースにして EXPLAIN PLAN を作成する場合、図 2–1 に示すように、「詳細」ウィンドウは、「ナビゲータ」ウィンドウの左側、「SQL テキスト」ウィンドウの下側のスペースを占めます。データベースの初期化パラメータを検査する場合、図 2–5 に示すように、「詳細」ウィンドウは、「ナビゲータ」ウィンドウの右側のスペースすべてを占めます。

**図 2–5　初期化パラメータを示す「詳細」ウィンドウ**



# チューニング対象の文の選択

SQL のチューニング・セッションを開始する方法は複数ありますが、最も一般的なシナリオとして、システムにボトルネックを作り出している既存の SQL 文を識別することが考えられます。チューニングされた場合、パフォーマンスが向上する可能性が最も高い文には、次のものがあります。

- 全体のリソースを非常に多く消費する文
- 1 行ごと（または 1 つの実行ごと）に非常に多くのリソースを消費する文
- 実行頻度の最も高い文

TopSQL 機能を使うことにより、V$SQLAREA ビューに位置する文（データベースに対して既に実行されているか、実行の準備ができている文）を、リソースの消費順にソートできます。TopSQL については、次の項の「TopSQL を使った文の選択」で説明します。

チューニング対象の文を選択する他の方法には、次のものがあります。

- 新しい SQL 文を入力（2-14 ページで説明）
- SQL ファイルから文をインポート（2-14 ページで説明）
- 以前に使ったチューニング・セッションを開く（2-14 ページで説明）

## TopSQL を使った文の選択

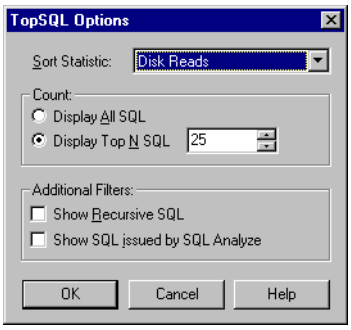
TopSQL では、消費するリソースをもとに、データベース上で使われている SQL 文を検査します。このリストの統計を使って、どの文が最も多くのリソースを消費しているかを判断し、その文をチューニングの対象として選択できます。

TopSQL は、その統計を V$SQLAREA ビューから取得します。V$SQLAREA ビューには、共有 SQL 領域上の統計がリスト表示されます。また、メモリ上にあり、解析済みで、実行の準備ができている SQL 文、あるいは既に実行済みの SQL 文についての統計が表示されます。

### TopSQL 分析の開始

1. 図 2–2 に示すように、「ナビゲータ」ウィンドウで TopSQL オブジェクトをクリックします。図 2–6 に示すように、「TopSQL オプション」ダイアログ・ボックスが表示されます。

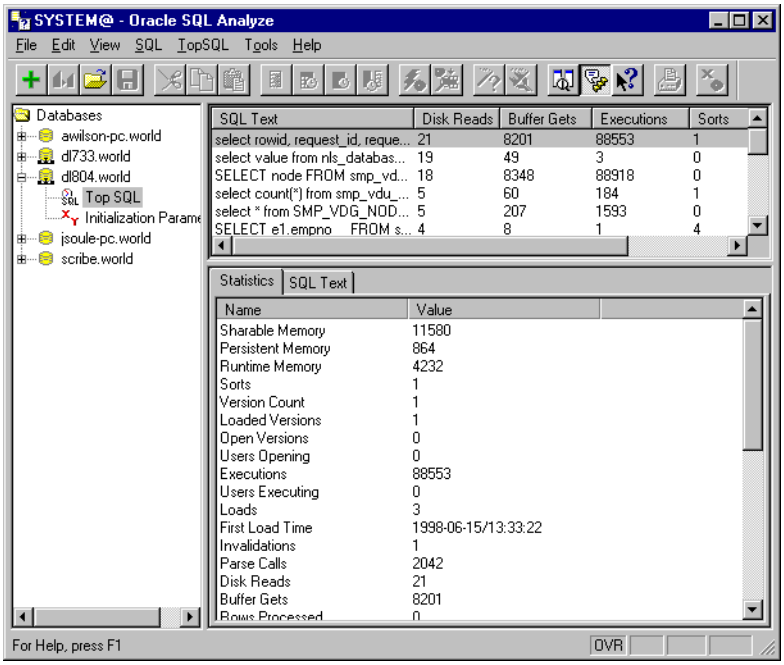
**図 2–6 「TopSQL オプション」ダイアログ・ボックス**




2. 「**ソート統計**」フィールドで、消費量を測定するリソースを選択します。
3. 「件数」領域で、表示する SQL 文の数を選択します。
4. 「その他のフィルタ」領域で、表示する SQL のタイプを選択します。
5. 「OK」をクリックします。

   TopSQL ビューに、リソースの消費順に文が表示されます。
6. チューニングする SQL 文を選択し、「ナビゲータ」ウィンドウにドラッグして、データベース・ノード上にドロップします。
7. 新しい SQL オブジェクトが作成されます。これで、Oracle SQL Analyze でこの文をチューニングするための準備が整いました。

**図 2–7　TopSQL ビュー**




TopSQL では、次のデータベース・リソースの使用量に基づいて文をソートできます。これらのリソースは、パフォーマンスに最も大きく影響するものです。

### バッファ・キャッシュ・ヒット率
Oracle が必要とするデータ・ブロックが、既にメモリ上に存在している確率のことです。ヒット率が 100% に近ければ近いほど、システムのパフォーマンスは高くなります。ヒット率を高めるには、共有プールのサイズをチューニングし、バッファ・キャッシュの値を大きくします。

### バッファ読取り
すべてのカーソルに対するバッファ読取りの数です。この値は CPU 使用量の測定値を表します。バッファ読取りが多すぎる場合、文をより詳しく検証しなければならない場合があります。

### 実行
その文が実行された回数です。

### 1 実行ごとの CPU 使用量
この統計は、1 実行ごとのバッファ読取りの平均数を示します。バッファ読取りが多すぎる場合、文をより詳しく検証しなければならない場合があります。

**1 行ごとの CPU 使用量**
1 実行ごとのバッファ読取りの数です。

**ディスク読込み**
すべてのカーソルに対するディスク読込みの数です。

**1 実行ごとのディスク読込み**
この統計は、文が実行された回数を取得し、1 実行ごとのディスク読込みの数を計算します。ディスク読込みが多すぎる場合、文をより詳しく検証しなければならない場合があります。

**実行**
オブジェクトがライブラリ・キャッシュ内に送られてから、そのオブジェクト上で発生した実行の数です。

**解析呼出し**
すべてのカーソルに対する解析呼出しの数です。

**1 実行ごとの解析呼出し**
SQL 文が 1 実行ごとに解析された回数です。SQL 文は 1 度だけ解析され、複数回実行されるのが理想的ですが、フロントエンド・アプリケーションの中には、アプリケーションの実行ごとに文を再解析するものがあります。この比率は 0 に近いのが理想的です。この率が 1 以上であることは、不必要な解析呼出しが発生していることを示します。

**処理された行**
解析された SQL 文が返す行の合計数です。文の目的によっては、処理される行が予想より多い、あるいは少ない場合があります。この場合、文をより詳しく検証する必要があります。

**ソート**
すべてのカーソルに対して実行されたソートの数です。ソートの数が多すぎることは、索引または構文の使い方が非効率的であり、それらの最適化が必要であることを示している場合があります。

---

**注意：** パフォーマンスに影響する最も重要な要素は、実行、ディスク読込み、ソートおよびバッファ読取りです。これらの統計は、すべての SQL 文に対してメイン・ウィンドウの上部のペインに示されます。

---

以上に示した統計に加えて、次の統計が下部のペインに表示されます。

- 共有可能メモリー
- 永続メモリー
- ランタイム・メモリー
- バージョン・カウント
- ロードしたバージョン
- オープンしているバージョン
- ユーザーによるオープン
- ユーザーによる実行
- ロード
- 無効化
- ディスク読込み
- コマンド・タイプ
- オプティマイザ・モード
- ユーザー ID の解析中
- スキーマ ID の解析中
- 保存バージョン
- アドレス
- ハッシュ値
- モジュール
- モジュール・ハッシュ
- アクション
- アクション・ハッシュ
- 連続発生可能な異常終了

V$SQLAREA 統計の完全なリストを読むことにより、文のパフォーマンスに関する十分な理解が得られます。また、どの文をチューニングする必要があるか、パフォーマンスに関するどの問題を処理する必要があるかを特定できます。統計については、『Oracle Server リファレンス』および『Oracle Server チューニング』でより詳しく説明します。

## 新しい文の入力

新しい SQL 文を入力するには、**「SQL」=>「新規作成」**を選択します。次に、メイン・ウィンドウの右上の「SQL 文」ウィンドウで新しい文を入力します。

## SQL ファイルからの文のインポート

既存の SQL 文を Oracle SQL Analyze にインポートまたはコピーするには、次の操作を行います。

SQL スクリプトからの場合

**「ファイル」=>「オープン」**を選択して、SQL スクリプトをオープンします。ダイアログ・ボックスが表示されるので、目的の SQL スクリプトを選択します。

TopSQL からの場合

「SQL テキスト」ウィンドウから、セッション・オブジェクトまたは SQL オブジェクト上に、目的の SQL 文をドラッグします。

## 以前に使ったチューニング・セッションのオープン

以前に使ったチューニング・セッションをオープンするには、目的の SQL 文オブジェクトまたは EXPLAIN PLAN オブジェクトをクリックします。

## 印刷

Oracle SQL Analyze では、SQL 文、EXPLAIN PLAN、文および計画の統計データなどを印刷できます。

SQL 文とそのパフォーマンス統計を印刷するには、「ナビゲータ」ウィンドウで SQL オブジェクトを選択し、次に**「ファイル」=>「印刷」**を選択します。

SQL 文、その EXPLAIN PLAN、およびその EXPLAIN PLAN のパフォーマンス統計を印刷するには、「ナビゲータ」ウィンドウで EXPLAIN PLAN オブジェクトを選択し、次に**「ファイル」=>「印刷」**を選択します。

## 保存

SQL 文をファイルに保存するには、**「ファイル」=>「SQL を別名保存」**を選択します。

現在のチューニング・セッションを保存するには、**「ファイル」=>「リポジトリに保存」**を選択します。

# 3
# 情報の収集と分析

この章では、次の項目について説明します。



## 統計情報の理解

Oracle SQL Analyze では、チューニング作業に不可欠な情報が提供されます。

" オプティマイザ・モード " または " ソート領域サイズ " などのデータベース環境に関する情報は、文に対する EXPLAIN PLAN を生成するときに Oracle オプティマイザが下す決定や、文が実行されるときの操作の効率性に影響します。Oracle SQL Analyze では、これらのパラメータの値が次の 2 つの場所に示されます。

- データベース・パラメータ・ビューには、Oracle SQL Analyze の内部で変更できないパラメータが示されます。
- 初期化パラメータ・ビューには、値を編集できるパラメータが示されます。これらのパラメータを編集して、さまざまなデータベース環境をシミュレートすることにより、チューニングに関するさまざまなシナリオをテストできます。

データベースの環境情報については、3-2 ページで説明します。初期化パラメータについては、3-7 ページで説明します。

ビュー、表、索引、クラスタなど、データベース内部の多くの論理構成体を検証することもできます。これらのオブジェクトは、情報の管理を容易にし、データ・アクセスの効率を高めるために作成されるものですが、オブジェクトの使用方法が適切でなければ効果は発揮されません。Oracle SQL Analyze によって提供されるオブジェクト詳細は、オブジェクトがメモリーおよびその他のリソースを効率的に使っているか、オブジェクトが配置している情報

がデータベースの使われ方と常に一致しているか、オプティマイザがこれらのオブジェクトを十分に利用しているか、あるいは無視しているのか、などを判断する際に役立ちます。利用可能なオブジェクト詳細とそれらの意味については、3-12 ページで説明します。

当然のことながら、SQL 文の効率性を計る最大の指標はパフォーマンス統計です。Oracle SQL Analyze では、さまざまな EXPLAIN PLAN を使って文を実行し、それらのパフォーマンスを比較できます。EXPLAIN PLAN、パフォーマンス統計およびそれらの分析については、3-26 ページで説明します。

# データベース環境の分析

データベースが起動されるたびに、システム・グローバル領域（SGA）が割り当てられ、Oracle のバックグラウンド・プロセスが起動されます。システム・グローバル領域とは、データベースのユーザーによって共有されるデータベース情報の格納に使われるメモリー領域のことです。バックグラウンド・プロセスとメモリー・バッファの組み合わせを Oracle インスタンスと呼びます。

Oracle SQL Analyze を使って、データベース・パラメータと、インスタンスの初期化パラメータを検証できます。これは、次の項で説明します。

## データベース・パラメータ

「データベース・パラメータ」ダイアログ・ボックスに表示されるデータベース・パラメータは、メモリーおよびディスクのパフォーマンスに影響します。Oracle SQL Analyze からは、これらの値を編集できません。しかし、これらのパラメータが持つ影響を理解しておくと、Oracle Expert を使って値をチューニングできます。

---

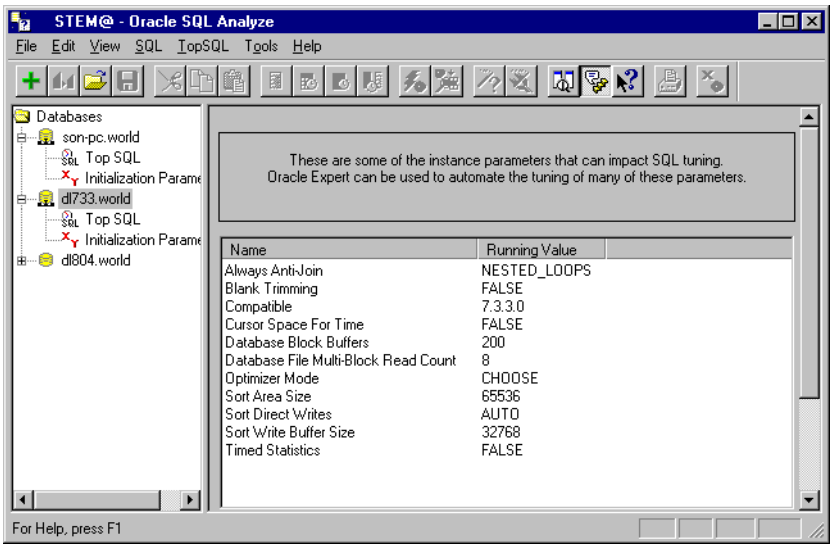
**注意：** データベースおよび初期化パラメータ、そのチューニングの意味に関する次に説明する情報について、ここではその概要だけを説明します。詳細は、『Oracle Server 管理者ガイド』、『Oracle Server チューニング』および『Oracle Server リファレンス』を参照してください。

---

### データベース・パラメータ・ビューのオープン

データベース・パラメータ・ビューをオープンするには、次に示すように「ナビゲータ」ウィンドウでデータベース・ノードをクリックします。

**図 3–1　データベース・パラメータ・ビュー**




右側の「詳細」ウィンドウには、次の統計が表示されます。

## 常に結合不可

**説明**　Oracle Server が使う、結合不可の型を設定します。NESTED_LOOPS、MERGE または HASH から選択します。システムは、結合不可の実行が正当かどうかを検証し、正当であれば、このパラメータの値に従って副問合せを処理します。デフォルトの設定は NESTED_LOOPS です。

**チューニング上の考慮事項**「常に結合不可」パラメータは、コストベース最適化において、NOT IN 句のパラレル処理を最も効率的に利用するために役立ちます。

このパラメータの値を HASH に設定すると、NOT IN 句が最も効率的に処理されます。このときパラレル・ハッシュ結合不可が使われ、NOT IN 演算子はパラレルに評価されます。このパラメータが HASH に設定されていない場合、NOT IN は（連続的な）相関副問合せとして評価されます。

パラメータが NESTED LOOPS に設定されると、NOT IN 句の処理効率は最も低くなります。

データ・ウェアハウジング・アプリケーションにおいては、しばしばパラメータをこの値に設定する必要があります。

## ビットマップ・マージ領域サイズ

**説明** 索引の範囲走査によって検索されたビットマップをマージするために使われる、メモリー容量を指定します。

**チューニング上の考慮事項** このパラメータのデフォルト値は 1MB です。より大きな値を指定すると、オプティマイザが索引をより頻繁にビットマップするようになるため、多くの場合パフォーマンスが向上します。

## ブランク切捨て

**説明** 文字データ型のデータ割当て方法を指定します。

**チューニング上の考慮事項** このパラメータの値が TRUE のとき、ソースのデータ長が宛先よりも長い場合でも、ソース文字列 / 変数を宛先の文字列 / 変数に割り当てられます。ただし、この場合、宛先のデータ長を超えたデータはすべて空白になります。パラメータが FALSE のとき、ソースのデータ長が宛先を超える場合のデータ割当ては許可されず、SQL92 エントリ・レベルのセマンティクスに戻されます。

## 互換性

**説明** Oracle Server が互換性を維持しなければならないリリースを指定します。デフォルト値は、互換性を保証できる最新のリリースです。

**チューニング上の考慮事項** このパラメータを使うことにより、新リリースにおけるメンテナンス面の改良点を、使用している環境で新機能をテストすることなく、既存のシステム上で直ちに利用できます。また、以前のリリースに戻る必要があるときの備えとして、このパラメータを使って以前のリリースとの下位互換性を保ちながら、新しいリリースを使えます。

このパラメータを以前のリリースに設定すると、現行リリースの機能が一部制限される、あるいは無効とされる場合があります。最新のパフォーマンス機能の十分な効果を得るには、このパラメータが現行のリリースと等しい値に設定されていることを確認します。

## 時間のカーソル領域

**説明** 共有 SQL 領域の割当てをライブラリ・キャッシュから解除して、新しい SQL 文を格納する余地を作り出すタイミングを指定します。このパラメータのデフォルト値は FALSE です。

**チューニング上の考慮事項** この値が FALSE の場合、その SQL 文に関連付けられたアプリケーション・カーソルがオープンしているかどうかに関係なく、共有 SQL 領域の割当てをライブラリ・キャッシュから解除できます。この場合、その SQL 文が含まれる共有 SQL 領

域がライブラリ・キャッシュ内にあることが、Oracle によって検証されなければなりません。

このパラメータの値を TRUE に設定すると、時間が少し節約され、実行コールのパフォーマンスがわずかながら改善される場合があります。パラメータの値が TRUE の場合、その文に関連付けられているアプリケーション・カーソルがすべてクローズされているときに限り、共有 SQL 領域の割当てを解除できます。この場合、共有 SQL 領域は、その領域に関連付けられたアプリケーション・カーソルがオープンしている限り割当て解除されないため、共有 SQL 領域がキャッシュ内にあることが Oracle によって検証される必要はありません。

次のような場合には、この値を TRUE に設定しないでください。

- 実行コール上にライブラリ・キャッシュ・ミスを発見した場合。そのようなライブラリ・キャッシュ・ミスの発生は、共有プールの大きさが十分でなく、同時にオープンしている、すべてのカーソルの共有 SQL 領域を保持できないことを示しています。
- 各ユーザーがプライベート SQL 領域のために使用可能なメモリー容量が不足している場合。同時にオープンしているすべてのカーソルのためのプライベート SQL 領域が、ユーザーが使用可能なメモリー領域を満たしてしまい、新しい SQL 文のためにプライベート SQL 領域を割り当てる空間がない場合、その文は解析できず、メモリー不足を示すエラーが Oracle によって返されます。

詳細は、『Oracle8 Server 概要』を参照してください。

## データベース・ブロック・バッファ

**説明** このパラメータは、システム・グローバル領域（SGA）のバッファ・キャッシュ内のバッファ数を定義するために使われます。個々のバッファ・プールは、デフォルトのバッファ・プールに割り当てられた残りの容量をこの数値で割って作成されます。

**チューニング上の考慮事項** バッファの数は、キャッシュのパフォーマンスに影響します。キャッシュ・サイズが大きいほど、修正されたデータがディスクに書き込まれる回数は減ります。ただし、キャッシュを大きくするとメモリーの消費も大きくなり、その結果メモリーのページングまたはスワッピングが発生する場合があります。

データベース・ブロック・バッファのパラメータは、データベース・ブロック・サイズのパラメータとともにバッファ・キャッシュの合計サイズを決定します。バッファ・キャッシュを効率的に使うことにより、データベース上の I/O 負荷を大幅に低減できます。データベース・ブロック・サイズはデータベースを最初に作成するときにしか指定できないため、バッファ・キャッシュのサイズ制御にはデータベース・ブロック・バッファを使います。

詳細は、『Oracle Server 概要』を参照してください。また、デフォルト値については、使用しているオペレーティング・システムに固有の Oracle ドキュメンテーションを参照してください。

## データベース・バッファ・キャッシュ

**説明** SGA のデータベース・バッファには、データベース・データの最も新しく使われたブロックが格納されます。インスタンス内のデータベース・バッファの集合をデータベース・バッファ・キャッシュといいます。バッファ・キャッシュには、未修正のブロックだけでなく修正済みのブロックも格納されます。最も新しく使われたデータ（ほとんどの場合は最も頻繁に使われるデータ）がメモリー上に保持されるため、必要なディスク I/O が減り、パフォーマンスが向上します。

**チューニング上の考慮事項** Oracle のユーザー・プロセスがデータに最初にアクセスするとき、プロセスはデータへのアクセスに先立って、データをディスクからバッファ・キャッシュにコピーしなければなりません。これをキャッシュ・ミスと呼びます。既にキャッシュ内にあるデータにプロセスがアクセスするとき、プロセスはデータをメモリーから直接読み出します。これをキャッシュ・ヒットと呼びます。キャッシュ・ヒットを通じたデータへのアクセスは、キャッシュ・ミスを通じたデータ・アクセスよりも高速です。

キャッシュのサイズは、要求されたデータがキャッシュ上にヒットする確率に影響します。キャッシュが大きい場合、要求されたデータがキャッシュに含まれている確率は高くなります。キャッシュのサイズを大きくすると、キャッシュにヒットするデータ要求の割合が増します。ただし、キャッシュが大きすぎることにより、過度のスワッピングおよびページングが発生する場合があります。

## データベース・ファイル・マルチブロック読込みカウント

**説明** このパラメータは、マルチブロック I/O に対して使われ、逐次走査の間に 1 回の I/O 操作で読み込むことのできる最大ブロック数を指定します。

**チューニング上の考慮事項** 全表走査の実行に必要な I/O 総数は、次の要素によって決まります。

- 表のサイズ
- マルチブロック読込みカウント
- 操作にパラレル問合せが利用されているかどうか

通常、このパラメータに大きな値を指定すると、表走査のコストが下がります。この設定は、索引上の表走査に適しています。デフォルト値は 8 です。OLTP 環境およびバッチ環境では通常、このパラメータの値を 4 ～ 16 の範囲内に設定します。DSS データベース環境では、このパラメータに上限値を指定することにより、最大の効果が得られる傾向があります。

### オプティマイザ・モード

**説明** このパラメータは、インスタンス起動時のオプティマイザのモードを設定します。ルールベース、スループット優先のコストベース、応答時間優先のコストベース、または統計の存否に応じた選択ベースの中から指定します。

**チューニング上の考慮事項** このパラメータは、オプティマイザのデフォルトの動作を指定します。ほとんどの場合、コストベース最適化の方が、ルールベース最適化よりも良い結果をもたらします。Oracle SQL Analyze では、文が ANALYZE SQL コマンドによって分析済みの場合、4 種類のオプティマイザ・モードのすべてを使って文をテストできます。ヒントを使うと、デフォルトのパラメータを上書きできます。

オプティマイザの詳細は、『Oracle Server 概要』および『Oracle Server チューニング』を参照してください。

## 初期化パラメータ

初期化パラメータ・ビューに表示される初期化パラメータは、メモリーおよびディスクのパフォーマンスに影響します。Oracle SQL Analyze からこれらの値を編集して、さまざまな環境をシミュレートできます。あるいは、これらのパラメータがデータベース・パフォーマンスに及ぼす影響をテストできます。

### 初期化パラメータの表示

これらのパラメータを表示するには、「ナビゲータ」ウィンドウから目的の初期化パラメータオブジェクトを選択します。「詳細」ウィンドウには、初期化セッション・パラメータ、パラメータの現在の実行値、およびユーザーが設定する新しい値（存在する場合）が表示されます。

## 初期化パラメータの編集

初期化パラメータの設定を編集するには、次の操作を行います。

1. ツリー・メニューから目的のセッション・オブジェクトを選択します。

   右ウィンドウにウィンドウが開き、データベース・パラメータ、パラメータの現在の実行値、およびユーザーが設定する新しい値（存在する場合）が表示されます。

2. 変更する値をダブルクリックします。

   ダイアログ・ボックスが表示されます。ダイアログ・ボックスの外観は、数値またはブール値のどちらを変更するかによって異なります。

3. 数値を変更する場合、「**値**」フィールドに新しい値を入力します。

   値がブール値（TRUE、FALSE または AUTIO）の場合、適切なラジオ・ボタンを選択します。

> **注意：** ここで行う変更は、データベース自体でなく、その時点で選択されているチューニング・セッションだけに影響します。

次の初期化パラメータが使用可能です。

## ハッシュ領域サイズ

**説明** ハッシュ結合に使うメモリーの最大容量をバイト単位で指定します。

**チューニング上の考慮事項** この値を大きくすると、ハッシュ結合のコストが下がるため、オプティマイザがハッシュ結合を選択する確率が高くなります。大きすぎる値を指定すると、システムがメモリー不足を起こす場合があります。このパラメータが設定されない場合、ソート領域サイズの 2 倍の値がデフォルト値として使われます。

推奨値は、結合操作への入力の小さい方のサイズ（単位は MB）を S とした場合、S の平方根の約半分です。この値を 1MB 未満に設定しないでください。

## ハッシュ結合可能

**説明** ハッシュ結合機能を使用可能または使用不可にします。

**チューニング上の考慮事項** このパラメータは、オプティマイザが結合メソッドとしてハッシュ結合の使用を考慮すべきかどうかを指定します。FALSE を指定すると、ハッシュ結合は無効とされ、オプティマイザが選択を考慮できる結合メソッドとして利用できなくなります。TRUE を指定すると、オプティマイザはハッシュ結合のコストをその他の結合と比較し、ハッシュ結合のコストが最善と判断した場合にそれを選択します。データ・ウェアハウジング・アプリケーションに対しては、このパラメータを常に TRUE に設定してください。

## ハッシュ・マルチブロック I/O カウント

**説明** ハッシュ結合が I/O の際に、いくつのブロックを連続して読み書きするかを指定します。

**チューニング上の考慮事項** このパラメータは、入力が分割されるパーティション数を制御するため、パフォーマンスに大きく影響します。この値を大きくすると、ハッシュ結合のコストが下がるため、より多くのハッシュ結合が行われます。

このパラメータを変更する必要はほとんどありません。このパラメータを変更する場合は、次の式が成立することを確認してください。

```
R/M <= Po2(M/C)
```

R = ( 結合への左入力 ) のサイズ、M = ( ハッシュ領域サイズ )*0.9、Po2(n)=n 未満である 2 の最大の累乗、C = ( ハッシュ・マルチブロック I/O カウント )*( データベース・ブロック・サイズ )

## NLS ソート

**説明** ORDER BY 問合せの照合順番を指定します。

**チューニング上の考慮事項** この値が BINARY の場合、ORDER BY 問合せの照合順番は文字の数値に基づきます（システムのオーバーヘッドが小さいバイナリ・ソート）。値が名前付きの言語ソートの場合、ソーティングは定義済み言語ソートの順序に基づきます。NLS_LANGUAGE パラメータによってサポートされる言語のほとんどは、同名での言語ソートもサポートします。

NLS ソートの値を BINARY 以外に設定すると、オプティマイザによって選択されたパスに関係なく、ソートは全表走査を使います。索引はキーのバイナリ順に従って構築されるため、BINARY は例外です。このため、NLS ソートが BINARY に設定されているとき、オプティマイザは索引を使って、ORDER BY 句の要求を満たせます。NLS_SORT がいずれかの言語ソートに設定されている場合、オプティマイザは実行計画に全表走査および完全ソートを含める必要があります。

このパラメータのデフォルト値は、NLS_LANGUAGE パラメータの値に応じて変化します。

このパラメータの詳細は、『Oracle Server 管理者ガイド』を参照してください。

## オプティマイザ・パーセント・パラレル

**説明** オプティマイザがどの程度積極的に、与えられた実行計画のパラレル化を試みるかを決定します。デフォルトの 0 は、オプティマイザが最善のシリアル計画を選択することを意味します。この値が 100 の場合、オプティマイザは各オブジェクトの並行度を使って、全表走査のコストを計算します。

**チューニング上の考慮事項** 小さい値は索引走査に、大きい値は表走査に適しています。

## オプティマイザ検索制限

**説明** 考えられるすべての結合組合せが考慮される、FOM 句内の表の最大数です。

**チューニング上の考慮事項** このパラメータは、オプティマイザの検索範囲を指定します。推奨値は、100/number_of_concurrent_users です。

## パーティション・ビュー使用可

**説明** パーティション・ビューを使用可にします。

**チューニング上の考慮事項** データ・ウェアハウジング・アプリケーションにおいては、しばしばこのパラメータを設定する必要があります。パーティション・ビュー使用可が TRUE に設定されていると、オプティマイザはパーティション・ビューにおける不必要な表アクセスを切り詰めます（あるいは、スキップします）。このパラメータは、コストベース・オプティマイザが、ビューの基礎を形成する表の上での統計から、パーティション・ビュー上の統計を計算する方法にも変化を与えます。

## ソート領域サイズ

**説明** ソートのために使われるプログラム・グローバル領域（PGA）メモリーの最大容量をバイト単位で指定します。

**チューニング上の考慮事項** システムに搭載されているメモリーが少ない場合、ソート領域サイズに大きな値を設定することが有効です。この値を大きくすると、操作全体がメモリー内で実行される確率が高くなるため、ハッシュ操作および大規模なソートのパフォーマンスを大幅に向上できます。

ソート領域が小さすぎると、データは小さな断片に分割され、それぞれの断片または実行が個別にソートされます。I/O の量があまりに多くなると、実行を 1 回のソートにマージし直

す必要が生じます。ソート領域サイズが非常に小さい場合、マージしなければならない実行が多くなり、複数のパスが必要になる場合があります。ソート領域サイズが減少するにつれて、I/O の量は増加します。

ソート領域が大きすぎると、オペレーティング・システムのページング率が過度に増加してしまいます。各パラレル・サーバーは、各ソートに対してこのメモリー容量を割り当てることができるため、累積されたソート領域は急激に増加します。

システムのメモリー容量が少ない場合、ソートおよびハッシュ操作に割り当てられるメモリー容量を制限できます。そのかわり、一時ソート・セグメントからのデータ・ブロックをバッファ・キャッシュにキャッシュできるように、バッファ・キャッシュのサイズを大きめに確保してください。

## ダイレクト書込みソート

**説明** このパラメータは、ソート・データがバッファ・キャッシュを回避して、ソート結果をディスクに直接書き込むかどうかを制御します。

**チューニング上の考慮事項** システム上でメモリーおよび一時スペースが使用可能な場合、ダイレクト書込みソートによってソートのパフォーマンスを改善できます。

この値を TRUE に設定すると、各ソートの間にメモリーから追加のバッファが割り当てられます。これによりソートのコストが低減され、オプティマイザはより多くのソート結合を使うようになります。

この値がデフォルトの AUTO に設定されており、ソート領域サイズの値がブロック・サイズの 10 倍を超えているとき、メモリーはソート領域から割り当てられます。

FALSE に設定すると、ディスクへの書込みを行うソートは、バッファ・キャッシュを通じて書込みを行います。

詳細は、『Oracle Server チューニング』を参照してください。

# 論理構造の分析

表、ビュー、索引、クラスタなどのオブジェクトは、管理を容易にし、データベース内のデータへの高速かつ効率的なアクセスを実現するために作成されます。同じオブジェクトでもその監視を怠ると、肥大しすぎて過度のメモリーを消費してしまうことがあります。また、ユーザーの現在の行動に合わせて設計されていないオブジェクトが、まったく役に立たなくなることもあります。さらに、オブジェクトが古くなって使われなくなることも考えられます。

Oracle SQL Analyze は、ビューとオブジェクトの詳細情報を提供し、表、索引またはクラスタがその効果を発揮しているか、あるいはこれらの論理構造の一部を編集または再作成することを考慮すべきか、などの判断を支援します。

## オブジェクト詳細と統計の表示

SQL Analyze は、EXPLAIN PLAN によって使われる任意の表、索引またはクラスタの詳細を示します。

次の方法によって、これらの統計を EXPLAIN PLAN から表示できます。

1. オブジェクトに関係する EXPLAIN PLAN に含まれるエントリを選択。これは、「オブジェクト名」列でオブジェクトの名前を見つけることによって判断できます。
2. エントリを右クリック。

   選択項目として「**オブジェクト詳細**」を含むメニューが表示されます。
3. 「**オブジェクト詳細**」を選択。

   3-13 ページに示すように、「オブジェクト詳細」ダイアログ・ボックスが表示されます。

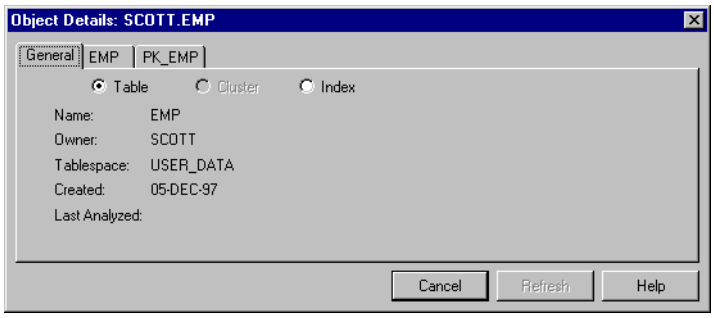
「オブジェクト詳細」ダイアログ・ボックスでは、タブを使って表、索引およびクラスタの統計を参照できます。統計の意味については、次の項で説明します。

---

**注意：** オブジェクト詳細および統計、そのチューニングの意味に関する次に説明する情報について、ここではその概要だけを説明します。詳細は、『Oracle Server 管理者ガイド』、『Oracle Server 概要』、『Oracle Server チューニング』、『Oracle Server リファレンス』、および 『Oracle Server アプリケーション開発者ガイド』を参照してください。

---

**図 3–2 「オブジェクト詳細」ダイアログ・ボックス**




## 一般詳細の表示

オブジェクトについての詳細が表示される一般的な項目には、次のものがあります。

- オブジェクトの名前。
- 所有者。
- 表領域の場所。
- 作成日。
- オブジェクトが ANALYZE コマンドによって最後に分析された日付。この値は、統計が現在のものでないは場合、コストベース・オプティマイザの有効性に影響します。
- オブジェクトの型。「クラスタ」ラジオ・ボタンが選択されている場合、型は INDEX または HASH です。「索引」ラジオ・ボタンが選択されている場合、型は UNIQUE または NON-UNIQUE です。

「索引」タブが選択されている場合、「列」シートには、「**名前**」フィールドで識別された次の索引情報が表示されます。

**名前**
その索引のベースとなっているキー列です。

**位置**
表中での列の位置です。

**型**
キー列のデータ型です。

**NULL 可**
列内の値が、値を必要とするか、空白（NULL）のままでも構わないかの区別です。

# 表詳細

表は、Oracle データベースにおけるデータ記憶の基本ユニットです。表を作成すると、Oracle は表のデータ・セグメントに、指定されたデータ・ブロック数の初期容量を割り当てます。これらのブロックがいっぱいになると、データベースのパフォーマンスは低下します。したがって、次にリストされた統計を監視して、データベースが既存の領域を有効に使っていること、さらに、必要な領域を保持していることの確認が重要です。

表に割り当てられているブロック領域に問題があると判断した場合、次のことを行うために、PCTFREE パラメータおよび PCTUSED パラメータの使用を考慮できます。

- データ・セグメントまたは索引セグメントの書込みおよび検索のパフォーマンスを上げる
- データ・ブロック内の未使用領域を減らす
- データ・ブロック間の行連鎖を減らす

表の管理に関するヒントは、『Oracle Server アプリケーション開発者ガイド』を参照してください。

**表詳細の表示**

1. オブジェクト詳細と統計の表示での説明に従って、「オブジェクト詳細」ダイアログ・ボックスを表示します。
2. 「**表**」ラジオ・ボタンを選択して、EXPLAIN PLAN の選択行によって参照されている表の名前を表示します。
3. 参照される表と同じ名前を持つタブをクリックします。

「表詳細」プロパティのページでは、次の統計をレビューできます。

## エクステント

**説明** 特定の種類の情報を格納するために割り当てられる、特定数の隣接したデータ・ブロックのことです。

**チューニング上の考慮事項** エクステントのサイズを適切に設定することは、全表走査のパフォーマンスを管理する上で鍵となる要素です。エクステントのサイズが適切に設定されていないと、エクステントの数およびサイズが原因で、全表走査の間にデータベースが実行する作業量が著しく増大することがあります。エクステントのサイズは、エクステントごとのデータ・ブロックの数によって判断できます。

大規模な走査（全表走査または大規模な索引範囲走査）を介して頻繁に読み込まれるデータベース・オブジェクトは、少ない数のエクステントに格納してください。エクステントの数を小さく保つことにより、次に読み取られるデータが、現在読み取られているデータに物理的に近くなる可能性が高くなります。

複数のエクステントの、パフォーマンスへの潜在的な影響を回避するには、各エクステントのサイズが、各マルチブロック読込みの間に読み込まれるブロック数の倍数であることを確認する必要があります（データベース・パラメータの項の「データベース・ファイル・マルチブロック読込みカウント」を参照）。多くのシステムでは、1 回の読込みの間に 64KB または 128KB のデータが読み込まれます。したがって、それぞれ 64KB または 128KB の倍数になるように、エクステントのサイズを設定してください。

## 割当てブロック

**説明** 表を作成すると、Oracle は表のデータ・セグメントに、指定された数のデータ・ブロック数の初期エクステントを割り当てます。行はまだ挿入されていませんが、初期エクステントに対応する Oracle のブロックはその表の行のために予約されているか、あるいは割り当てられています。

**チューニング上の考慮事項** エクステントのサイズを指示します。

## 使用ブロック

**説明** 表に既に割り当てられているブロックです。

**チューニング上の考慮事項** エクステントのサイズを指示します。

## 空ブロック

**説明** 表の行において Oracle が使用可能とみなすブロックです。

**チューニング上の考慮事項** エクステントのサイズを指示します。

## 連鎖行

**説明** 行が最初に挿入されるときに、行が大きすぎて 1 データ・ブロックに収まらない場合、Oracle はその行のデータを、そのセグメントのために予約された（1 つまたは複数の）鎖状のデータ・ブロックに格納します。行連鎖は、データ型 LONG または LONG RAW の列を含むような大きな行を扱うときに最も頻繁に発生します。

**チューニング上の考慮事項** 連鎖行は、それらの行に関連付けられた I/O のパフォーマンスを低下させます。

## 行

**説明** 表または索引のデータを含んでいるデータ・ブロックの部分です。この値は、表中の行の数です。

**チューニング上の考慮事項** 表のサイズと、全表走査において走査する必要のある行数を指示します。

### 行の平均の長さ

**説明** 表または索引のデータが含まれるデータ・ブロックの平均サイズをバイト単位で表します。

**チューニング上の考慮事項** 行または索引のエントリへの更新によって、行が大きくなりブロックにまたがる（連鎖行が発生する）と、処理コストが増大します。

### ブロック当りの平均空き領域

**説明** 空きリスト上のすべてのブロックの平均空き領域です。空きリストは、セグメントのエクステントに対して割り当てられているデータベース・ブロックのリストで、PCTFREE の設定よりも大きな空き領域を持っています。

**チューニング上の考慮事項** 表のブロック内部に行が濃密に詰め込まれるほど、読み取る必要のあるブロック数は減ります。データベースの各ブロックは、行のデータによって使われるヘッダ / トレーラ領域と、空きの領域を持ちます。ブロック内部の行の密度を向上させるには、領域管理を行う際に、これら 3 つの領域をすべて考慮する必要があります。

## クラスタ詳細

クラスタは、さまざまな表の関連する行を同一データ・ブロックに格納します。このことからは、2 つの主要な利点が得られます。

- ディスク I/O が減少し、クラスタ化された表の結合にかかるアクセス時間が短縮されます。
- クラスタでは、複数の表のどれだけ多くの行にその値が含まれるかに関係なく、クラスタのキー値（すなわち、互いに関連した値）は一度だけ格納されます。したがって、クラスタにおける、互いに関連した表データの格納には、クラスタ化されていない表形式の場合に比べて、必要な記憶領域が少なくて済む場合があります。

クラスタ化された形式で格納する方が適しているデータを識別するには、参照整合性制約を通じて互いに関連する表を探すか、結合を使って頻繁に同時アクセスされる表を探します。表データの結合に使われる列上の表をクラスタ化すると、問合せを処理するためにアクセスする必要があるデータ・ブロックの数が減少します。クラスタ・キー上の結合に必要なすべての行は、同一ブロック内にあります。

逆に言えば、クラスタ・キーに対するすべての行が単一のブロック内に収まっていない場合、結合文によって消費されるリソースが増加する場合があります。

また、クラスタを使用する場合、表をそれ自体の索引とともに個別に格納するときに比べて、DML 文（INSERT、UPDATE および DELETE）のパフォーマンスが低下する場合がある

ので注意が必要です。この不利益は、領域の使用率と、表を走査するためにアクセスする必要があるブロック数に関係します。複数の表が各ブロックを共有するため、同じ表がクラスタ化されずに格納された場合に比べて、クラスタ化された表の格納にはより多くのブロックが必要です。クラスタを使うかどうかを判断する際には、これらのトレードオフに留意する必要があります。

クラスタ管理の詳細は、『Oracle Server 概要』および『Oracle Server アプリケーション開発者ガイド』を参照してください。

**クラスタ詳細の表示**

1. オブジェクト詳細と統計の表示で説明するようにして、「オブジェクト詳細」ダイアログ・ボックスを表示します。
2. 「**クラスタ**」ラジオ・ボタンを選択して、EXPLAIN PLAN の選択行によって参照されているクラスタの名前を表示します。
3. 参照されるクラスタと同じ名前を持つタブをクリックします。

「クラスタ統計」ダイアログ・ボックスに次の統計が表示されます。

## エクステント

**定義** 特定の種類の情報を格納するために割り当てられる、特定数の隣接したデータ・ブロックのことです。

**チューニング上の考慮事項** 3-14 ページの、表のエクステントに関する説明を参照してください。

## 割当てブロック

**説明** クラスタ・キーと、それらのキーに関連付けられた行を格納するために割り当てられるブロックです。

**チューニング上の考慮事項** クラスタのサイズを指示します。

## クラスタ・キー当りの平均ブロック

**説明** 表中のブロック数を、ハッシュ・キーの数で割ることによって得られる値です。

**チューニング上の考慮事項** デフォルトでは、Oracle は 1 つのクラスタ・キーとそのキーに関連付けられた行だけを、クラスタのデータ・セグメントの各データ・ブロックに格納します。クラスタ・キー値に対するすべての行が 1 ブロックに収まらない場合、そのキーの内部のすべての値へのアクセスを高速化するために、ブロックが互いに連結されます。クラスタ・キー当りの行が少なすぎると、領域が浪費され、パフォーマンスがごくわずかしか向上

しない場合があります。行が多すぎると、そのキーに対応する行を見つけるために、オプティマイザによって過度の検索が行われる場合があります。

## 空ブロック

**説明** クラスタ・キーと、キーに関連付けられた行を格納するために使用可能な割り当てブロックです。

**チューニング上の考慮事項** エクステントのサイズを指示します。

## 固有のハッシュ値

**説明** 特定のクラスタ・キー値に基づいた固有のハッシュ値の数です。

**チューニング上の考慮事項** ハッシュ・クラスタは、頻繁に等価問合せを受ける静的な各表、またはクラスタ化された表のグループを格納するために使われます。ハッシュは表データを格納し、データ検索のパフォーマンスを改善するためのもう 1 つの方法です。ハッシュを使うには、ハッシュ・クラスタを作成して、表をクラスタにロードします。

ハッシュは、次の条件が満たされているときに最も効果を発揮します。

- 問合せのほとんどが、次に示すようなクラスタ・キー上の等価問合せである。

  ```
  SELECT .. . WHERE cluster_key = . . ;
  ```

  このような場合、等価条件におけるクラスタ・キーはハッシュされ、対応するハッシュ・キーは通常 1 回の読込みで検出されます。対照的に、索引表については、索引内のキー値をまず検出し（通常数回の読込みを要します）、次に表から行を読み込む（さらに読込みが必要）必要があります。

- ハッシュ・クラスタにおける表のサイズは、クラスタ内の表に必要な行数および領域を判断できるように、基本的には静的です。ハッシュ・クラスタ内の表が、クラスタへの初期割り当てよりも広い領域を必要とする場合、オーバーフロー・ブロックが必要になるため、パフォーマンスが著しく損なわれる場合があります。

## 列統計

「列統計」ラジオ・ボタンを選択すると、次の列統計が表示されます。

### 列名

共通列、またはクラスタ内の表によって共有される列です。

### 固有の値

列中の固有の値の数です。

**密度**
列中に固有の値が現れる回数です。

# 索引詳細

Oracle において索引は、表中の行への高速なアクセスを実現するために使われます。表中の行の小さな部分を返す操作では、索引を使うとデータへのアクセスがより速くなります。索引によってかなりの時間を節約できる一方で、SQL エンジンは、表に対して定義されたすべての索引を、それらが使われるかどうかに関係なく維持する必要があります。これにより、I/O 量の多いアプリケーション上では、CPU および I/O の負担が非常に大きくなります。したがって、使わない索引は削除してください。

索引の効果を判断するには、索引を作成、分析し、SQL Analyze で問合せに対して EXPLAIN PLAN を実行して、オプティマイザがその索引を使うかどうかを確認します。オプティマイザが索引を使うのであれば、それを維持するために必要なコストが極端に大きくない限り、その索引を保持します。また、索引がある場合とない場合の両方における、オプティマイザのコストを比較する方法もあります。

ただし、索引の使われ方には、文の実行計画の検証によっては直接明らかにできないものが存在することに注意してください。特に Oracle8 では、フォーリン・キーの制約を施行するとき、親表上での共有ロックが必要となることを回避するために、フォーリン・キー索引上で " ピン "（非トランザクション・ロック）が使われます。多くのアプリケーションでは、このフォーリン・キー索引はまったく（あるいはまれにしか）問合せをサポートしません。

索引の作成および管理についてのガイドラインは、『Oracle Server 概要』、『Oracle Server アプリケーション開発者ガイド』および『Oracle Server チューニング』を参照してください。

**索引詳細の表示**

1. オブジェクト詳細と統計の表示で説明するようにして、「オブジェクト詳細」ダイアログ・ボックスを表示します。
2. 「**索引**」ラジオ・ボタンを選択して、EXPLAIN PLAN の選択行によって参照されているクラスタの名前を表示します。
3. 参照される索引と同じ名前を持つタブをクリックします。

「索引統計」ダイアログ・ボックスに次の統計が表示されます。

## エクステント

**説明** 特定の種類の情報を格納するために割り当てられる、特定数の隣接したデータ・ブロックのことです。

**チューニング上の考慮事項** 3-14 ページの、表のエクステントに関する説明を参照してください。

### 割当てブロック

**説明** 索引キーと、キーに関連付けられた行を格納するために割り当てられるブロックです。

**チューニング上の考慮事項** 索引のサイズを指示します。

### ツリーの深さ

**説明** B- ツリー索引の深さです。

**チューニング上の考慮事項** この値が 4 を超える（B- ツリー索引が 4 レベル以上に分岐する）場合、この索引を削除し、作り直すことを検討してください。

### リーフ・ブロック

**説明** 現在の索引におけるリーフ・ブロック（B- ツリー索引における最下位の索引ブロック）の数です。

**チューニング上の考慮事項** 最下位の索引ブロック（リーフ・ブロック）には、索引化されたすべてのデータ値と、それに対応する ROWID が含まれます。この ROWID は、実際の行位置を特定するために使われます。この値は、索引のサイズと選択性を示します。

### 固有キー

**説明** 固有の索引値の数です。

**チューニング上の考慮事項** この値が小さい場合、データへのアクセスには B*- ツリー索引よりもビットマップ索引のほうが効率的です。

### キー当りの平均リーフ・ブロック

**説明** 索引中の各固有値が現れるリーフ・ブロックの平均値です。この統計は、最も近い整数値に丸められます。

**チューニング上の考慮事項** 索引の選択性を指示します。この値が大きいほど、問合せでは多くの行が選択されます。UNIQUE 制約および PRIMARY KEY 制約を施行する索引については、この値は常に 1 です。

### キー当りの平均データ・ブロック

**説明** 索引中の固有値が指す表中のデータ・ブロックの平均数です。この統計は、索引列に対してある値を含む行を格納するブロックの平均数です。この値は、最も近い整数値に丸められます。

**チューニング上の考慮事項** 索引の選択性を示します。UNIQUE 制約および PRIMARY KEY 制約を適用する索引については、この値は常に 1 です。

### クラスタ化係数

**説明** 表中の行の順序の量を、索引の値に基づいて表します。

**チューニング上の考慮事項** 索引の値がブロック数に近い場合、表は非常によく順序付けられているといえます。そのような場合、単一のリーフ・ブロック内の索引エントリは、同じデータ・ブロック内の行を指す傾向があります。索引の値が行数に近い場合、表の順序付けは非常に不規則であるといえます。そのような場合、リーフ・ブロック内の索引エントリが、同じデータ・ブロック内の行を指すことは、ほとんどありません。

## 検証ビュー

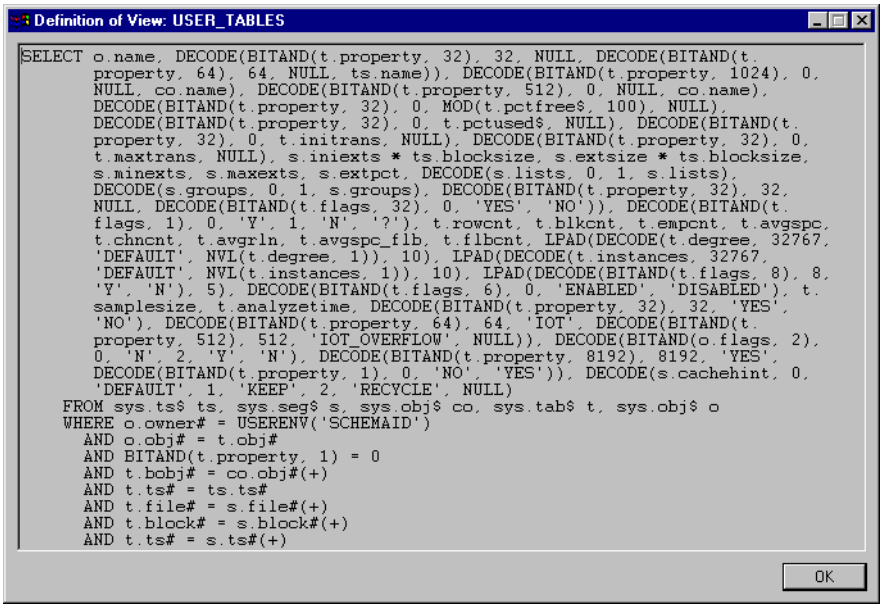
ビューのデータは、そのビューの基になっている表から導出されます。そのような表をビューの実表といいます。実表は表になることも、あるいはそれ自体がビューになることもあります。表と同様にビューに対しても、問合せ、更新、挿入および削除の各操作を制約なしに実行できます。ビュー上で実行されるすべての操作は、ビューの実表に影響を与えます。

ビューを作成するための基礎となる、選択基準について理解しておくと役に立ちます。「**ビュー定義を表示**」コマンドを使うことにより、ビューを作成する SQL を参照できます。

**ビューを検証するには、次の操作を行います。**

1. 「SQL テキスト」ウィンドウで、SQL 文からビューを選択。
2. **「SQL」=>「ビュー定義を表示」**を選択。
3. ビューを作成する SQL が「ビュー定義」ダイアログ・ボックスに表示されます。図 3–3 に、サンプルのビューを示します。

**図 3–3 「ビュー定義」ダイアログ・ボックス**




# Oracle オプティマイザの理解

最適化は、SQL 文を実行する最も効率的な方法を選択するプロセスです。これは、SELECT、INSERT、UPDATE、DELETE などの、すべてのデータ操作言語（DML）文の処理において重要なステップです。表または索引がアクセスされる順番が異なるように、SQL を実行するための異なる方法も多数存在します。文の実行に Oracle が使うプロシージャによって、文の実行速度に大きく影響する場合があります。

オプティマイザと呼ばれる Oracle のツールは、最も効率的であると判断した実行方法を選択します。オプティマイザは多くの要素を評価して、代替のアクセス・パスの中から選択を行います。オプティマイザが選択するアクセス・パスは、EXPLAIN PLAN を生成することにより表示できます。

オプティマイザと、オプティマイザがアクセス・パスを選択する方法について完全に説明することは本書の目的外ですが、次の項では、EXPLAIN PLAN と、その中の非効率的な部分を識別する方法の理解に役立つ基本的な概念の一部を説明します。

Oracle オプティマイザについての詳細は、『Oracle Server 概要』を参照してください。

## コストベースおよびルールベースの最適化

SQL の実行計画を選択するために、オプティマイザはコストベースまたはルールベースの 2 種類のアプローチのいずれかを使います。

## コストベースのアプローチ

コストベースのアプローチでは、利用可能なアクセス・パスを考慮し、文がアクセスするスキーマ・オブジェクト（表、クラスタまたは索引）のデータ・ディクショナリの統計に基づいた情報を計算に入れることにより、オプティマイザはどの実行計画が最も効率的であるかを判断します。コストベースのアプローチでは、ヒント（文中のコメントに記された最適化に関する提案）も考慮の対象となります。

概念的には、コストベースのアプローチは次のステップから構成されます。

1. オプティマイザは利用可能なアクセス・パスとヒントに基づいて、文に対していくつかの有力な実行計画を生成します。
2. オプティマイザは、データ・ディクショナリ内の表、クラスタおよび索引のデータ分布および記憶特性の統計に基づいて、各実行計画のコストを推定します。

   コストは、その実行計画を使って文を実行するために、必要と予想されるリソースに比例する推定の値です。オプティマイザは、その計画を使って文を実行するために必要な、概算のコンピュータ・リソースに基づいてコストを計算します。リソースには I/O、CPU 時間およびメモリーなどが含まれますが、これら以外の要素も考慮されます。

   コストの大きいシリアル実行計画は、よりコストの小さい計画よりも多くの実行時間を要します。ただし、パラレル実行計画を使っているとき、リソースの使用量は経過時間に直接関係しません。
3. オプティマイザは各実行計画のコストを比較し、最もコストの小さい計画を選択します。

## コストベース・アプローチの目標

デフォルトでは、コストベース・アプローチの目標はスループットが最大であること、すなわち、文によってアクセスされるすべての行の処理に必要なリソースが最小であることです。

Oracle は応答時間が最も短くなること、すなわち SQL によってアクセスされる第 1 行の処理に必要なリソースが最も少なくなることを目標にして、文を最適化できます。

## コストベース・アプローチの統計

コストベースのアプローチは、統計を使って各実行計画のコストを評価します。これらの統計は、表、列、索引およびパーティションのデータ分布および記憶特性を測定します。これらの統計は、ANALYZE コマンドを使って生成できます。オプティマイザはこれらの統計を基に、特定の実行計画を使って SQL を実行するために必要な I/O、CPU 時間およびメモリー容量を推定します。

## コストベース・アプローチを使う場面

Oracle SQL Analyze では、ルールベースおよびコストベースの両方のアプローチをテストできます。ただし、どちらのアプローチを選択するかについては、次のガイドに注意してください。一般に、すべての新規アプリケーションに対しては、コストベースのアプローチを使ってください。ルールベースのアプローチは、コストベースの最適化が利用可能になる前に作成されたアプリケーションのために用意されたものです。コストベースの最適化は、関係データ型およびオブジェクト型のどちらに対しても使用できます。

次の機能は、コストベースの最適化だけを使用できます。

- パーティション表
- パーティション・ビュー
- 索引構成表
- 逆キー索引
- ビットマップ索引
- パラレル問合せおよびパラレル DML
- スター型変換
- スター結合

コストベースのアプローチは一般に、特に複数の結合または複数の索引を使う大規模な問合せに対して、ルールベースのアプローチによって選択される計画と同程度、またはそれ以上に効率的な実行計画を選択します。コストベースのアプローチを使った場合、SQL 文をユーザー自身がチューニングする必要がないため、生産性も向上します。最終的に、Oracle のパフォーマンス関連の機能の多くは、コストベースのアプローチを通じてのみ利用できます。

効率的なスター問合せのパフォーマンスを実現するには、コストベースの最適化を使う必要があります。同様に、ハッシュ結合やヒストグラムについても、コストベースの最適化を使う必要があります。コストベースの最適化は常に、パラレル問合せおよびパーティション表とともに使われます。統計を最新の状態に保つには、ANALYZE コマンドを使用する必要があります。

## ルールベースのアプローチ

ルールベースのアプローチを使うと、オプティマイザは利用可能なアクセス・パスとそれらのランクに基づいて、実行計画を選択します。ルールベースの最適化では、関係データ型およびオブジェクト型の両方にアクセスできます。

Oracle によるアクセス・パスのランク付けは発見的なものです。SQL を実行する方法が複数ある場合、ルールベースのアプローチは常に、ランクが低い操作を使います。通常、ランクが低い操作は、ランクが高い構成体に関連付けられた操作よりも高速に実行されます。

## アクセス方法

この項では、Oracle がデータにアクセスするための基本的な方法を説明します。

### 全表走査

全表走査は、表から行を検索します。全表走査を実行するために、Oracle は表中のすべての行を読み込み、各行を検証して、それが文の WHERE 句の条件を満たしているかを判断します。Oracle は表に割り当てられたすべてのデータ・ブロックを連続して読み込みます。したがって、マルチブロック読込みを使うと、全表走査が非常に効率的に実行されます。Oracle は各データ・ブロックを 1 度だけ読み取ります。

### ROWID による表アクセス

ROWID による表アクセスもまた、表から行を検索します。行の ROWID は、その行が格納されているデータファイルおよびデータ・ブロックと、ブロック内での行の位置を指定します。ROWID による行位置の特定は、Oracle が単一の行を最も高速に発見する方法です。

ROWID によって表にアクセスするため、Oracle はまず文の WHERE 句から、あるいは表中の 1 つまたは複数の索引の走査によって、選択された行の ROWID を取得します。次に、その ROWID に基づいて、選択された各行の表中での位置を特定します。

### クラスタ走査

クラスタ走査では、索引クラスタに格納された表から、同じクラスタ・キー値を持つ行が検索されます。索引クラスタにおいて、同じクラスタ・キー値を持つ行はすべて同一データ・ブロックに格納されます。クラスタ走査を実行するために、Oracle はまずクラスタ索引を走査して、選択された行のうちの 1 行の ROWID を取得します。次に、この ROWID に基づいて、行の位置を特定します。

### ハッシュ走査

Oracle ではハッシュ走査を使うことにより、ハッシュ値に基づいて、ハッシュ・クラスタ内での行の位置を特定できます。ハッシュ・クラスタにおいて、同じハッシュ値を持つすべての行は同一データ・ブロックに格納されます。ハッシュ走査を実行するために、Oracle はまず、文によって指定されたクラスタ・キー値にハッシュ関数を適用して、ハッシュ値を取得します。次に、このハッシュ値を持つ行が格納されているデータ・ブロックを走査します。

### 索引走査

索引走査では、索引の 1 列または複数列の値に基づいて、索引からデータを検索します。索引走査を実行するために、Oracle は索引から、文によってアクセスされる索引列の値を探し

ます。文が索引の列だけにアクセスする場合、Oracle は索引列の値を、表からではなく索引から直接読み取ります。索引には索引値だけでなく、その値を持っている表中の行の ROWID も含まれます。したがって、文が索引列に加えてその他の列にアクセスする場合、Oracle は ROWID またはクラスタ走査による表アクセスによって、表中の行を発見できます。

索引走査の型のリストは、『Oracle Server 概要』を参照してください。

# パフォーマンス統計の理解

Oracle SQL Analyze では、SQL コードのパフォーマンスを監視および検証するためのさまざまな方法が用意されています。

- 既に実行されている文を、それらが消費するリソースによって分析してソートするには、TopSQL を使います。
- オプティマイザが文をどのように実行するかを知るには、さまざまな EXPLAIN PLAN を生成してください。EXPLAIN PLAN を使って、オブジェクト詳細および実行統計を検証することもできます。
- オプティマイザによって選択された結合の方法論を知るには、コンパクト・ビューを生成します。
- Oracle SQL Analyze の内部から文を実行して、統計を検証します。
- EXPLAIN PLAN および実行統計を互いに比較します。

この章の残りの部分では、Oracle SQL Analyze を使ってパフォーマンス統計を表示する方法を示します。また、EXPLAIN PLAN について説明し、Oracle SQL Analyze を使って EXPLAIN PLAN をより簡単に扱う方法についても説明します。

## TopSQL 統計

「チューニング対象の文の選択」で説明したように、TopSQL は Oracle SQL Analyze に統合された機能の 1 つで、SQL 文が消費するリソースを測定するために使います。これらの統計を使って、どの SQL 文が最も多くのリソースを消費しているかを判断し、それらをチューニングの対象として選択します。

TopSQL オブジェクトは、「ナビゲータ」ウィンドウに表われる、それぞれのデータベース・セッションに存在します。TopSQL は、文が消費するリソースを示す V$SQLAREA ビューからの統計を表示し、パフォーマンスに関する問題の識別を支援します。

TopSQL が示すパフォーマンス統計については、第 2 章の「チューニング・セッションの開始」で説明します。

## EXPLAIN PLAN の理解

数多くの表からデータを検索する SQL 文は、さまざまなバリエーションの表結合メソッド、結合順およびアクセス・パスを使って、同じ結果の集合を得ることができます。Oracle オプティマイザは、次に示すような多くの要素（このほかにもあります）に基づいて、これらの操作に対して最適なアクセス・パスを発見する必要があります。

- 使用可能な索引
- SQL 文中の表および列の順序
- 文中で参照されるオブジェクトのカーディナリティの統計
- ヒント

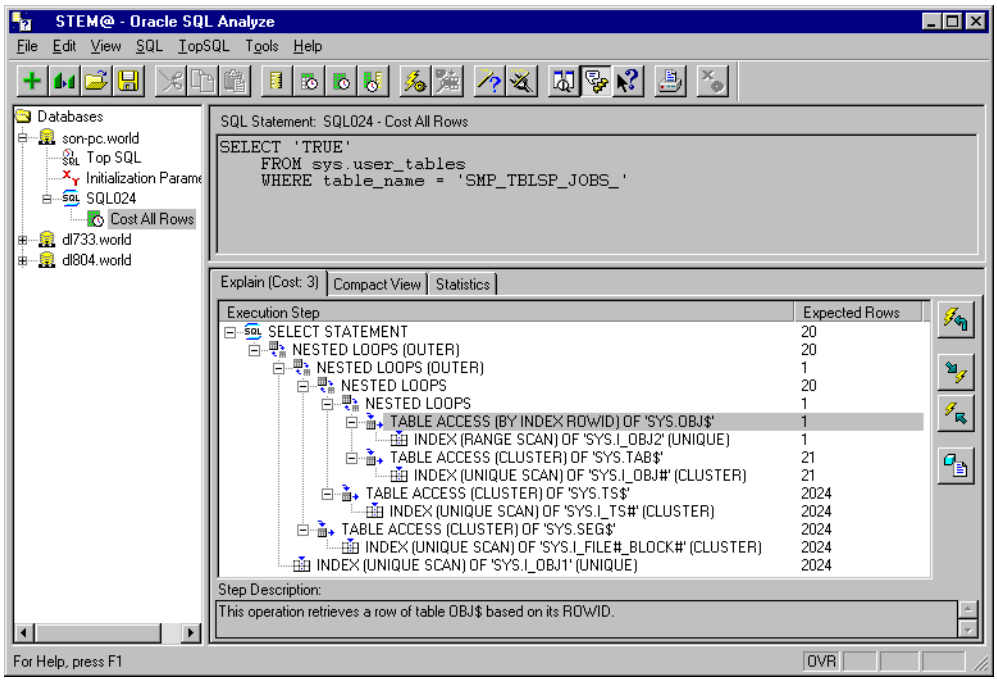
ルールベース、応答時間優先のコストベース、あるいはスループット優先のコストベースのうち、オプティマイザがどのアプローチを使うかに応じて、これらの要素は異なります。SQL 文の実行パスは、EXPLAIN PLAN を通じて表示できます。EXPLAIN PLAN は、文の実行に伴う操作をリスト形式で示します。EXPLAIN PLAN を検証することにより、Oracle が SQL 文をどのように実行しているかを正確に知ることができます。

Oracle SQL Analyze には、EXPLAIN PLAN を簡単に生成するための機能が用意されています。生成した EXPLAIN PLAN を使って、SQL 文がさまざまなオプティマイザ・モードの下でどのように動作するかを査定できます。それぞれのオプティマイザ・モードの下で SQL ノードを実行して、その文に対する EXPLAIN PLAN を生成できます。また、コストベースのオプティマイザが使われている場合には、実行の " コスト " を知ることもできます。コストは、I/O および CPU 消費量などのコンピュータ・リソースや、文の実行を完了するための時間などの、さまざまな要素を測定した値です。

### EXPLAIN PLAN の生成

EXPLAIN PLAN を生成するには、メニューの**「SQL」=>「解説」**から、目的の最適化パスを選択します。図 3–4 に示すように、「詳細」ウィンドウに EXPLAIN PLAN が表示され、EXPLAIN PLAN オブジェクトが「ナビゲータ」ウィンドウに追加されて、関連の SQL 文に接続されます。

**図 3–4　EXPLAIN PLAN**




SQL 文の EXPLAIN PLAN を表示するために、4 種類の最適化パスを選択できます。

- ルール（ルールベース最適化）
- 最初の行をコスト（応答時間優先のコストベース最適化）
- すべての行をコスト（スループット優先のコストベース最適化）
- 選択（オプティマイザの選択による）

次の項では、EXPLAIN PLAN の使い方について順を追って学習します。詳細は、『Oracle Server チューニング』を参照してください。よりわかりやすいアプローチとして、Oracle SQL Analyze では、" ウォークスルー " を使った EXPLAIN PLAN のガイドも用意されています。

## EXPLAIN PLAN の読み方

EXPLAIN PLAN を検証するには、まず処理がどこで始まっているかを理解し、次にパスを追う必要があります。これについて、次に説明します。次の SQL 文を例にします。

```
SELECT "name", product_id, amount_in_stock, state
FROM inventory, product, warehouse
```

```
WHERE product.id = inventory.product_id
AND amount_in_stock > 500
AND warehouse.id = inventory.warehouse_id;
```

この文は、ルールベースのオプティマイザを使った場合、次の EXPLAIN PLAN として表されます。

```
SELECT STATEMENT
   NESTED LOOPS
NESTED LOOPS
   TABLE ACCESS (BY ROWID) OF 'INVENTORY'
      INDEX (RANGE SCAN) OF 'AMOUNT_IN_STOCK_PK' (NON-UNIQUE)
   TABLE ACCESS (BY ROWID) OF 'WAREHOUSE'
      INDEX (UNIQUE SCAN) of 'WAREHOUSE_ID_PK' (UNIQUE)
   TABLE ACCESS (BY ROWID) of 'PRODUCT'
      INDEX (UNIQUE SCAN) of 'PRODUCT_ID_PK' (UNIQUE)
```

実行パスは駆動表として INVENTORY を使い、実行パスは次のようになります。

1. 表の AMOUNT_IN_STOCK_PK 索引の範囲走査を Oracle が実行。
2. 索引からの複数の ROWID を検索した後、Oracle はこれらの値を使って、INVENTORY 表から行を検索。
3. WAREHOUSE_ID_PK 索引を使って、Oracle が ROWID を検索。
4. ステップ 3 では、Oracle は ROWID によって WAREHOUSE 表にアクセス。
5. 次に、2 つの表から返された集合の NESTED LOOPS 結合を Oracle が実行。
6. PRODUCT 表を含む操作を実行。
7. 最終の操作は、PRODUCT 表からの集合と、INVENTORY 表と WAREHOUSE 表の結合の結果である集合との NESTED LOOPS 結合。

これまで説明したように、計画における位置付けとは逆の順序で操作が実行されます。

## EXPLAIN PLAN のウォークスルー

計画を " ウォークスルー " することにより、文がどのように実行されているか、各操作はどのステップを実行しているのかについて即座に理解できます。各操作は、実行の順に強調表示されます。ウォークスルーのペースは制御可能です。また、任意の時点でバックアップを取ったり、最初からやり直したりできます。ある操作が強調表示されると、その操作の説明が EXPLAIN PLAN 表示の下の、「ステップ説明」ウィンドウに表示されます。特定の操作に関連する任意のオブジェクトについて、詳細を表示するよう選択できます。

EXPLAIN PLAN をいったんオープンし、それが表示されたら、Oracle SQL Analyze を使ってその EXPLAIN PLAN をステップの実行順に検証できます。Oracle SQL Analyze では、SQL の実行用語でステップを説明します。

EXPLAIN PLAN **をウォークスルーするには、次の操作を行います。**

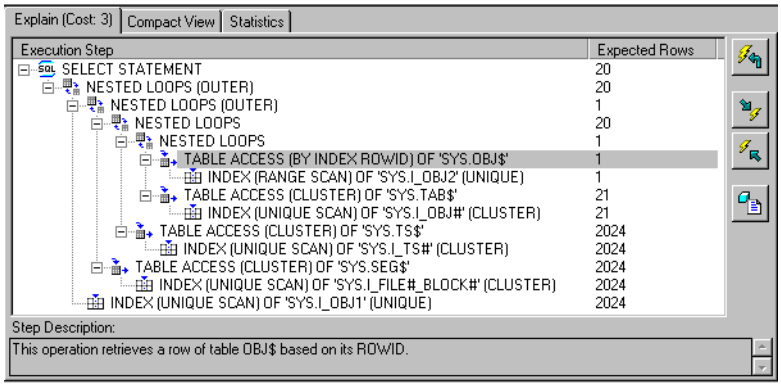
1. **「表示」=>「ステップ説明** (&D)**」**を選択して、「ステップ説明」フレームを有効にする。

   「ステップ説明」フレームは、「詳細」ウィンドウの右下部分を占めます。右側には 4 つのナビゲーション・ボタンがあります。下部には「ステップ説明」ボックスがあります。図 3–5 に示すように、「ステップ説明」フレームが表示されます。

2. ウォークスルー・ナビゲーション・ボタンを使って、EXPLAIN PLAN 間を移動し、表、クラスタまたは索引など、選択されたオブジェクトに対するオブジェクト詳細を呼び出せます。

EXPLAIN PLAN を検証する過程で、選択されたステップの説明が「詳細」ウィンドウの下部に表示されます。

**図 3–5　EXPLAIN PLAN とステップ説明**




## EXPLAIN PLAN の統計

EXPLAIN PLAN は、次の列を表示します。

### 実行手順

オプティマイザによって実行される操作です。

### 実行見込み行

実行計画の中で、このステップによってアクセスされる行の数です。

パラレル問合せに対する EXPLAIN PLAN には、さらに次の列が含まれます。

### 操作ノード

操作からの出力が消費される順序を説明します。

**操作タイプ**
実行される操作の種類を説明します。

**問合せテキスト**
問合せサーバーによって使われる問合せを説明します。

Oracle8 パーティションの EXPLAIN PLAN には、次の 3 つの列が追加されます。

**パーティション起動**
アクセスされるパーティション範囲の開始パーティションです。

**パーティション停止**
アクセスされるパーティション範囲の停止パーティションです。

**パーティション ID**
パーティション起動およびパーティション停止の値の組を計算した PARTITION ステップの ID です。

これらの列についての完全な説明とその意味については、『Oracle Server チューニング』を参照してください。

## コンパクト・ビューのウォークスルー

EXPLAIN PLAN の別種にコンパクト・ビューがあります。コンパクト・ビューは、現在の EXPLAIN PLAN で使われる結合の方法論に重点を置いて、EXPLAIN PLAN を表示します。結合表は子としてではなく、ピアとして示されます。これにより、どの表が結合されているか、それらの結合にどの方法が使われているかをより明確に確認できます。

図 3–6 に、コンパクト・ビューのサンプルを示します。

コンパクト・ビューには次の列が示されます。

**実行手順**
結合を強調表示するように再構成された EXPLAIN PLAN を示します。

**結合メソッド**
EXPLAIN PLAN の各結合表に対して使われる結合の種類を表示します。

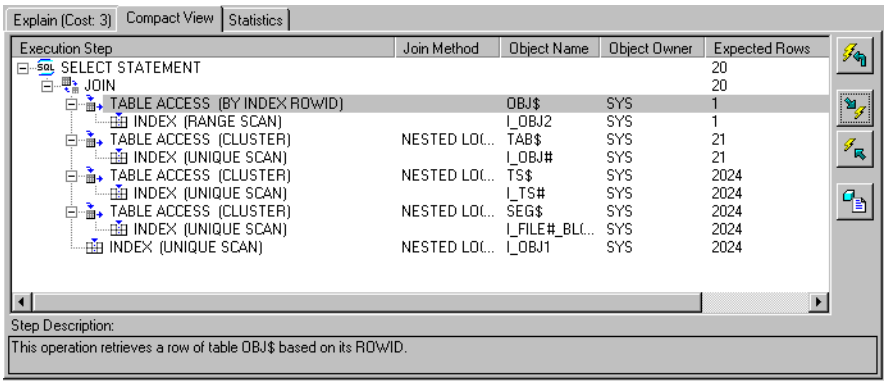
**オブジェクト名**
その結合に関連付けられた、表および索引の一覧を表示します。

**オブジェクト所有者**
オブジェクトが属するセッションの名前です。

**実行見込み行**
文が実行されるときに、そのステップがフェッチすると予想される行数です。

標準の EXPLAIN PLAN と同様に、コンパクト・ビューをウォークスルーし、オブジェクト詳細を検証し、実行統計をレビューできます。

**図 3–6　コンパクト・ビュー**




## 実行統計の表示

実行統計は、データベースからのデータにアクセスするときの SQL 文のパフォーマンスに関する情報を示します。

**実行統計を表示するには、次の操作を行います。**

1. 選択された SQL 文の EXPLAIN PLAN を生成。
2. **「SQL」=>「実行」**を選択。SQL 文の実行には、多少時間がかかる場合があります。
3. 「詳細」ウィンドウから、「統計」ページを選択。

SQL 文とその EXPLAIN PLAN の実際のパフォーマンスをより正確に知るために、文を何度か実行して、統計の平均集合を得ることができます。

## SQL 文と EXPLAIN PLAN の比較

EXPLAIN PLAN の比較は、SQL のチューニングの間に達成したパフォーマンス向上を分析するための強力なツールです。Oracle SQL Analyze では、2 つの異なる SQL 文を開き、結果を比較するためのスプリット・ビューを作成できます。

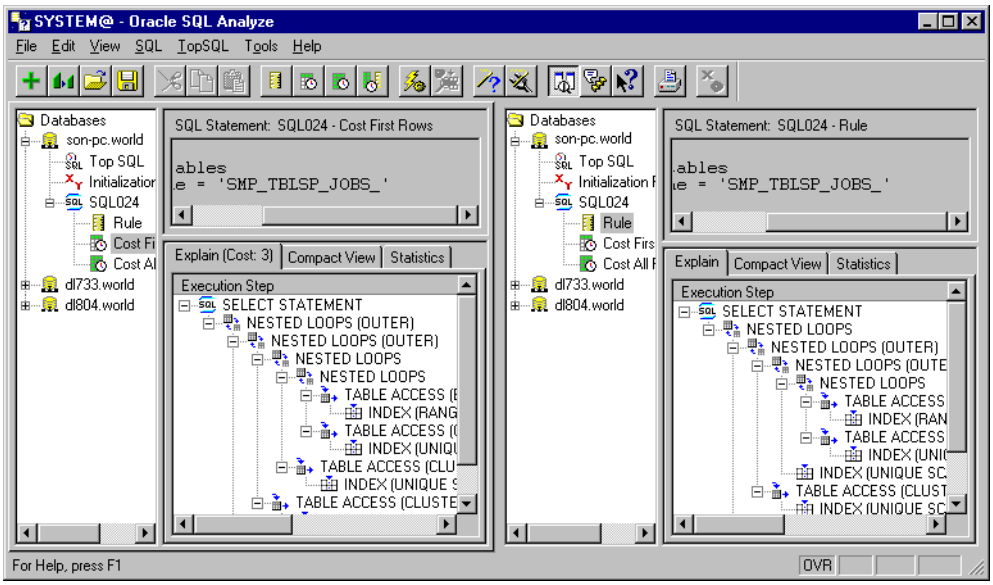
SQL 文および EXPLAIN PLAN を比較するには、メニューから**「表示」=>「比較」**を選択します。

図 3–7 に示すように、メイン・ウィンドウは、それぞれ同一のナビゲーション・ウィンドウおよびメイン・ウィンドウを持つ 2 つのペインに分かれています。

2 つのペイン間を移動して、代替の SQL 文、EXPLAIN PLAN およびパフォーマンス統計を表示および比較できます。

単一のメイン・ウィンドウに戻るには、**「表示」=>「比較」**を再び選択します。

**図 3–7　比較ビュー**



# 4

# SQL 文のチューニング

この章では、次の項目について説明します。



## SQL 文のチューニング

Oracle SQL Analyze は、チューニングに対するさまざまな視点からのアプローチを可能にする、柔軟性のあるツールです。

たとえば、TopSQL を通じて文を選択し、そのパフォーマンスを分析して、文をチューニングするための適切な基準を判断できます。構文を手動で編集する、あるいはヒント・ウィザードを使ってヒントを追加するなど、どのような方法を使う場合でも、チューニングのために必要な作業を的確に判断できます。

ファイルに格納された文に対して、必要なチューニングを実施することもできます。また、パフォーマンスの統計をユーザー自身が検証しなくとも、チューニング・ウィザードを使って文を処理し、ウィザードに文を自動的にチューニングさせられます。

この章ではまず、手動での編集とチューニングの方法論について説明します。次に、ヒントとチューニング・ウィザードを使って、チューニング・プロセスを自動化する方法を説明します。

# 手動での文の編集

「SQL テキスト」ウィンドウにテキストを入力することにより、手動で文を編集できます。文を入力している間、構文のチェックは行われませんが、それでも EXPLAIN PLAN を生成したり、文を実行したり、その結果を編集前の文または分析済みの他の文と比較したりして、文をテストできます。

次の条件に 1 つでも該当する場合には、文を編集できません。

- 「TopSQL テキスト」ウィンドウからドラッグされた文
- EXPLAIN PLAN が既に生成されている文

これらの条件に該当する場合、**「SQL」=>「類似作成」**を選択して、その文の編集可能なコピーを作成します。新しい文に対して、ナビゲーション・ツリーに SQL 文オブジェクトが作成されます。次に、新しい文の編集に進みます。

# ヒントの理解

アプリケーション・デザイナがあるデータについて持っている情報の中には、オプティマイザの側からは認識できないものも存在します。たとえば、ある種の問合せに対して、オプティマイザの判断よりもより選択に値する、特定の索引が存在することを開発者が認識しているような場合があります。このような情報に基づいて、オプティマイザの選択よりも効率的な実行計画を選択できる場合があります。そのような場合、ヒントを使って、ユーザーが選択した実行計画をオプティマイザが使うようにできます。

ヒントを使って、次のことを指定できます。

- SQL 文に対する最適化のアプローチ
- SQL 文に対するコストベース・アプローチの目標
- 文によってアクセスされる表のアクセス・パス
- 結合文の結合順序
- 結合文における結合操作

## ヒントの指定

ヒントは、表示される文ブロックの最適化だけに適用されます。文ブロックとは、次のような文のどれか 1 文、あるいは文の集合の一部分のことです。

- 単純な SELECT、UPDATE または DELETE 文
- 複雑な文の親文または副問合せ
- 複合問合せの一部

文中のコメントの中にヒントを記述することにより、その SQL 文についてのヒントをオプティマイザに送ります。

**注意：** コメントの詳細は、『Oracle Server SQL リファレンス』を参照してください。

ヒントを含むコメントは、各文に対して 1 つだけ含めます。このようなコメントは、SELECT、UPDATE または DELETE の各キーワードの直後にのみ指定できます。

ヒントの指定が正しくない場合、Oracle はそのヒントを無視しますが、エラーは返しません。

- ヒントを含むコメントが、DELETE、SELECT または UPDATE の各キーワードの直後以外の場所に記述されている場合、Oracle はヒントを無視します。
- 構文エラーを含むヒントは無視されますが、同じコメント内のその他の正しいヒントは有効です。
- 互いに競合するヒントの組は無視されますが、同じコメントの中のその他のヒントは有効です。

PL/SQL バージョン 1 を使う環境では、SQL 文中のヒントはすべて無視されます。

オプティマイザは、コストベースのアプローチを使っているときに限り、ヒントを認識します。何らかのヒント（RULE ヒントを除く）が文ブロックに含まれていると、オプティマイザは自動的にコストベースのアプローチを使います。

下記のヒントは、ヒントが影響を及ぼす最適化領域によって分類されており、手動で、またはヒント・ウィザードを使って SQL 文に追加できます。

データベースのバージョンによっては、一部のヒントについてその使用が制限されている場合があります。

**実行パス**

ALL ROWS

CHOOSE

FIRST ROWS

RULE

**結合順序**

ORDERED

STAR**

STAR_TRANSFORMATION*

**結合操作**

DRIVING_SITE*

USE_HASH**

USE_MERGE

NO_MERGE*

USE_NL

**その他のヒント**

CACHE

NOCACHE

PUSH_SUBQ

**アクセス方法**

AND_EQUAL

CLUSTER

FULL

HASH

HASH_AJ

INDEX

INDEX_ASC

INDEX_COMBINE*

INDEX_DESC

INDEX_FFS*

MERGE_AJ**

ROW_ID

USE_CONCAT

**パラレル実行**

APPEND*

NOAPPEND*

NOPARALLEL

PARALLEL

PARALLEL_INDEX*

* Oracle8 データベースでのみ使用可能

* Oracle7.3 および Oracle8 データベースでのみ使用可能

ヒントの詳細は、Oracle SQL Analyze オンライン・ヘルプおよび『Oracle Server チューニング』を参照してください。

# ルールの理解

SQL 文の構文は、パフォーマンスに大きな影響を及ぼします。特定のコマンド句を使うことにより、索引が使用禁止とされたり、データのソートおよびフィルタが非効率的になることがあります。コマンド句の使用順序や、データおよび表の参照順序によって、リソースの負荷が増大するような場合もあります。

Oracle SQL Analyze では、データベースに知識の深いユーザーによって培われたルールの集合が用意されています。このルールに従って SQL 文が評価され、可能であれば代替の文が提示されます。これらのルールは、パフォーマンスを最適化するための次のような原則に焦点を合わせたものです。

- 索引を使用可能にして、全表走査の必要を回避する。
- 必要なソート、マージおよびフィルタ操作の数を減らす。
- ソート、フィルタまたはマージが必要な行数を減らす。

Oracle SQL Analyze では、ユーザーがチューニング・ウィザードを使って文をチューニングするときにこれらの " ルール " が適用され、可能であれば代替の SQL 文が提示されます。Oracle SQL Analyze では、文が次のようなルールに従っているかどうかがチェックされます。これらのルールについては、他の項で説明します。

- NOT IN のかわりに NOT EXISTS を使用
- MINUS のかわりにヒント付きの NOT EXISTS または NOT IN を使用
- TRUNC の使用方法の変更による索引の有効化
- 演算子の使用方法の変更による索引の有効化
- 演算子の両側での列の非使用
- HAVING のかわりに WHERE を使用
- UNION のかわりに UNION ALL を使用

### NOT IN のかわりに NOT EXISTS を使用

NOT IN のかわりに NOT EXISTS を使うと、問合せに対する制限条件が追加されます。これにより、必要な全表走査の回数を減らすことができます。

次の例では、EMPLOYEE 表には部局 ID が存在しない場合に、NOT IN 句を使って DEPARTMENT 表から名前および部局 ID が検索されます。

```
SELECT name, department_id
FROM department
WHERE department_id NOT IN
(SELECT department_id FROM employee)
```

NOT IN は制限条件を使わないため、Oracle は DEPARTMENT の全表走査を実行します。DEPARTMENT の各レコードに対して、副問合せが実行されます。副問合せは制限を行う WHERE 句を持たないため、DEPARTMENT の全表走査において、すべてのレコードに対して全表走査が実行されます。

この場合、かわりに NOT EXISTS を使うことにより、DEPARTMENT 表の各行に対する副問合せにおいて、ネストされた索引走査が使われます。NOT EXIST 句の論理では、両方の表に一致を発見した場合、その行を返さないように Oracle に指示します。DEPARTMENT から返される唯一のレコードは、副問合せから行を返さないレコードであり、副問合せによる全表走査は実行されません。したがって、次の文は以前の例に比べてより効率的です。

```
SELECT name, department_id
FROM department,
WHERE NOT EXISTS
(SELECT department_id
FROM employee
WHERE department.department_id=employee.department_id)
```

### MINUS のかわりにヒント付きの NOT EXISTS または NOT IN を使用

MINUS は、最初の問合せによって得られた行の集合から、次の問合せによって得られた行の集合を除いた結果を返します。NOT EXISTS または NOT IN を使って問合せを記述し直すと、索引の利用が可能になり、句が必要とする全表走査の回数を減らすことができます。

ハッシュ結合不可（HASH_AJ）は通常、ソートを要求しないため、MINUS よりもよい結果をもたらすと Oracle SQL Analyze によって判断される場合があります。

たとえば、次の問合せは、EMPLOYEE 表の名前と誕生日を STOCKHOLDER 表と照合し、株主でない従業員の名前と誕生日を返します。MINUS は索引を使わないため、MINUS 操作が実行可能になる前に、Oracle は 2 回の全表走査を使って、各テーブル上でソートを実行します。

```
SELECT birth_date, last_name, first_name
FROM employee
MINUS
```

```
SELECT birth_date, last_name, first_name
FROM stock_holder
```

NOT EXISTS を使って文が記述し直されると、主文での行に対する副問合せにおいて、Oracle はネストされた索引走査を使用できます。

```
SELECT birth_date, last_name, first_name
FROM employee
WHERE NOT EXISTS
(SELECT 1
FROM stock_holder
WHERE stock_holder.birth_date = employee.birth_date
AND stock_holder.first_name = employee.first_name)
```

ハッシュ結合不可の方がよい結果が得られると Oracle SQL Analyze が判断した場合、例の問合せは、ソートおよびマイナスの操作を実行するかわりに、2 回の全表走査と非結合のアルゴリズムを使って行を結合するように記述し直せます。

```
SELECT birth_date, last_name, first_name
FROM employee
WHERE (birth_date, last_name, first_name)NOT IN
(SELECT /*+ hash_aj (stock_holder) */ birth_date, last_name, first_name
FROM stock_holder)
```

## TRUNC の使用方法の変更による索引の有効化

索引列上で切捨てコマンド（TRUNC）を使うと、索引は使用禁止とされます。切捨てられる行が少なくなるように問合せを記述し直すと、索引の利用が可能になり、パフォーマンスを改善できます。

次の例では、trans_date は索引列ですが、TRUNC コマンドにより索引は使用禁止とされています。

```
SELECT account_name, trans_date
FROM transaction
WHERE TRUNC(trans_date) = TRUNC(sysdate)
```

trans_date 索引を使い、パフォーマンスを改善するために、この問合せを次のように記述し直せます。

```
SELECT account_name, trans_date
FROM transaction
WHERE trans_date BETWEEN TRUNC(sysdate) AND TRUNC(sysdate) + .99999
```

## 演算子の使用方法の変更による索引の有効化

索引列が関数の一部（WHERE 句の中で）となっている場合、オプティマイザは索引を使いません。等式を書き直すことにより演算子の使用を避けられると Oracle SQL Analyze が判断した場合、文を記述し直せます。

この例の問合せ中の等式は、単純な不等式句に書き直せます。次の問合せを例とします。

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE amount + 3000 < 5000
```

この問い合わせは、次のように記述し直せます。

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE amount < 2000
```

## 演算子の両側での列の非使用

索引列が演算子の両端に現れるとき、その列に対する索引は使用禁止とされます。Oracle SQL Analyze はこの状況を検出し、可能であれば文を記述し直して、索引を使用可能にします。

次の例では、列 account_name は索引列ですが、索引は使用禁止とされています。

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE account_name = NVL(:acc_name, account_name)
```

索引列が演算子の一方の側だけに現れるように、LIKE を使ってこの問合せを記述し直せます。

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE account_name LIKE NVL(:acc_name, '%')
```

## HAVING のかわりに WHERE を使用

HAVING 句は、GROUP BY 句によって収集された列が集約された後にのみ、それらの列を制限します。可能であれば常に、検索された列がマージおよびソートされて集合体になる前に、それらの列の数を制限することが理想的です。HAVING のかわりに WHERE を使うことにより、集合体に追加される前に列を排除できます。

次の文は、アイテムのリスト全体を数によってソートし、次に値が 40 に満たないアイテムをすべて集合体から削除します。

```
SELECT quantity, AVG(actual_price)
FROM item
GROUP BY quantity
```

```
HAVING quantity > 40
```

この文を記述し直して、集合体がソートされる前に、QUANTITY が 40 に満たないすべての行が削除されるようにできます。

```
SELECT quantity, AVG(actual_price)
FROM item
WHERE quantity >40
GROUP BY quantity
```

HAVING 句が集約関数に対して適用される場合、それを WHERE で置き換えることはできないため注意が必要です。たとえば、次の問合せでは、HAVING が SUM 関数に適用されています。

```
SELECT program_name
   ,count
   ,min(end_date-start_date) "Min Runtime"
   ,avg(end_date-start_date)"Avg Runtime"
   ,max((end_date-start_date)"Max Runtime"
   ,sum(end_date-start_date)"tot Runtime"
FROM jobs
WHERE start_date>sys_date - 7
GROUP BY program_name
HAVING sum((end_date-start_date)>0.25 or max(end_date-start_date) > 0.04
```

### UNION のかわりに UNION ALL を使用

UNION と UNION ALL との間の違いは、UNION の場合、2 つの行集合にまたがって複製された行の削除にソート操作が必要であることに対し、UNION ALL の場合、行が複製されている場合であってもすべての行が返されることです。複製された行が重要でない場合、UNION ALL を使うことにより、コストが高い可能性のあるソート、マージおよびフィルタ操作を避けることができます。

次に例を示します。

```
SELECT acct_num, balance_amt
FROM debit_transactions
WHERE tran_date = '31-DEC-96'
UNION
SELECT acct_num, balance_amt
FROM credit_transactions
WHERE tran_date = '31-DEC-96'
```

この文は、次のように記述し直せます。

```
SELECT acct_num, balance_amt
FROM debit_transactions
WHERE tran_date = '31-DEC-96'
UNION ALL
SELECT acct_num, balance_amt
FROM credit_transactions
WHERE tran_date = '31-DEC-96'
```

# 結合の方法論の理解

ほとんどの SQL 問合せは、複数の表からのデータ選択を伴います。このような操作では、複数の表からのデータがマージ、または「結合」され、目的の結果の集合が生成されます。使用される結合方法の種類および表結合の順序は、索引の存否、選択プロセスに関係する列のカーディナリティなど、さまざまな要素によって決定されます。問合せに関係する 1 枚または複数枚の表に索引が存在する場合、各表から特定の行を選択でき、次に選択された行集合を比較および結合できます。

表から取得されたデータが、別の表からのデータ選択の基準として使われる場合、表アクセス操作の順序は、複数の表に対する問合せのパフォーマンスを決定する主要な要素となります。結合される表の数が増えるにつれて、適切な結合順序を決定するプロセスはより複雑になります。n 枚の表に対して考えられる結合順序の数は、n の階乗で表されます。6 枚の表の結合に対しては、720 通りの結合順序が考えられます。

コストベースおよびルールベースのオプティマイザは、特定のルール、データおよび方針を使って、結合順序を決定します。経験のある開発者であれば、どちらのオプティマイザの動作に対しても影響を与えることができます。

理想的な結合順序を判断し、選択文において結合順序の逆（最下位から最上位）に表を列挙することにより、複数の表を結合する際のパフォーマンスを改善できます。この場合、オブ

ジェクトに対して平等なルール条件が存在するとすれば、表の順序によって結合順序が決定されます。

SQL ヒントを使って、コストベース・オプティマイザによる最適化を支援し、実行計画を制御できます。このことは、オプティマイザが認識できない次のような詳細事項について、ユーザーが認識しているような、特定の問合せを行う際に役立ちます。

- 特別な問合せのパフォーマンス目標（スループットまたは応答時間）
- ある種のオブジェクトに対する、古いまたは既に存在しない統計
- フィルタ条件に影響する可能性があるバインド変数

特定の問合せに対して、ヒントを使って結合メソッドおよび順序を制御する作業は複雑であり、また危険を伴います。この作業を支援するために、Oracle SQL Analyze のチューニング・ウィザードは、適切な局面で代替の結合方針を評価するために使われる、自動化された方法論を提供します。Oracle 内部のコストベース・オプティマイザと同様に、Oracle SQL Analyze はさまざまな方法で、文を実行する際のコストを推定します。

- アクセス方法のコストは、その方法によってアクセスされるブロックの数に正比例します。
- 結合方法のコストは、その結合方法によってアクセスされるブロックの数と、結合を "駆動 " する表の種類（行カウントが最も少ないと見込まれる表）に左右されます。

代替の NESTED LOOPS 結合順序を使用できる場合、Oracle SQL Analyze は必要なヒントを加えるか、オブジェクトの順序を入れ替えるかして、文を記述し直します。文をさらに分析して、パフォーマンスを改善するためにその他の結合方法を追加できるかどうかも判断できます。

たとえば、意思決定を支援するための次の問合せは、長い実行時間を必要とします。

```
SELECT a.name, b.description, c.count, d.name, e.name
FROM a, b, c, d, e, f, g
WHERE a.id = :id
     AND a.id = b.id
     AND a.date BETWEEN :lo_date and :hi_date
     AND b.id = c.id
     AND c.val = DECODE(:frequency, 'COMMON', 0, 'UNKNOWN', g.unknown_val,
         'RARE', 2)
     AND c.id = d.id
     AND d.id = e.id
     AND d.in_stock < 1000
     AND e.id = f.id
     AND e.subid = f.subid(+)
     AND f.date BETWEEN :lo_date AND :hi_date
     AND f.id = g.id
```

この問合せには、7 の階乗（5040）通りの結合順序が考えられます。データ・ディクショナリの統計が古かったり、ある種のオブジェクトについての統計が欠けているような場合、オプティマイザが結合順序を算定する作業には大きな障害が伴います。正確な統計が利用できたとしても、フィルタ条件 BETWEEN については、オプティマイザによって推測される必要があります。

フィルタ条件の選択性および推定された統計に基づいて、Oracle SQL Analyze はさまざまな結合方針のコストを推定します。その結果、結合順序は g、c、d、a、b、e、f の順に決定されます。結合方法については、ハッシュ結合が必要な f 以外のすべての表に対して、ネストされたループ結合が使われます。アクセス方法については、全表走査が必要な g および d を除くすべての表に対して、一意の索引参照が使われます。チューニング・ウィザードを使ってこの文をチューニングすると、Oracle SQL Analyze はオブジェクト順を修正し、NESTED LOOPS ヒントを追加して、前述の分析結果をインプリメントします。

結合の方法論の詳細は、『Oracle Server 概要』を参照してください。

# チューニング・ウィザードの使用方法

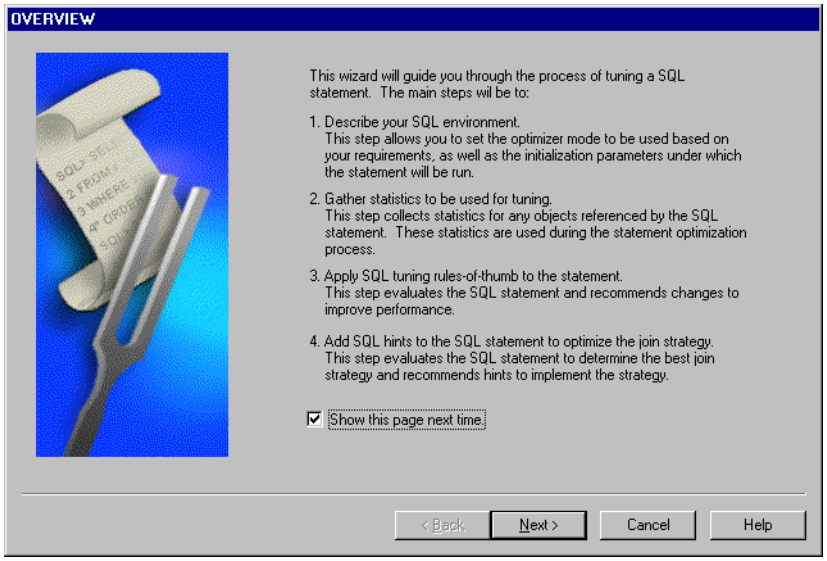
チューニング・ウィザードは、SQL 文のチューニング・プロセスを順を追ってガイドするツールです。ルールおよび結合の方法論のルールを使って SQL 文を評価し、既存の文を置き換える、最適化された SQL 文を生成します。

**チューニング・ウィザードを使うには、次の操作を行います。**

**「ツール」=>「チューニング・ウィザード」**を選択します。チューニング・ウィザードは、次のチューニング・プロセスを順を追ってガイドします。

1. SQL 文について説明
2. チューニングのための統計を収集
3. SQL チューニングのルールを SQL 文に適用
4. SQL 文に SQL ヒントを追加して、結合方針を最適化

**図 4–1　チューニング・ウィザードの起動画面**



# チューニング・ウィザードのプロセス

チューニング・ウィザードは、SQL 文をチューニングするためのプロセスを、順を追って自動的にガイドします。プロセス全体を通じて、ウィザードが特定の SQL 文を最適化するための支援となる選択を行うことができます。選択にあたって詳しい情報が必要な場合は、「**ヘルプ**」をクリックします。

チューニング・ウィザードは、次のステップを順を追ってガイドします。

1. SQL 文について説明
   - チューニング環境（OLTP または DSS）について説明
   - 初期化パラメータを設定
2. チューニングのための統計を収集
   - オブジェクトの統計を収集する方法を設定
3. SQL チューニングのルールを SQL 文に適用
4. ルールを適用
   - SQL 文に SQL ヒントを追加して、結合方針を最適化
   - 最適化を行うための、結合の方法論を選択
   - バインド変数に値を割当て
5. 結合の方法論を適用
6. 元の文と修正済みの文を比較

チューニング・ウィザードの最も強力な機能の 1 つは、元の文と修正済みの文を比較できることです。文を修正でき、それによって文のパフォーマンスが改善されたかどうかをすぐに知ることができます。

# ヒント・ウィザードの使用方法

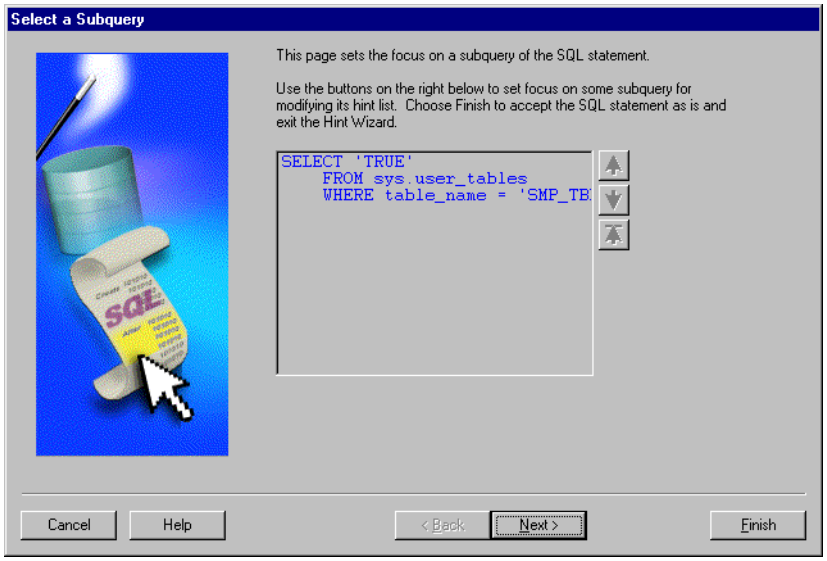
ヒント・ウィザードは、文中のヒントを識別し、文に追加できる他のヒントをユーザーが提示することを可能にします。ヒント・ウィザードでは、選択されたヒントに対する説明が表示され、ヒントが追加された場合と削除された場合の新しい SQL 文が自動的に生成されます。

**ヒント・ウィザードを使うには、次の操作を行います。**

**「ツール」=>「ヒント・ウィザード」**を選択します。ヒント・ウィザードは、次のプロセスを順を追ってガイドします。

1. 「ヒント・ウィザード」ページから、分析対象の副問合せを選択
2. 現在のヒントを表示または削除
3. 追加する新しいヒントを選択
   - 必要に応じて、表パラメータを提示
   - 必要に応じて、索引パラメータを提示
4. 現在のヒントをレビュー
5. SQL 文にヒントを適用

**図 4–2　ヒント・ウィザードの起動画面**



# 5

# パフォーマンスの検証

## SQL のパフォーマンス改善を検証する方法

この章では、これまでのチューニング作業によって、文のパフォーマンスが実際に改善されたことを検証する重要性について再確認します。また、Oracle SQL Analyze を使ってその検証作業を行う方法についても示します。

文のパフォーマンスが改善されたことを検証するには、これまで情報の収集に使ってきたのと同じ、次の方法を使います。

- 3-32 ページで説明するように、文を実行してその結果を比較
- 3-32 ページで説明するように、新しい EXPLAIN PLAN を生成して比較
- 3-12 ページで説明するように、オブジェクトの詳細をレビューして、効率的に使われていることを確認

# 用語集

### ANALYZE SQL コマンド

コストベース・オプティマイザの効率的な操作に必要不可欠な、表および索引の統計を収集し、格納するためのコマンド。

オブジェクトが 1 度も分析されたことがない場合、EXPLAIN PLAN の生成時にはルールベースの最適化方法が使われる。

ユーザーがオブジェクトを分析する頻度は、オブジェクト内部の変化の比率によって決まる。表を分析する場合、その表に関連付けられた表も自動的に分析される。

このコマンドによって統計が収集される間、データベースの一部分がロックされる。

### EXPLAIN PLAN

問合せオプティマイザによって決定されたアクセス・パスを列挙する SQL 文。

Oracle SQL Analyze は、さまざまなオプティマイザ・モードに基づいた EXPLAIN PLAN を表示し、実行パスのウォークスルーを支援する。

詳細は、2-8 ページの「EXPLAIN PLAN の理解」を参照。

### EXPLAIN PLAN オブジェクト

特定の SQL 文に対して生成された EXPLAIN PLAN を表現する EXPLAIN PLAN ノード。

EXPLAIN PLAN オブジェクトを選択すると、「SQL テキスト」ウィンドウに EXPLAIN PLAN が表示される。

### SQL オブジェクト

「ナビゲータ」ウィンドウに表示される SQL オブジェクトは、自身が接続されているデータベース・セッションに対してチューニングを実行できる特定の SQL 文を表現する。

SQL オブジェクトは作成、コピーまたは削除が可能である。SQL オブジェクト内部の文は、さまざまな方法で編集できる。

ただし、文に対していったん EXPLAIN PLAN が生成されると、そのオブジェクトは読取り専用となるので注意が必要である。読取り専用オブジェクトの編集を続行するには、**「SQL」=>「類似作成」**コマンドを使って、そのノードのコピーを作成する必要がある。

### TopSQL

TopSQL は、Oracle SQL Analyze に統合された機能の 1 つで、SQL 文が消費するリソースの測定を可能にする。この統計を使って、どの文が最も多くのリソースを消費しているかを特定し、チューニングの対象としてその文を選択できる。

TopSQL は、V$SQLAREA からすべての SQL 文および統計を取得する。V$SQLAREA ビューには、共有 SQL 領域上の統計がリストされ、1 つの SQL 文字列ごとに 1 行が含まれる。このビューは、メモリ上に存在し、解析済みで、実行の準備ができている、あるいはすでに実行済みの SQL 文についての統計を提供する。

### TopSQL オブジェクト

「ナビゲータ」ウィンドウに表示される TopSQL オブジェクトは、自身が接続されているデータベース・オブジェクトについて利用可能な TopSQL セッションを表現する。

別のデータベースに接続するたびに、別個の TopSQL オブジェクトが作成される。

### V$SQLAREA

Oracle TopSQL は、V$SQLAREA からの情報を使って、特定の SQL 文によって使われるリソースを決定する。

V$SQLAREA は、全体として見ればインスタンスのビューであり、インスタンスの起動時からの統計、または現在の値の統計を記録する。このビューは、SGA 空間の再割り当てを行う何らかの必要によって変更されるまでは不変のままである。メモリー上に存在し、解析済みで、実行の準備ができている SQL 文についての統計を利用できる。

詳細は、『Oracle Server チューニング』および『Oracle Server 管理者ガイド』を参照。

### 意思決定支援システム（DSS）

意思決定支援アプリケーションまたはデータ・ウェアハウジング・アプリケーションは、大量の情報から分かりやすいレポートを抽出する。一般に、意思決定支援アプリケーションは、OLTP アプリケーションによって収集された大量のデータに対して問合せを実行する。意思決定支援システムにおいて鍵となる目標は、応答時間、精度および可用性である。

意思決定支援システムの例として、人口統計の調査から収集された情報に基づいて、消費者の商品購入パターンを判断するマーケティング・ツールが挙げられる。人口統計のデータは整理され、システムに入力される。市場調査員はこのデータに対して問合せを実行し、どの地域でどの商品が最もよく売れるかを判断する。生成されたレポートは、各地域において購入および販売すべき商品を判断する材料として役立つ。

データ・ウェアハウジング・システムにおいて鍵となる目標は、応答時間、精度および可用性である。

### 応答時間優先のコストベース最適化

コストベース最適化は、文の実行パスを決定するにあたり、表および索引内部のデータの分量および分布についての統計情報を考慮する。次に、最も「コスト」の低い実行パスの選択を試みる。

ここでいう「コスト」とは、（I/O および CPU 消費量などの）コンピュータ・リソースや実行を完了するための時間などの、さまざまな要素の測定値のことである。

応答時間優先の最適化（「すべてのコストを」最適化ともいう）を行う際、データの最初の行が最も効率的に検索される方法で文を実行するように、オプティマイザは選択を行う。

| | |
|---|---|
| **注意：** | コストベース最適化の実行に先立ち、参照されるオブジェクト（表および索引）をあらかじめ分析しておく必要がある。 |

詳細は、3-22 ページの「Oracle オプティマイザの理解」および『Oracle Server チューニング』を参照。

### オブジェクト詳細

オブジェクト詳細は、SQL 文によって参照されるデータ・オブジェクトについての情報を提供する。

これらの詳細には、そのオブジェクトに関係する表、クラスタまたは索引についての情報が含まれる。このような詳細情報を利用して、関連する EXPLAIN PLAN のパフォーマンスについての理解を深めることができる。

### オプティマイザ

Oracle オプティマイザは、SQL 文のコマンドを実行するためにとるべき実行パスを決定する。

利用可能な最適化のモードは次の 4 種類がある。

- ルール
- 応答時間優先のコストベース最適化
- スループット優先のコストベース最適化
- 選択

最適化に使われるモードは、各インスタンスの init.ora ファイル中の OPTIMIZER_MODE パラメータによって設定される。このモードは、ALTER SESSION SET OPTIMIZER_GOAL コマンドを使用するか、文にヒントを追加することにより上書きできる。

### オンライン・トランザクション処理（OLTP）

オンライン・トランザクション処理（OLTP）アプリケーションは、高いスループットを必要とし、挿入 / 更新がきわめて頻繁に行われるシステムである。これらのシステムの特徴に、常に増大し続ける膨大なデータを持つことや、そのデータに数百のユーザーが同時にアクセスすることが挙げられる。典型的な OLTP アプリケーションには、空港の予約システム、注文入力アプリケーション、銀行業務アプリケーションなどがある。OLTP アプリケーションにおいて鍵となる目標は、可用性、速度、並行性および回復可能性である。

オンライン・トランザクション処理（OLTP）アプリケーションは、高いスループットを必要とし、挿入 / 更新がきわめて頻繁に行われるシステムである。これらのシステムの特徴に、常に増大し続ける膨大なデータを持つことや、そのデータに数百のユーザーが同時にアクセスすることが挙げられる。典型的な OLTP アプリケーションには、空港の予約システム、大規模な注文入力アプリケーション、銀行業務アプリケーションなどがある。OLTP アプリケーションにおいて鍵となる目標は、（時として丸 1 週間もの）可用性、速度（スループット）、並行性および回復可能性である。

### カーソル

特定の文に関連付けられたメモリーに対するハンドル（名前またはポインタ）のこと。Oracle Call Interface（OCI）では、これらは文ハンドルと呼ばれる。たとえば、プリコンパイラ・アプリケーションの開発において、カーソルはそのプログラムに対して使用可能な名前付きのリソースであり、そのアプリケーションの内部に埋め込まれた SQL 文の解析のために特別に使用できる。

### 結合方針最適化

結合は、通常共通のキー値に基づいて、複数の表のマージを可能にする操作である。オプティマイザを使って、SQL 文の実行過程における表の結合方針を改良することは、パフォーマンスを改善するための基本的な方法である。

Oracle SQL Analyze では、ヒントを追加したり、チューニング・ウィザードを使って文を分析したりすることにより、結合順序の最適化に影響を与えることができる。

### 構造化問合せ言語（SQL）

構造化問合せ言語（SQL）は、特に関係表へのアクセスをサポートするために開発された、標準のプログラミング言語である。

SQL 言語の一般的な特徴を次に示す。

- 1 行ごとの処理とは対照的な、データの集合に対する操作をサポートする。
- データの物理的位置から独立してデータにアクセスできる。
- 非手続き的である。換言すれば、SQL はデータの検索方法ではなく、検索されるデータのみを記述する言語である。

Oracle7 および Oracle8 のデータベースは、SQL および PL/SQL をサポートする。PL/SQL は、SQL の補完的なインプリメンテーションである。Oracle SQL Analyze は、現時点では標準 SQL のみをサポートする。

### データ定義文（DDL）

DDL 文はオブジェクトを定義およびメンテナンスし、必要がなくなればそのオブジェクトを削除する。DDL 文には、データベースおよびデータベース内部の特定のオブジェクトにアクセスする権限または権利を、あるユーザーが他のユーザーに付与するための文が含まれる。

### データ操作文（DML）

DML 文は、データベースのデータを操作する。たとえば、表中の行の問合せ、挿入、更新または削除はすべて DML の操作である。表またはビューのロック、および SQL 文の EXPLAIN PLAN の検証も、DML の操作に含まれる。

### コンパクト・ビュー

Oracle オプティマイザによって選択された結合の方法論を図示する EXPLAIN PLAN の表現。

### システム・グローバル領域

1 つの Oracle インスタンスに対する、データおよび制御情報を格納する共有メモリー領域。Oracle のインスタンスは、SGA と Oracle のバックグラウンド・プロセスによって構成される。

Oracle はインスタンスの起動時にシステム・グローバル領域の割り当てを行い、インスタンスのシャットダウン時にその割当てを解除する。それぞれのインスタンスに対して、固有のシステム・グローバル領域が存在する。ある時点で Oracle Server に接続されているユーザーは、システム・グローバル領域のデータを共有する。最適なパフォーマンスを得るためには、システム・グローバル領域全体の大きさを（実メモリーの範囲内で）可能な限り広く確保して、メモリー上にできるだけ多くのデータが格納されるようにし、ディスク I/O を減らすことが推奨される。

システム・グローバル領域内部に格納された情報は、何種類かのメモリー構造に分けられる。これにはデータベース・バッファ、REDO ログ・バッファ、共有プールなどが含まれる。これらの領域サイズは固定であり、インスタンス起動の間に作成される。

### 初期化パラメータ・オブジェクト

「ナビゲータ」ウィンドウに表示される初期化パラメータ・オブジェクトは、ナビゲーション・ツリー上で自身が接続されているデータベースの特定のインスタンスを表現する。

初期化パラメータ・オブジェクトを選択することにより、表示されたパラメータを編集して、現在のデータベースの環境とは異なるさまざまな SQL チューニング環境をシミュレートできる。

### スター型変換

スター問合せを効率的に実行することを目的とした、コストベースの問合せ変換。スター最適化は、少数のディメンションと稠密な実表を持つスキーマに対して効果を発揮する。その一方で、スター型変換は、次の条件のいずれかに該当する場合の代替として考慮できる。

- ディメンションの数が多い
- 実表のデータが散在している

- 問合せの中には、すべてのディメンション表が制約述語を持たないものも存在する

スター型変換は、ディメンション表の直積演算の計算に依存しない。このことから、実表がまばらであるため、またはディメンションの数が多いため、あるいはその両方が原因で大規模な直積演算が行われるにもかかわらず、実表の中に実際に一致する行がほとんどないような場合には、スター型変換の方がより適しているといえる。加えて、スター型変換は、連結索引に依存するのではなく、個々の実表列上での結合ビットマップ索引に基づいている。

このためスター型変換では、制約ディメンションに正確に対応する索引の結合を選択できる。さまざまな列順が、さまざまな問合せにおけるさまざまなパターンの制約ディメンションに一致するような、多くの連結索引を作成する必要はない。

スター型変換は、実表に対するビットマップ索引アクセス・パスを駆動するために使用できる、新しい問合せを生成することによって効果を発揮する。

### スター問合せ

データ・ウェアハウスの設計の中には、「スター」スキーマとして知られるものがある。これは通常、1 枚または複数枚のきわめて大きな「実表」と、比較的小さな多数の「ディメンション」または参照表から構成される。スター問合せは、通常、問合せに含まれる述語によって、多数のディメンション表を実表の中の 1 枚に結合する問合せのことである。

Oracle のコストベース最適化はスター問合せを認識し、この種の問合せに対して効率的な EXPLAIN PLAN を生成する。実際、スター問合せを効率的に実行するためには、コストベースの最適化を使用する必要がある。コストベース最適化を有効にするには、単に ANALYZE コマンドを使って表を分析し、オプティマイザ・モードがデフォルト値の CHOOSE に設定されていることを確認する。

### スループット優先のコストベース最適化

コストベース最適化は、文の実行パスを決定するにあたり、表および索引内部のデータの分量および分布についての統計情報を考慮する。次に、最も「コスト」の低い実行パスの選択を試みる。

ここでいう「コスト」とは、（I/O および CPU 消費量などの）コンピュータ・リソースや実行を完了するための時間などの、さまざまな要素の測定値のことである。

スループット優先の最適化を行う際、指定されたすべての行が最も効率的に処理される方法で文を実行するように、オプティマイザは選択を行う。

| | |
|---|---|
| **注意 :** | コストベース最適化の実行に先立ち、参照されるオブジェクト（表および索引）をあらかじめ分析しておく必要がある。 |

詳細は、3-22 ページの「Oracle オプティマイザの理解」および『Oracle Server チューニング』を参照。

### 選択

選択（オプティマイザの選択）を選ぶと、オプティマイザは少なくとも 1 つのオブジェクトが（ANALYZE コマンドを使って）分析されているかどうかを判断する。分析済みのオブジェクトが存在する場合、オプティマイザはスループット優先のコストベース最適化を使用する。分析済みのオブジェクトが存在しない場合は、ルールベースの最適化が使用される。

詳細は、『Oracle Server チューニング』を参照。

### データ・ディクショナリ

関連付けられたデータベースについての情報を提供する、読取り専用の表の集合。データ・ディクショナリが提供できる情報には、次のものがある。

- Oracle ユーザーの名前
- 各ユーザーに付与されている権限およびロール
- スキーマ・オブジェクト（表、ビュー、スナップショット、索引、クラスタ、シノニム、順序、プロシージャ、関数、パッケージ、トリガーなど）の名前
- 統合性制約についての情報
- 列のデフォルト値
- データベース内のオブジェクトへの領域割当て量と、そのオブジェクトによる現在の領域使用量
- 監査情報
- その他の一般的なデータベース情報

データ・ディクショナリは、他のデータベース・データと同様に、表およびビューの形で構成される。データ・ディクショナリは読取り専用であるため、ユーザーはデータ・ディクショナリの表およびビューに対して、問合せ（SELECT 文）だけを発行できる。

### データベース・オブジェクト

アクティブな SQL 文の実行対象となるデータベースを表現する。

データベース・オブジェクトが選択されているとき、そのデータベースの特性の一部を参照できるが、それらを編集することはできない。

### バインド変数

データ値または検索キーなどの SQL 文内部の変数を、SQL 文のパラメータとして定義することを可能にする変数。このアプローチにより、SQL 文を再解析せずに再実行することが可能になる。

SQL 文の EXPLAIN PLAN を生成する間、Oracle SQL Analyze はバインド変数のサンプル値を要求する。値の入力は求められないが、その値は、Oracle SQL Analyze が文および文の環境を評価し、最も理想的な最適化計画を生成するために役立つ。

### ビットマップ索引

ビットマップ索引は標準の索引と同じ機能を提供するが、異なる内部表現を使用する。これにより、検索が非常に高速になり、領域が効率的に利用される。ビットマップ索引は、各キーが従業員の性別などの多数のレコードを参照するときに最も効果を発揮する。

### ヒント

EXPLAIN PLAN の生成時に特定の方法を使うよう、Oracle オプティマイザに指示を送るために SQL コード内に記述する命令。

### プログラム・グローバル領域（PGA）

単一のプロセス（サーバーまたはバックグラウンドの）についてのデータおよび制御情報を格納するメモリー領域。このため、PGA はプログラム・グローバル領域またはプロセス・グローバル領域として参照される。

### 並行度

ある単一の操作に関連付けられたパラレル・サーバー・プロセスの数を並行度という。

並行度は文レベルで（ヒントまたは PARALLEL 句を使って）、あるいは表または索引のレベルで（それらの定義において）指定される。さらに、ディスクまたは CPU の数に基づいたデフォルト値によっても指定される。

並行度は、操作内並行性だけに直接適用されることに注意が必要である。操作内の並行処理が可能な場合、1 つの文に対するパラレル・サーバー・プロセスの総数は、指定された並行度の 2 倍に達する場合がある。2 つを超える操作を同時に実行することはできない。

### リポジトリ

Oracle SQL Analyze のリポジトリには、SQL のチューニング・セッションに関する次のような情報が格納される。

- そのセッションのナビゲーション・ツリーを構成するために必要な情報
- データベースおよびセッションのパラメータ
- SQL 文
- EXPLAIN PLAN および関連の統計

オブジェクト詳細は、リポジトリには保存されない。

保存されたリポジトリには随時戻ることができ、保存された時点からチューニング・セッションを続行できる。

### ルール

SQL 文に適用するためのガイドの集合で、文の構文を改善し、文をより効率的なものにする。これらのガイドは、オラクル社のデータベース専門家によって長年にわたり培われてきたものである。

チューニング・ウィザードを通じて SQL 文を分析することにより、Oracle SQL Analyze を使って文にルールを適用できる。

ルールの詳細と適用方法については、2-16 ページの「ルールの理解」を参照。

### ルールベース最適化

ルールベース最適化が選択されているとき、Oracle オプティマイザは構文上の規則の集合と、さまざまなアクセス・パスの順位付けに基づいて、SQL 文を実行する。このとき、表および索引内部のデータの分量および分布に関連する統計情報は考慮されない。

コストベースの最適化は新機能および強化された機能をサポートするため、新規アプリケーションおよびデータ・ウェアハウジング・アプリケーションに対しては、ほとんどの場合、コストベース最適化の方が望ましいアプローチである。

一般に、ルールベース最適化は次のアプローチを採用する。

1. WHERE 句の中の各表に対して、オプティマイザは考えられるすべてのアクセス・パスを考慮し、順位を付ける。
2. オプティマイザは最も順位の低いアクセス・パスを選択する。
3. オプティマイザは、残りの各表に対して考えられる、すべてのアクセス・パスに順位を付ける。
4. オプティマイザは最も順位の低いアクセス・パスを選択する。
5. すべての表が結合されるまで、このプロセスを繰り返す。

詳細は、2-8 ページの「EXPLAIN PLAN の理解」および『Oracle Server チューニング』を参照。

### 連鎖行

列のヘッダを格納するデータ・ブロックにその列が収まらなくなったとき、その列は残りのデータを別のブロックまたはブロックの集合に格納できる。そのような行を連鎖行という。連鎖行が発生すると、単一の行を読み込むためにより多くのブロックを読み込む必要が生じるため、パフォーマンスが低下する場合がある。

# 索引

## 数字



## A



## C



## E



## H



## N



## O



## S



## T



## U

## さ



## し



## す



## せ



## そ



## た



## ち



## つ



## て



## と



## な

## め



## り



## る



## れ



## わ