

Oracle Text

アプリケーション開発者ガイド

リリース 9.2

2002 年 7 月

部品番号 : J06301-01

ORACLE®

Oracle Text アプリケーション開発者ガイド, リリース 9.2

部品番号 : J06301-01

原本名 : Oracle Text Application Developer's Guide, Release 9.2

原本部品番号 : A96517-01

原本著者 : Colin McGregor

原本協力者 : Omar Alonso, Shamim Alpha, Steve Buxton, Chung-Ho Chen, Yun Cheng, Michele Cyran, Paul Dixon, Mohammad Faisal, Elena Huang, Garrett Kaminaga, V. Jegrag, Ji Sun Kang, Bryn Llewellyn, Wesley Lin, Yasuhiro Matsuda, Gerda Shank, and Steve Yang

Copyright © 2000, 2002 Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、**Oracle Corporation**（米国オラクル）または**日本オラクル株式会社**（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である **Oracle Corporation**（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『**Restricted Rights**』と共に提供してください。この場合次の **Notice** が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	ix
------------	----

1 Oracle Text の概要

Oracle Text の概要	1-2
問合せアプリケーションのタイプ	1-2
サポートされているドキュメント形式	1-3
テーマ機能	1-3
問合せ言語と演算子	1-4
ドキュメント・サービスおよびシソーラスの使用	1-4
問合せアプリケーション作成の前提条件	1-4
テキスト表のロードの概要	1-5
テキスト表へのテキストの格納	1-7
ファイルのパス名の格納	1-7
URL の格納	1-7
関連するドキュメント情報の格納	1-7
サポートされている列型	1-8
サポートされているドキュメント形式	1-8
ロード方法	1-8
ドキュメントの索引付け	1-9
索引のタイプ	1-10
CONTEXT 索引を作成する場合	1-11
CTXCAT 索引を作成する場合	1-12
CTXRULE 索引を作成する場合	1-13
索引のメンテナンス	1-13
単純なテキスト問合せアプリケーション	1-14

索引の問合せ方法の理解	1-15
CONTAINS を使用した問合せ方法	1-15
構造化フィールドの検索	1-17
シソーラスを使用した問合せ	1-17
ドキュメントのセクション検索	1-17
その他の問合せ機能	1-18
ヒットリストの表示	1-19
ヒットリストの例	1-19
構造化フィールドの表示	1-21
ヒットリストの順序付け	1-21
ドキュメントのヒット件数の表示	1-21
ドキュメントの表示方法およびハイライト表示	1-22
ハイライト表示の例	1-23
ドキュメントのテーマ・リストの例	1-24
要旨の例	1-25

2 索引付け

Oracle Text の索引の概要	2-2
Oracle Text の CONTEXT 索引の構造	2-3
Oracle Text の索引付け処理	2-3
パーティション表とパーティション索引	2-6
オンラインでの索引の作成	2-6
パラレル索引付け	2-7
索引付けの制限事項	2-8
索引付け時の考慮事項	2-8
索引のタイプ	2-8
テキストの場所	2-11
ドキュメント形式とフィルタ処理	2-12
索引付け時の行のバイパス	2-13
ドキュメントのキャラクタ・セット	2-13
ドキュメントの言語	2-13
特殊文字の索引付け	2-14
大 / 小文字を区別した索引付けおよび問合せ	2-15
各国語別の機能	2-16
ファジー・マッチングおよびステミング	2-18

ワイルド・カード問合せのパフォーマンスの向上	2-18
ドキュメントのセクション検索	2-19
ストップワードとストップテーマ	2-19
索引のパフォーマンス	2-20
問合せのパフォーマンスと LOB 列の格納	2-20
索引の作成	2-20
CONTEXT 索引の作成手順	2-20
プリファレンスの作成	2-21
セクション検索のためのセクション・グループの作成	2-25
ストップワードおよびストップリストの使用	2-26
索引の作成	2-27
CONTEXT 索引の作成	2-27
CTXCAT 索引の作成	2-29
CTXRULE 索引の作成	2-33
索引のメンテナンス	2-34
索引エラーの表示	2-34
索引の削除	2-35
失敗した索引付けの再開	2-35
索引の再構築	2-35
プリファレンスの削除	2-36
CONTEXT 索引に関する DML 操作の管理	2-36
保留中の DML の表示	2-36
索引の同期化	2-37
索引の最適化	2-38

3 問合せ

問合せの概要	3-2
CONTAINS による問合せ	3-2
CATSEARCH による問合せ	3-4
MATCHES による問合せ	3-5
ワード問合せと句問合せ	3-7
ABOUT 問合せおよびテーマ	3-8
問合せ式	3-8
大 / 小文字を区別した検索	3-10
問合せのフィードバック	3-10

問合せの実行計画	3-11
CONTEXT 文法	3-11
ABOUT 問合せ	3-12
論理演算子	3-12
セクション検索	3-13
NEAR 演算子による近接問合せ	3-13
FUZZY、STEM、SOUNDEX、ワイルド・カードおよびシソーラスの拡張演算子	3-14
CTXCAT 文法の使用法	3-14
ストアド・クエリー式	3-14
CONTAINS での PL/SQL ファンクションのコール	3-15
CTXCAT 文法	3-16
CATSEARCH での CONTEXT 文法の使用	3-16
応答時間短縮のための最適化	3-17
問合せの応答時間に影響するその他の要因	3-17
ヒット数のカウント	3-18
SQL によるヒット数のカウント例	3-18
構造化述語によるヒット数のカウント	3-18
PL/SQL によるヒット数のカウント例	3-18

4 ドキュメントの表示方法

問合せ語句のハイライト表示	4-2
テキストのハイライト表示	4-2
テーマのハイライト表示	4-2
CTX_DOC のハイライト表示プロシージャ	4-2
テーマのリスト、要旨およびテーマ・サマリーの取得	4-3
テーマのリスト	4-4
要旨およびテーマ・サマリー	4-5

5 パフォーマンス・チューニング

統計を使用した問合せの最適化	5-2
統計の収集	5-2
統計の再収集	5-4
統計の削除	5-4
応答時間短縮のための問合せの最適化	5-4
問合せの応答時間に影響するその他の要因	5-5

FIRST_ROWS(n) による ORDER BY 問合せの応答時間の短縮	5-5
ローカル・パーティション CONTEXT 索引を使用した応答時間の短縮	5-7
スコア順のローカル・パーティション索引を使用した応答時間の短縮	5-8
スループット向上のための問合せの最適化	5-9
CHOOSE および ALL ROWS モード	5-9
FIRST_ROWS モード	5-9
パラレル問合せ	5-10
ブロック操作による問合せのチューニング	5-11
問合せのパフォーマンスに関する FAQ (よくある質問)	5-12
問合せのパフォーマンスとは何を意味しますか?	5-12
テキスト問合せのうち、最速のタイプはどれですか?	5-12
表に関する統計を収集する必要がありますか?	5-12
データのサイズは問合せにどのように影響しますか?	5-12
データの形式は問合せにどのように影響しますか?	5-13
機能的検索と索引付き検索とは、何を意味しますか?	5-13
問合せで使用する表はどれですか?	5-13
テキスト条件のみの検索速度は、結果をソートすることにより低下しますか?	5-14
スコアによる ORDERBY 順の問合せを速く行うには、どうすればよいですか?	5-14
どのメモリ設定が問合せに影響しますか?	5-14
元表の LOB 列を表外に格納すると、パフォーマンスが向上しますか?	5-15
複数の列に対する CONTAINS 問合せを高速にするには、どうすればよいですか?	5-15
問合せに多数の拡張を使用してもかまいませんか?	5-16
ローカル・パーティション索引はどのような場合に便利ですか?	5-17
問合せはパラレルに実行する方がよいですか?	5-17
テーマには索引を付けた方がよいですか?	5-17
CTXCAT 索引はどのような場合に作成するとよいですか?	5-18
CTXCAT 索引が適さないのは、どのような場合ですか?	5-18
使用可能なオプティマイザ・ヒントは何ですか。また、その機能はどのようなものですか?	5-18
索引付けのパフォーマンスに関する FAQ (よくある質問)	5-19
索引付けにはどのくらいの時間が必要ですか?	5-19
どの索引メモリ設定を使用すればよいですか?	5-19
索引付けはどの程度のディスク・オーバーヘッドが必要ですか?	5-20
データの形式によって索引付けにどのような影響がありますか?	5-20
索引付けはパラレルに実行できますか?	5-21
ローカル・パーティション索引をパラレルに作成するには、どうすればよいですか?	5-21

索引付けの進捗を確認するには、どうすればよいですか？	5-22
索引の更新に関する FAQ（よくある質問）	5-22
新規レコードまたは更新済みレコードに索引を付ける頻度は、どれくらいがよいですか？	5-22
索引の断片化は、どのようにするとわかりますか？	5-23
メモリー割当ては索引の同期化に影響しますか？	5-23

6 ドキュメントのセクション検索

ドキュメントのセクション検索	6-2
セクション検索の使用可能化	6-2
セクションのタイプ	6-5
HTML のセクション検索	6-10
HTML のセクションの作成	6-10
HTML の Meta タグの検索	6-11
XML のセクション検索	6-11
自動セクション	6-12
属性の検索	6-12
ドキュメント・タイプ別のセクションの作成	6-13
パス・セクション検索	6-14

7 シソーラスの使用

シソーラスの概要	7-2
シソーラスの作成とメンテナンス	7-2
大 / 小文字を区別するシソーラス	7-3
大 / 小文字を区別しないシソーラス	7-3
デフォルトのシソーラス	7-4
提供されるシソーラス	7-4
シソーラスの用語の定義	7-5
シノニムの定義	7-5
階層関係の定義	7-6
問合せアプリケーションでのシソーラスの使用	7-6
カスタム・シソーラスのロードおよびシソーラス問合せの発行	7-7
カスタム・シソーラスによるナレッジ・ベースの補強	7-7
提供されるナレッジ・ベース	7-9
言語固有のナレッジ・ベースの追加	7-10

8 管理

Oracle Text のユーザーとロール	8-2
CTXSYS ユーザー	8-2
CTXAPP ロール	8-2
ユーザーへのロールおよび権限の付与	8-2
DML キュー	8-3
CTX_OUTPUT パッケージ	8-3
サーバー	8-3
管理ツール	8-3

A CONTEXT 問合せアプリケーション

Web 問合せアプリケーションの概要	A-2
PSP Web アプリケーション	A-2
Web アプリケーションの前提条件	A-3
Web アプリケーションの作成	A-3
PSP のサンプル・コード	A-5
loader.ctl	A-6
loader.dat	A-6
search_htmlservices.sql	A-7
search_html.psp	A-9
JSP Web アプリケーション	A-11
Web アプリケーションの前提条件	A-11
JSP のサンプル・コード : search_html.jsp	A-11

B CATSEARCH 問合せアプリケーション

CATSEARCH Web 問合せアプリケーションの概要	B-2
JSP Web アプリケーション	B-2
JSP Web アプリケーションの作成	B-2
JSP のサンプル・コード	B-5
loader.ctl	B-5
loader.dat	B-5
catalogSearch.jsp	B-6

索引

はじめに

このマニュアルでは、Oracle Text を使用した問合せアプリケーションの作成方法を説明します。ここでは、次の項目について説明します。

- [対象読者](#)
- [このマニュアルの構成](#)
- [関連文書](#)
- [表記規則](#)

対象読者

『Oracle Text アプリケーション開発者ガイド』は、次の作業を行うユーザーを対象としています。

- Oracle Text アプリケーションの開発
- Oracle Text のインストールの管理

このマニュアルを活用するには、Oracle オブジェクト・リレーショナル・データベース管理システム、SQL、SQL*Plus および PL/SQL での実践経験が必要です。

このマニュアルの構成

このマニュアルの構成は、次のとおりです。

第1章「Oracle Text の概要」

この章では、Oracle Text の基本機能について説明します。また、Oracle Text を使用した基本的な問合せアプリケーションの作成方法についても説明します。

第2章「索引付け」

この章では、ドキュメント・セットの索引付け方法を説明します。CONTEXT、CTXCAT および CTXRULE の各索引の作成方法と索引付けに関する考慮事項を説明します。

第3章「問合せ」

この章では、ドキュメント・セットの問合せ方法を説明します。CONTAINS、CATSEARCH および MATCHES の各演算子の使用方法の例を示します。

第4章「ドキュメントの表示方法」

この章では、問合せアプリケーションに対するドキュメントの表示方法を説明します。

第5章「パフォーマンス・チューニング」

この章では、応答時間およびスループットの改善に必要な問合せのチューニング方法を説明します。

第6章「ドキュメントのセクション検索」

この章では、HTML と XML でセクション検索を使用可能にする方法を説明します。

第7章「シソーラスの使用」

この章では、使用中のアプリケーションでシソーラスを使用して作業する方法を説明します。また、シソーラスでナレッジ・ベースを補強する方法についても説明します。

第 8 章「管理」

この章では、Oracle Text の管理について説明します。

付録 A「CONTEXT 問合せアプリケーション」

この付録では、Oracle Text の CONTEXT サンプル Web アプリケーションを示します。

付録 B「CATSEARCH 問合せアプリケーション」

この付録では、Oracle Text の CATSEARCH サンプル Web アプリケーションを示します。

関連文書

Oracle Text の詳細は、次のマニュアルを参照してください。

- 『Oracle Text リファレンス』

Oracle9i の詳細は、次のマニュアルを参照してください。

- 『Oracle9i データベース概要』
- 『Oracle9i データベース管理者ガイド』
- 『Oracle9i データベース・ユーティリティ』
- 『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』
- 『Oracle9i SQL リファレンス』
- 『Oracle9i データベース・リファレンス』
- 『Oracle9i アプリケーション開発者ガイド - 基礎編』
- 『Oracle9i XML データベース開発者ガイド - Oracle XML DB』

PL/SQL の詳細は、次のマニュアルを参照してください。

- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパー、その他の関連文書は、OTN-J (Oracle Technology Network Japan) に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

<http://otn.oracle.co.jp/membership/>

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

<http://otn.oracle.co.jp/document/>

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- 本文の表記規則
- コード例の表記規則

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に記載されている用語、あるいはその両方を示します。	ub4 、 sword または OCINumber などの C データ型が有効です。 この句を指定する場合は、 索引構成表 を作成します。
イタリック	イタリックは、問合せ語句、構文の句またはプレースホルダを示します。	<i>parallel_clause</i> を指定できます。 Uold_release .SQL を実行します。 <i>old_release</i> はアップグレード前にインストールしたリリースを指します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、 Recovery Manager キーワード、 SQL キーワード、 SQL*Plus またはユーティリティ・コマンド、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、データベースのバックアップを作成できます。 データ・ディクショナリ・ビューの USER_TABLES 表にある TABLE_NAME 列を問い合わせます。 ROLLBACK_SEGMENTS パラメータを指定します。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。

規則	意味	例
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイルとサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。	sqlplus と入力して、SQL*Plus をオープンします。 hr.departments 表には、department_id、department_name および location_id の列があります。 初期化パラメータ QUERY_REWRITE_ENABLED を true に設定します。 oe ユーザーで接続します。

コード例の表記規則

コード例では、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、コード例の表記規則と使用例を示しています。

規則	意味	例
[]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (digits [, precision])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE DISABLE}
	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE DISABLE} [COMPRESS NOCOMPRESS]
...	水平の省略記号は、次のどちらかを示します。 <ul style="list-style-type: none">■ 例に直接関係のないコード部分が省略されていること。■ コードの一部が繰り返し可能であること。	CREATE TABLE ...AS subquery; SELECT col1, col2, ...,coln FROM employees;
.	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	

規則	意味	例
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、表示されているとおりに入力してください。	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
イタリック	イタリックの文字は、特定の値を指定する必要がある変数を示します。	<pre>CONNECT SYSTEM/system_password</pre>
大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr</pre>

Oracle Text の概要

この章では、**Oracle Text** の主な機能について説明します。この章は、索引付け、問合せおよびドキュメントの表示方法についてのスタート・ガイドです。

次の項目について説明します。

- [Oracle Text の概要](#)
- [テキスト表のロードの概要](#)
- [ドキュメントの索引付け](#)
- [単純なテキスト問合せアプリケーション](#)
- [索引の問合せ方法の理解](#)
- [ヒットリストの表示](#)
- [ドキュメントの表示方法およびハイライト表示](#)

Oracle Text の概要

Oracle Text は、テキスト問合せアプリケーションおよびドキュメント分類アプリケーションを作成できるツールです。Oracle Text には、テキストに対する索引付け機能、ワードとテーマの検索機能および結果表示機能が備わっています。

問合せアプリケーションのタイプ

Oracle Text を使用すると、次の 2 つのタイプのアプリケーションを作成できます。

- テキスト問合せアプリケーション
- ドキュメント分類アプリケーション

テキスト問合せアプリケーション

テキスト問合せアプリケーションの目的は、検索語句を含むテキスト文書を検索できるようにすることです。テキストは通常ドキュメントの集まりです。優れたアプリケーションでは、プレーン・テキスト、HTML、XML または Microsoft Word などの一般的なドキュメント形式を索引付けおよび検索できます。たとえば、ブラウザ・インタフェースを備えたアプリケーションでは、HTML ファイルで構成された企業の Web サイトに問合せを行い、問合せに一致するファイルを戻すことができます。

テキスト問合せアプリケーションを作成する場合、CONTEXT 索引を作成して CONTAINS 演算子を使用するか、あるいは CTXCAT 索引を作成して CATSEARCH 演算子を使用することで、その索引を問い合わせることができます。

関連項目： これらの索引の詳細は、この章の「[ドキュメントの索引付け](#)」を参照してください。

ドキュメント分類アプリケーション

ドキュメント分類アプリケーションは、ドキュメントの内容に基づいてドキュメントの着信ストリームを分類するアプリケーションです。このアプリケーションは、ドキュメント・ルーティング・アプリケーションまたはドキュメント・フィルタ処理アプリケーションとも呼ばれます。たとえば、オンライン通信社では、記事の着信時にその着信ストリームを政治、犯罪、スポーツなどのカテゴリに分類する必要があります。

Oracle Text では、CTXRULE 索引タイプを使用して、このような分類アプリケーションを構築できます。この索引によって、各クラスを定義するルール（問合せ）を索引付けします。ドキュメントの着信時には、MATCHES 演算子によって、各ドキュメントとそのドキュメントを選択するルールを照合できます。

ドキュメント・セットに対してルールを生成するには、CTX_CLS.TRAIN プロシージャを使用できます。その後、このプロシージャの出力を使用して、CTXRULE 索引を作成できます。

関連項目： CTX_CLS.TRAIN の詳細は、『Oracle Text リファレンス』を参照してください。

注意： Oracle Text では、プレーン・テキスト、HTML および XML のドキュメントに対してのみドキュメント分類をサポートしています。

サポートされているドキュメント形式

テキスト問合せアプリケーションの場合、Oracle Text では、プレーン・テキスト、HTML、および Microsoft Word のように書式設定されたドキュメントも含めて、ほとんどのドキュメント形式に対する索引付けや問合せをサポートしています。

ドキュメント分類アプリケーションの場合は、プレーン・テキスト、XML および HTML のドキュメントの分類をサポートしています。

テーマ機能

Oracle Text では、使用言語が英語やフランス語の場合は、ドキュメント・テーマで検索できます。ドキュメント・テーマで検索するには、CONTAINS 問合せに ABOUT 演算子を使用します。たとえば、*politics*（政治）という概念に関連するすべてのドキュメントを検索できます。この場合、*elections*（選挙）、*governments*（政府）または *foreign policy*（外交政策）に関するドキュメントが戻る可能性があります。そのドキュメントに、ワード *politics* が含まれていなくても、ヒットします。

テーマ情報は、提供されているナレッジ・ベースから導出されます。このナレッジ・ベースには、カテゴリと概念が階層式にリストされています。提供されているナレッジ・ベースには一般的な概念しか含まれていないため、後から、業界固有の新しい概念を追加することができます。ナレッジ・ベースを補強すると、ドキュメント・テーマがよりインテリジェントに処理されるため、テーマ検索の精度が向上します。

提供されている PL/SQL パッケージを使用すると、ドキュメント・テーマをプログラムで取得できます。

関連項目： ABOUT 演算子の詳細は、『Oracle Text リファレンス』を参照してください。

その他の言語のテーマ

英語やフランス語以外の言語で、ABOUT 問合せなどのテーマ機能を使用可能にするには、その言語固有のナレッジ・ベースをロードします。

関連項目： [第7章「シソーラスの使用」の「言語固有のナレッジ・ベースの追加」](#)

問合せ言語と演算子

問合せには、SQL の SELECT 文を使用します。索引に応じて、CONTAINS 演算子（CONTEXT 索引の場合）か CATSEARCH 演算子（CTXCAT 索引の場合）のいずれかを使用してテキストの問合せができます。

この 2 つの演算子は、SELECT 文の WHERE 句で使用します。たとえば、ワード *oracle* を含むすべてのドキュメントを検索するには、CONTAINS を次のように使用します。

```
SELECT SCORE(1), title FROM news WHERE CONTAINS(text, 'oracle', 1) > 0;
```

単一のドキュメントの分類には、MATCHES 演算子と CTXRULE 索引を併用します。

CONTAINS 演算子によるテキスト問合せの場合、Oracle Text では、複数の演算子による豊富な問合せ言語を使用できます。これによって、単純なワード問合せ、ABOUT 問合せ、論理問合せ、ワイルド・カードおよびシソーラスを使用した拡張問合せなどの多様な問合せを発行できます。

CATSEARCH 演算子でも、CONTAINS で使用可能な演算の一部をサポートしています。

関連項目： [第 3 章「問合せ」](#)

ドキュメント・サービスおよびシソーラスの使用

提供されている Oracle Text の PL/SQL パッケージを使用すると、ドキュメント表示やシソーラスのメンテナンスなどの高度な機能を利用できます。ドキュメント表示とは、アプリケーションが問合せ結果セットをユーザー・ドキュメントに表示する方法です。シソーラスをメンテナンスすると、問合せおよびアプリケーションを拡張できます。

関連項目：

- [第 7 章「シソーラスの使用」](#)
- [第 4 章「ドキュメントの表示方法」](#)

問合せアプリケーション作成の前提条件

Oracle Text の問合せアプリケーションを作成するには、次のものが存在している必要があります。

- 移入済みのテキスト表
- Oracle Text の索引

次の各項では、これらの前提条件を説明し、一般的なテキスト問合せアプリケーションの主な機能についても説明します。

テキスト表のロードの概要

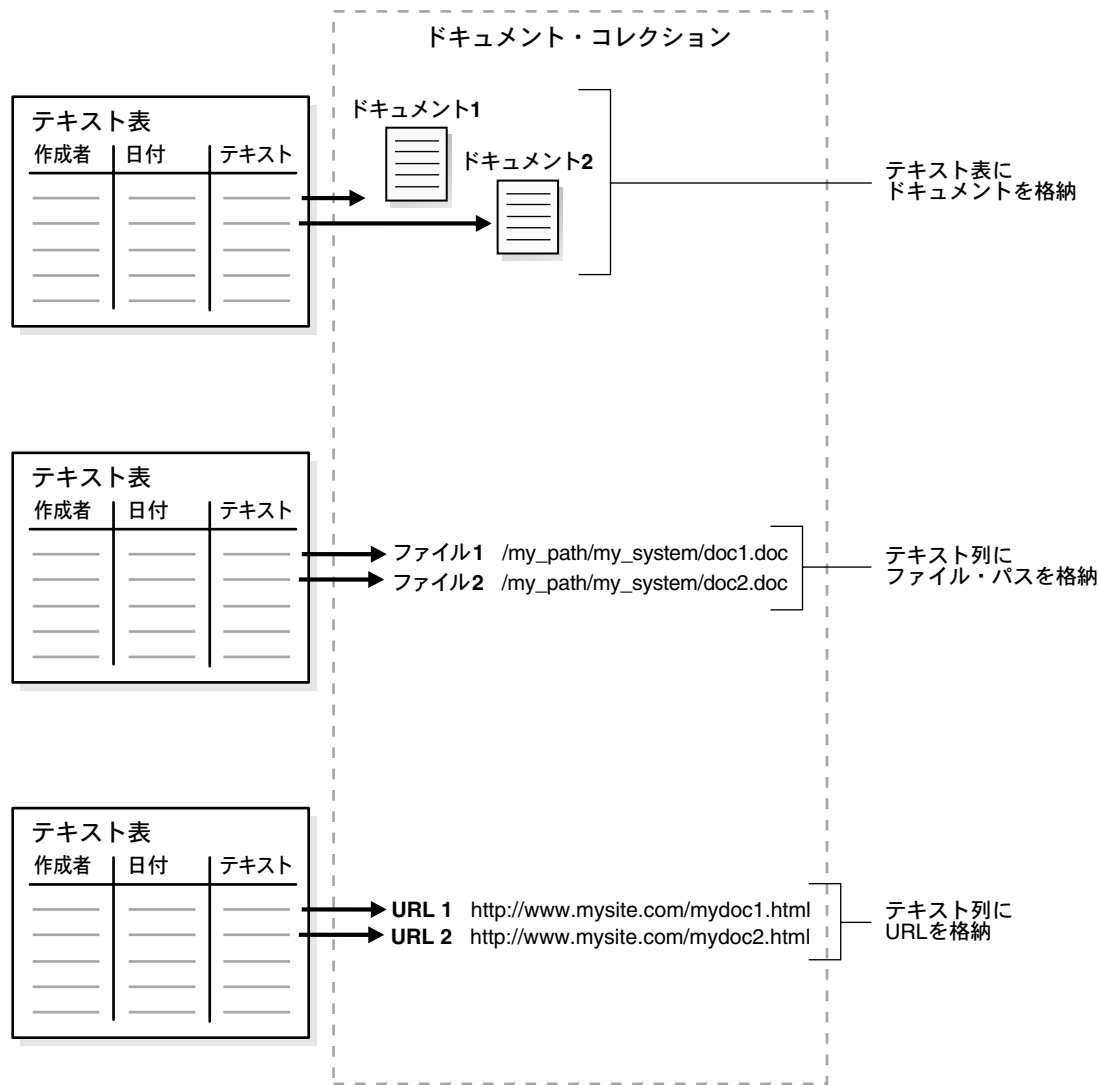
Oracle Text の問合せアプリケーションの基本前提条件は、移入済みのテキスト表が存在していることです。テキスト表には、ドキュメント・コレクションに関する情報が格納されます。テキスト表は、索引付けをする際に必要になります。

次の要素のいずれかを使用して、テキスト表に行を移入できます。

- テキスト情報（ドキュメントまたはテキスト断片）
- ファイル・システムにあるドキュメントのパス名
- World Wide Web のドキュメントを指定する URL

図 1-1 「様々なテキスト格納方法」は、これらの方法を示しています。

図 1-1 様々なテキスト格納方法



デフォルトでは、索引付け操作は、ドキュメント・テキストが直接テキスト表にロードされているものとして動作します。これが、前述の 1 番目の方法です。

ファイル名や URL からなるドキュメントについては、それぞれに対応するデータ記憶域索引プリファレンスを使用することにより、識別方法を変更できます。

テキスト表へのテキストの格納

ドキュメントは様々な方法でテキスト表に格納できます。

`DIRECT_DATASTORE` データ格納タイプを使用すると、1つの列にドキュメントを格納できます。また、`MULTI_COLUMN_DATASTORE` タイプを使用すると、複数の列にドキュメントを格納できます。複数の列にテキストが格納されている場合、Oracle は索引付けを行うためにこれらの列を連結して仮想ドキュメントを作成します。

また、ドキュメントのマスター / デティール関係を作成し、1つのドキュメントを複数の行に格納することもできます。マスター / デティール索引を作成するには、`DETAIL_DATASTORE` データ格納タイプを使用します。

テキスト表には、複数列にまたがって、名前、説明およびアドレスといったテキスト断片を格納し、`CTXCAT` 索引を作成することもできます。`CTXCAT` 索引は、複合問合せのパフォーマンスを改善します。

また、`NESTED_DATASTORE` タイプを使用して、ネストした表にテキストを格納できます。

Oracle Text では、XML ドキュメントの格納に使用できる `XMLType` データ型の索引付けをサポートしています。

ファイルのパス名の格納

テキスト表には、ファイル・システムに格納されているファイルのパス名を格納できます。ファイルのパス名を格納する場合は、索引付け時に、`FILE_DATASTORE` プリファレンス型を使用します。

URL の格納

URL 名を格納して、Web サイトを索引付けすることができます。URL 名を格納する場合は、索引付け時に、`URL_DATASTORE` プリファレンス型を使用します。

関連するドキュメント情報の格納

テキスト表に列を追加すると、問合せアプリケーションに必要な、主キー、日付、説明または作成者などの構造化情報を格納できます。

形式列とキャラクタ・セット列

ドキュメントが複合形式または複合キャラクタ・セットの場合は、次の列を追加できます。

- 索引付け時のフィルタ処理に必要な形式 (`TEXT` または `BINARY`) を記録する形式列。形式列を `IGNORE` に設定すると、索引付け時に無視する行を指定できます。この機能は、テキスト索引付けと互換性のないイメージなどのデータが含まれた行をバイパスする場合に役立ちます。

- 行ごとにドキュメントのキャラクタ・セットを記録するキャラクタ・セット列。

索引の作成時に、形式列またはキャラクタ・セット列の名前を、CREATE INDEX の PARAMETERS 句に指定する必要があります。

サポートされている列型

Oracle Text では、VARCHAR2、CLOB、BLOB、CHAR、BFILE、XMLType および URIType 型の列を使用して、CONTEXT 索引を作成できます。

注意： NCLOB、DATE および NUMBER の列型は索引付けできません。

サポートされているドキュメント形式

システムでは、HTML、PDF、Microsoft Word およびプレーン・テキストを含むほとんどのドキュメント形式を索引付けできるため、ユーザーはサポートされている任意のドキュメント・タイプをテキスト列にロードできます。

テキスト列に複合形式がある場合は、必要に応じて、索引付け時のフィルタ処理に役立つ形式列を組み込むことができます。その形式列を使用して、ドキュメントを BINARY（形式設定済み）または TEXT（形式未設定、HTML など）のいずれかに指定できます。

関連項目： サポートされているドキュメント形式の詳細は、『Oracle Text リファレンス』を参照してください。

ロード方法

次の各項では、情報をテキスト列にロードする方法を説明します。

INSERT 文を使用したテキストのロード

SQL の INSERT 文を使用すると、テキストを表にロードできます。

次の例では、CREATE TABLE 文を使用して、id 列および text 列を持つ表を作成します。この例では、id 列を主キーとしています。text 列は、VARCHAR2 です。

```
CREATE TABLE docs (id NUMBER PRIMARY KEY, text VARCHAR2(80));
```

この表への移入には、次の INSERT 文を使用します。

```
INSERT into docs values(1, 'this is the text of the first document');
INSERT into docs values(12, 'this is the text of the second document');
```


ファイル・システムからのテキストのロード

INSERT 文に加え、Oracle では次のような自動化された方法を使用して、ファイル・システムから表にテキスト・データ（ドキュメント、ドキュメントへのポインタおよび URL など）をロードできます。

- SQL*Loader
- BFILE から LOB をロードするための PL/SQL プロシージャ
DBMS_LOB.LOADFROMFILE()
- Oracle Call Interface

関連項目：

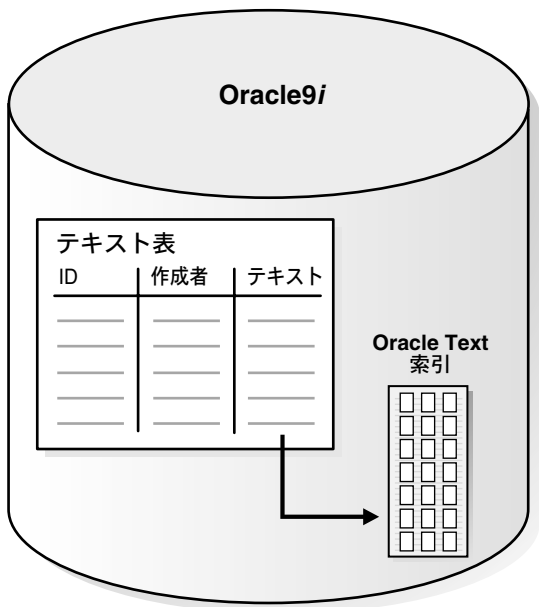
- SQL*Loader の例は、[付録 A 「CONTEXT 問合せアプリケーション」](#)を参照してください。
- DBMS_LOB パッケージの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- LOB の使用方法の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
- Oracle Call Interface の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

ドキュメントの索引付け

ドキュメント・コレクションを問い合わせる場合は、最初にテキスト表のテキスト列を索引付けする必要があります。索引付け操作によってテキストはトークンに分解されます。トークンは、通常、空白で区切られたワードです。トークンは、数字、頭字語、およびドキュメント内で空白により区切られているその他の文字列である可能性もあります。

CONTEXT 索引は、各トークンとそれが含まれているドキュメントを記録します。このような逆向きの索引は、ワードおよび句の問合せに使用できます。[図 1-2](#) は、Oracle9i とそれに関連付けられた Oracle Text 索引を示しています。

図 1-2 テキスト表および関連する Oracle Text 索引



索引のタイプ

Oracle Text では、アプリケーションとテキスト・ソースに応じて、3つのタイプの索引の作成をサポートしています。CREATE INDEX 文を使用して、Oracle Text のすべての索引タイプを作成します。

次の表では、この3つの索引とそれを使用して作成できるアプリケーションのタイプを説明します。第3列では、索引と併用する問合せ演算子を示しています。

索引のタイプ	説明	問合せ演算子
CONTEXT	テキストが大量のまとまったドキュメントで構成されている場合、テキスト検索アプリケーションを作成するには、この索引を使用します。Microsoft Word、HTML、XML またはプレーン・テキストなどの多様な形式のドキュメントを索引付けできます。 索引は様々な方法でカスタマイズできます。	CONTAINS

索引のタイプ	説明	問合せ演算子
CTXCAT	複合問合せのパフォーマンスを改善するには、この索引タイプを使用します。この索引タイプは、複数の列に格納されている日付、項目名および価格などの構造化基準を使用して、小さいテキスト断片を問い合わせるのに適しています。	CATSEARCH
CTXRULE	<p>ドキュメント分類アプリケーションの作成に使用します。この索引は、問合せ表に対して作成する索引です。問合せ表の各行には、ドキュメントの分類方法が格納されています。</p> <p>単一のドキュメント（プレーン・テキスト、HTML または XML）は、MATCHES 演算子を使用して分類できます。</p>	MATCHES

CONTEXT 索引を作成する場合

テキスト・データを表にロードした後、CREATE INDEX 文を使用して CONTEXT 索引を作成できます。索引の作成時に PARAMETERS 句を指定しないと、索引はデフォルトのパラメータを使用して作成されます。

たとえば、次のコマンドによって、docs 表内の text 列に myindex という CONTEXT 索引がデフォルトのパラメータを使用して作成されます。

```
CREATE INDEX myindex ON docs(text) INDEXTYPE IS CTXSYS.CONTEXT;
```

すべての言語に対するデフォルト

CREATE INDEX を使用して、パラメータを明示的に指定せずに CONTEXT 索引を作成すると、すべての言語に対するシステムのデフォルト動作は、次のようになります。

- 索引付けされるテキストは、テキスト列に直接格納されるとみなします。テキスト列の型は、CLOB、BLOB、BFILE、VARCHAR2、XMLType または CHAR のいずれでもかまいません。
- 列型を検出し、バイナリの列型に対してフィルタ処理を行います。ほとんどのドキュメント形式に対してフィルタ処理がサポートされています。列がプレーン・テキストで構成されている場合、システムはこの列に対してフィルタ処理を行います。

注意： システムで正しくドキュメントをフィルタ処理するために、使用している環境が **Inso** フィルタをサポートするように正しく設定されていることを確認してください。

Inso フィルタを使用するための環境の構成方法は、『Oracle Text リファレンス』を参照してください。

- 索引付けするテキストの言語は、データベース設定で指定した言語であるとみなします。
- データベース設定で指定した言語に対するデフォルトのストップリストを使用します。ストップリストは、索引付け時にシステムが無視するワードを識別します。
- 使用言語でのファジーおよびステミング問合せを有効にします（その言語に対してこの機能が使用可能な場合）。

デフォルトの索引付け動作は、ユーザー独自のプリファレンスを作成し、このカスタム・プリファレンスを **CREATE INDEX** の **PARAMETERS** 句に指定することによって、いつでも変更できます。

CONTEXT 索引のカスタマイズ

CREATE INDEX で **PARAMETERS** 句を使用すると、**CONTEXT** 索引をカスタマイズできます。たとえば、**PARAMETERS** 句で、テキストの格納場所、索引付け対象テキストのフィルタ処理方法およびセクション作成の有無を指定できます。

たとえば、テキスト列 **htmlfile** にロードした **HTML** ファイル・セットを索引付けする場合は、データストア、フィルタおよびセクション・グループの各パラメータを次のように指定して、**CREATE INDEX** 文を発行できます。

```
CREATE INDEX myindex ON doc(htmlfile) INDEXTYPE IS ctxsys.context PARAMETERS
('datastore ctxsys.default_datastore filter ctxsys.null_filter section group
ctxsys.html_section_group');
```

関連項目：

- 様々な索引の作成方法は、[第 2 章「索引付け」の「索引付け時の考慮事項」](#)を参照してください。
- **CREATE INDEX** 文の詳細は、『Oracle Text リファレンス』を参照してください。

CTXCAT 索引を作成する場合

CTXCAT 索引は、複合問合せ用に最適化された索引です。関連する構造化情報を持った小さなドキュメントまたはテキスト断片を格納する場合は、このタイプの索引を作成できます。この索引を問い合わせるには、**CATSEARCH** 演算子を使用して構造化句（ある場合）を指定し

ます。構造化問合せでは通常、CONTEXT 索引より CTXCAT 索引を使用したほうが、問合せパフォーマンスが向上します。複合問合せのパフォーマンスを向上させるには、CTXCAT 索引を正しく構成する必要があります。

関連項目： 詳細な例は、[第 2 章「索引付け」](#)の「[CTXCAT 索引の作成](#)」を参照してください。

CTXRULE 索引を作成する場合

CTXRULE 索引は、ドキュメントの着信ストリームをドキュメントの内容に基づいて分類するアプリケーションを作成する場合に作成します。分類ルールは、条件式として定義します。単一ドキュメントの分類には、MATCHES 演算子を使用します。

関連項目： 詳細な例は、[第 2 章「索引付け」](#)の「[CTXRULE 索引の作成](#)」を参照してください。

索引のメンテナンス

索引のメンテナンスは、アプリケーションが元表のドキュメントを挿入、更新または削除した後に必要です。索引のメンテナンスには、索引の同期化と最適化が含まれます。

元表が静的な場合（アプリケーションが最初の索引付け以降にドキュメントを更新、挿入または削除していない場合）は、索引を同期化する必要はありません。

ただし、アプリケーションが元表に対して DML 操作（挿入、更新または削除）を実行した場合は、索引を同期化する必要があります。索引は、PL/SQL プロシージャ CTX_DDL.SYNC_INDEX を使用して手動で同期化できます。

次の例では、2MB のメモリーを使用して索引 myindex を同期化します。

```
begin
    ctx_ddl.sync_index('myindex', '2M');
end;
```

索引を定期的に同期化する場合は、断片化を削減し古いデータを削除するために、索引の最適化を考慮することもできます。

関連項目： 索引の同期化および最適化の詳細は、[第 2 章「索引付け」](#)の「[CONTEXT 索引に関する DML 操作の管理](#)」を参照してください。

単純なテキスト問合せアプリケーション

典型的な問合せアプリケーションでは、ユーザーが問合せを入力できるようになっています。アプリケーションは問合せを実行し、その問合せを満たすドキュメントのリスト（ヒットリスト）を戻します。ヒットリストは通常、関連性の度合いによってランク付けされています。アプリケーションでは、戻されたヒットリスト内の 1 つ以上のドキュメントを表示できます。

たとえば、アプリケーションで World Wide Web 上の URL（HTML ファイル）を索引付けすることにより、索引付けされた一連の URL 内での問合せ機能が提供されます。この場合、問合せアプリケーションにより戻されるヒットリストは、ユーザーがアクセスできる URL で構成されます。

図 1-3 典型的な問合せアプリケーションのフローチャート

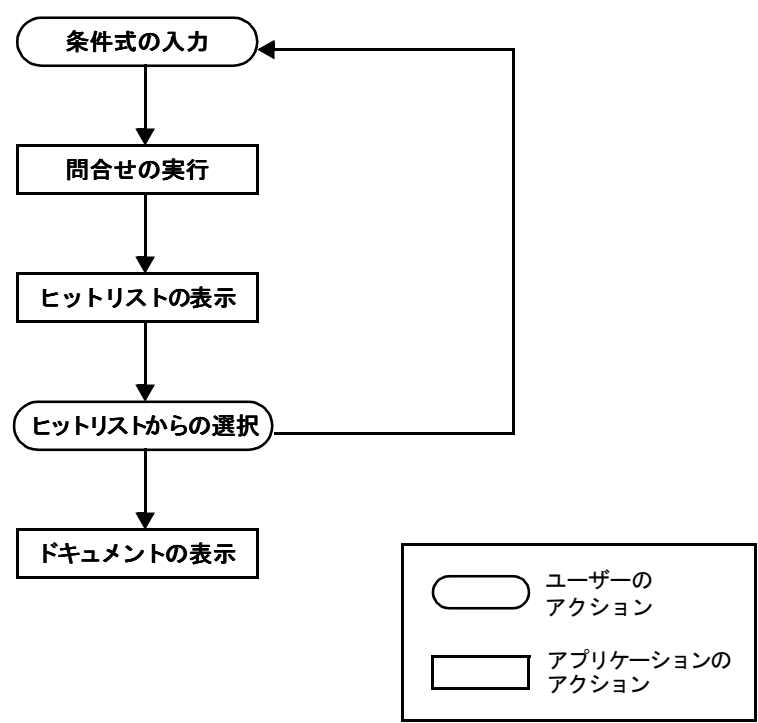


図 1-3 は、単純な問合せアプリケーションとユーザーとの対話方法をフローチャートで示しています。この図は、問合せの入力と結果の表示に必要な手順を示しています。楕円形のボックスはユーザーのタスクを、四角形のボックスはアプリケーションのタスクを示しています。

前述のとおり、問合せアプリケーションは、次の手順に従ってモデル化できます。

1. ユーザーが条件式を入力します。
2. アプリケーションが問合せを実行します。
3. アプリケーションがヒットリストを表示します。
4. ユーザーがヒットリストからドキュメントを選択します。
5. アプリケーションがドキュメントを表示します。

この章の後半では、Oracle Text でこの手順を実行する方法を説明します。

関連項目： 単純な Web 問合せアプリケーションの詳細は、[付録 A](#)「CONTEXT 問合せアプリケーション」を参照してください。

索引の問合せ方法の理解

Oracle Text では、CONTAINS 演算子を使用して CONTEXT 索引を問い合わせます。これは、問合せアプリケーションの作成に使用する最も一般的な演算子と索引です。

さらに高度なアプリケーションの場合は、CTXCAT 索引の問合せには CATSEARCH 演算子を、CTXRULE 索引の問合せには MATCHES 演算子をそれぞれ使用します。

CONTAINS を使用した問合せ方法

CONTAINS 演算子を使用すると、ワードまたは句を含むドキュメントを検索できます。CONTAINS 問合せを発行する前に、ドキュメントを索引付けしておく必要があります。

CONTAINS 演算子は、SELECT 文の中で使用します。CONTAINS を使用すると、次の 2 種類の問合せを発行できます。

- ワード問合せ
- ABOUT 問合せ

また、問合せを最適化すると、上位から指定した数（トップ *n*）のヒットを取得するようにして、応答時間を短縮できます。次の各項では、これらの問合せの使用例について、その概要を説明します。

ワード問合せ

ワード問合せでは、CONTAINS または CATSEARCH 演算子に一重引用符で囲んで入力したワードまたは句の完全一致を問い合わせます。

次の例では、ワード *oracle* を含む *text* 列のすべてのドキュメントを検索します。各行のスコアは、SCORE 演算子でラベル 1 を使用して選択されます。

```
SELECT SCORE(1), title FROM news WHERE CONTAINS(text, 'oracle', 1) > 0;
```


問合せ式では、AND や OR などのテキスト演算子を使用して多様な結果を得ることができます。また、構造化述語を WHERE 句に追加することもできます。

関連項目： 問合せに使用できる様々な演算子の詳細は、『Oracle Text リファレンス』を参照してください。

SQL の COUNT(*) 文または PL/SQL プロシージャ CTX_QUERY.COUNT_HITS を使用して、問合せに対するヒット数をカウントできます。

ABOUT 問合せ

ABOUT 問合せは、CONTAINS 句に ABOUT 演算子を使用して発行します。

ABOUT 問合せを使用すると、すべての言語で、問合せから戻される関連ドキュメント数が増加します。

英語とフランス語の ABOUT 問合せでは、デフォルトで作成された索引のテーマ・コンポーネントを使用できます。したがって、この演算子は指定したワードまたは句の完全一致のみでなく、問合せの概念に基づいてドキュメントを検出します。

たとえば、次の問合せでは、単に *politics* というワードを含むドキュメントのみでなく、*text* 列にある、*politics* という主題に関するすべてのドキュメントを検索します。

```
SELECT SCORE(1), title FROM news WHERE CONTAINS(text, 'about(politics)', 1) > 0;
```

関連項目： ABOUT 演算子の詳細は、『Oracle Text リファレンス』を参照してください。

応答時間短縮のための問合せの最適化

すべての CONTAINS 問合せ（ワードまたは ABOUT 問合せ）は、結果セット内で最もランクが高いヒットを最短時間で取り出せるように、応答時間を短縮するための最適化を行うことができます。応答時間を短縮するための最適化は、Web ベースの検索アプリケーションで有効です。

関連項目： 第 5 章「パフォーマンス・チューニング」の「応答時間短縮のための問合せの最適化」

構造化フィールドの検索

使用しているアプリケーション・インタフェースから、検索条件を追加する手段として、項目説明、作成者、日付などテキストに関連する構造化フィールドに対する問合せを選択できます。

SELECT 文で構造化句を使用して、CONTAINS 演算子による構造化検索を発行できます。ただし、パフォーマンスを最適化するには、CTXCAT 索引を作成することを検討してください。CATSEARCH 演算子を使用したほうが、構造化問合せのパフォーマンスが向上します。

また、使用しているアプリケーションで、ヒットリスト内の各ドキュメントに関連する構造化情報を表示できます。

関連項目： CATSEARCH による構造化問合せの改善に必要な CTXCAT 索引の作成方法は、[第 2 章「索引付け」](#)の「[CTXCAT 索引の作成](#)」を参照してください。

シソーラスを使用した問合せ

Oracle Text では、問合せアプリケーション用のシソーラスを定義できます。

カスタム・シソーラスを定義すると、問合せをより優れた方法で処理できます。使用しているアプリケーションのユーザーには、トピックを表現するワードがわからない場合があるため、予想される問合せ語句にシノニムまたは下位語を定義できます。シソーラス演算子を使用すると、シソーラス語句を含むように問合せを拡張できます。

関連項目： [第 7 章「シソーラスの使用」](#)

ドキュメントのセクション検索

セクション検索を使用すると、テキスト問合せをドキュメント内のセクションに絞り込むことができます。

セクション検索は、HTML や XML のドキュメントのように、ドキュメントに内部構造がある場合に実現できます。たとえば、<H1> タグに対してセクションを定義すると、WITHIN 演算子を使用してこのセクション内を問い合わせることができます。

XML ドキュメントからセクションを自動的に作成するようにシステムを設定できます。

また、属性セクションを定義して、XML ドキュメントの属性テキストを検索できます。

注意： セクション検索がサポートされているのは、CONTEXT 索引によるワード問合せのみです。

関連項目： [第 6 章「ドキュメントのセクション検索」](#)

その他の問合せ機能

問合せアプリケーションでは、近接検索などのその他の問合せ機能を使用できます。表 1-1 は、これらの問合せ機能のうちいくつかを示しています。

表 1-1 Oracle Text の問合せ機能

機能	説明	実施方法
大 / 小文字を区別する検索	大 / 小文字を区別する検索です。	索引作成時に BASIC_LEXER を使用します。
基本文字変換	ティルデ、アクセント、ウムラウトなどの発音区別符号に関係なく、ワードを問い合わせます。たとえば、スペイン語の基本文字索引を使用すると、 <i>energía</i> の問合せでは、 <i>energía</i> および <i>energia</i> が含まれているドキュメントが一致します。	索引作成時に BASIC_LEXER を使用します。
ワード分割処理 (ドイツ語およびオランダ語)	指定した語句が複合語の要素として含まれているワードを検索できます。	索引作成時に BASIC_LEXER を使用します。
代替スペル (ドイツ語、オランダ語およびスウェーデン語)	ワードの代替スペルを検索します。	索引作成時に BASIC_LEXER を使用します。
近接検索	相互に近接しているワードを検索します。	問合せ発行時に NEAR 演算子を使用します。
ステミング	指定した語句と同じ語幹を持つワードを検索します。	問合せ発行時に \$ 演算子を使用します。
ファジー検索	指定した語句に類似するスペルを持つワードを検索します。	問合せ発行時に FUZZY 演算子を使用します。
問合せ実行計画	問合せの解析情報を生成します。	索引作成後に PL/SQL プロシージャ CTX_QUERY.EXPLAIN を使用します。
階層問合せフィードバック	問合せに対する上位語、下位語および関連語の情報を生成します。	索引作成後に PL/SQL プロシージャ CTX_QUERY.HFEEDBACK を使用します。
索引のブラウズ	索引内のシード・ワードに関するワードをブラウズします。	索引作成後に PL/SQL プロシージャ CTX_QUERY.BROWSE_WORDS を使用します。

表 1-1 Oracle Text の問合せ機能（続き）

機能	説明	実施方法
ヒット数のカウント	問合せのヒット数をカウントします。	索引作成後に PL/SQL プロシージャ <code>CTX_QUERY.COUNT_HITS</code> を使用します。
ストアド・クエリー式	クエリー式を格納します。	索引作成後に PL/SQL プロシージャ <code>CTX_QUERY.STORE_SQE</code> を使用します。
シソーラスを使用した問合せ	シソーラスを使用して問合せを拡張します。	<code>SYN</code> および <code>BT</code> などのシソーラス演算子と <code>ABOUT</code> 演算子を使用します。 シソーラスのメンテナンスには <code>CTX_THES</code> パッケージを使用します。

ヒットリストの表示

問合せの実行後、問合せアプリケーションでは通常、問合せを満たすすべてのドキュメントのヒットリストが関連性スコアとともに表示されます。このリストは、ドキュメント・セットに応じて、ドキュメント・タイトルまたは URL のリストになります。


アプリケーションでは、次の 1 つ以上の方法でヒットリストを表示します。

- ドキュメントをスコア順に表示します。
- タイトルや作成者など、ドキュメントに関連する構造化フィールドを表示します。
- ドキュメントのヒット件数を表示します。

ヒットリストの例

図 1-4 は、ヒットリストを表示する問合せアプリケーションの画面です。















図 1-4 ヒットリストを表示する問合せアプリケーション



Search for

Page 1 of 4 Oracle found 34 results for Java and XML

[\[Next\]](#)

- [1 Oracle Corp. Document](#)  
Gist: Oracle has built new functionality on the object support of Oracle8 to produce O ...
Precision score: 100%, **Last modified:** 13-MAR-00, **Page size:** 56731 bytes
Keywords: Oracle8i, paradigms, Internet, CYBERSMARTS, delivery, technology, systems, Oracl ...
- [2 White Paper](#)  
Gist: Supporting standard EJB and CORBA deployment architectures, Oracle Business Comp ...
Precision score: 67%, **Last modified:** 13-MAR-00, **Page size:** 29053 bytes
Keywords: Oracle Corporation, Java, Oracle Business Components, logic, relation, writing, ...
- [3 Oracle JDeveloper Suite - Technical Documentation - Partners](#)  
Gist: Partner Software Solutions Halcyon Software produces a suite of products for c ...
Precision score: 43%, **Last modified:** 13-MAR-00, **Page size:** 6464 bytes
Keywords: partners, Oracle JDeveloper, computer software, Java, Oracle Corporation, conver ...
- [4 Oracle JDeveloper - Technical Resources - Documentation](#)  
Gist: Using the Oracle Business Components framework, the developer can focus on writi ...
Precision score: 19%, **Last modified:** 13-MAR-00, **Page size:** 7254 bytes
Keywords: Java, Oracle Corporation, Oracle JDeveloper, technology, logic, Oracle Business ...
- [5 Oracle Corp. Document](#)  
Gist: Oracle8i Adds InterMedia, iFS, and XML Support to Provide Leading Platform for I ...
Precision score: 18%, **Last modified:** 02-MAR-00, **Page size:** 10050 bytes
Keywords: Oracle Corporation, management, contents, Oracle8i, Internet, technology, Oracle ...
- [6 Oracle Corp. Document](#)  
Gist: "Over the last decade corporations have built meta-data islands. IT departments ...
Precision score: 18%, **Last modified:** 02-MAR-00, **Page size:** 4711 bytes
Keywords: Oracle Corporation, Oracle Repository, management, COM5, construction, Meta, fun ...
- [7 Oracle Corp. Document](#)  
Gist: "Over the last decade corporations have built meta-data islands. IT departments ...
Precision score: 18%, **Last modified:** 02-MAR-00, **Page size:** 4711 bytes
Keywords: Oracle Corporation, Oracle Repository, management, COM5, construction, Meta, fun ...

構造化フィールドの表示

テキスト列に関連する構造化列は、ドキュメントの識別に役立ちます。ヒットリストを表示するときに、ドキュメントを識別するためにドキュメント・タイトル、作成者、あるいはその他のフィールドの組合せを表示できます。

ヒットリストに含める構造化列の名前は、`SELECT` 文で指定します。

ヒットリストの順序付け

テキスト問合せまたはテーマ問合せのいずれかを発行すると、その問合せを満たすドキュメントのヒットリストが、各ドキュメントの関連性スコアとともに戻されます。このスコアを使用して、最も関連性の高いドキュメントを最初に表示するように、ヒットリストを順序付けることができます。

各ドキュメントのスコアは 1 から 100 までです。スコアが高いほど、問合せに対するドキュメントの関連性が高くなります。

Oracle では、`CONTAINS` 演算子と `CATSEARCH` 演算子の使用時に、スコアが計算されます。スコアは、`SCORE` 演算子を使用して取得します。

関連項目： [第 3 章「問合せ」](#)

ドキュメントのヒット件数の表示

`SELECT COUNT(*)` を使用して、問合せがヒットリストとともに戻すヒット数を表示します。たとえば、次の式を使用します。

```
SELECT COUNT(*) FROM docs WHERE CONTAINS(text, 'oracle', 1) > 0;
```

PL/SQL でヒット数をカウントする場合は、`CTX_QUERY.COUNT_HITS` プロシージャを使用できます。

ドキュメントの表示方法およびハイライト表示

通常、問合せアプリケーションでは、問合せから戻されるドキュメントを表示できます。ユーザーがヒットリストからドキュメントを選択すると、アプリケーションはドキュメントを特定の形式で表示します。

Oracle Text では、様々な方法でドキュメントを表示できます。たとえば、問合せ語句をハイライト表示させてドキュメントを表示できます。ハイライト表示できる問合せ語句は、ワード問合せのワードまたは英語での ABOUT 問合せのテーマのいずれかです。

また、PL/SQL パッケージ CTX_DOC を使用して、ドキュメントから要旨（ドキュメントのサマリー）とテーマ情報を取得することもできます。

表 1-2 は、取得可能な出力と、各出力を取得するためのプロシージャを示しています。

表 1-2 CTX_DOC の出力

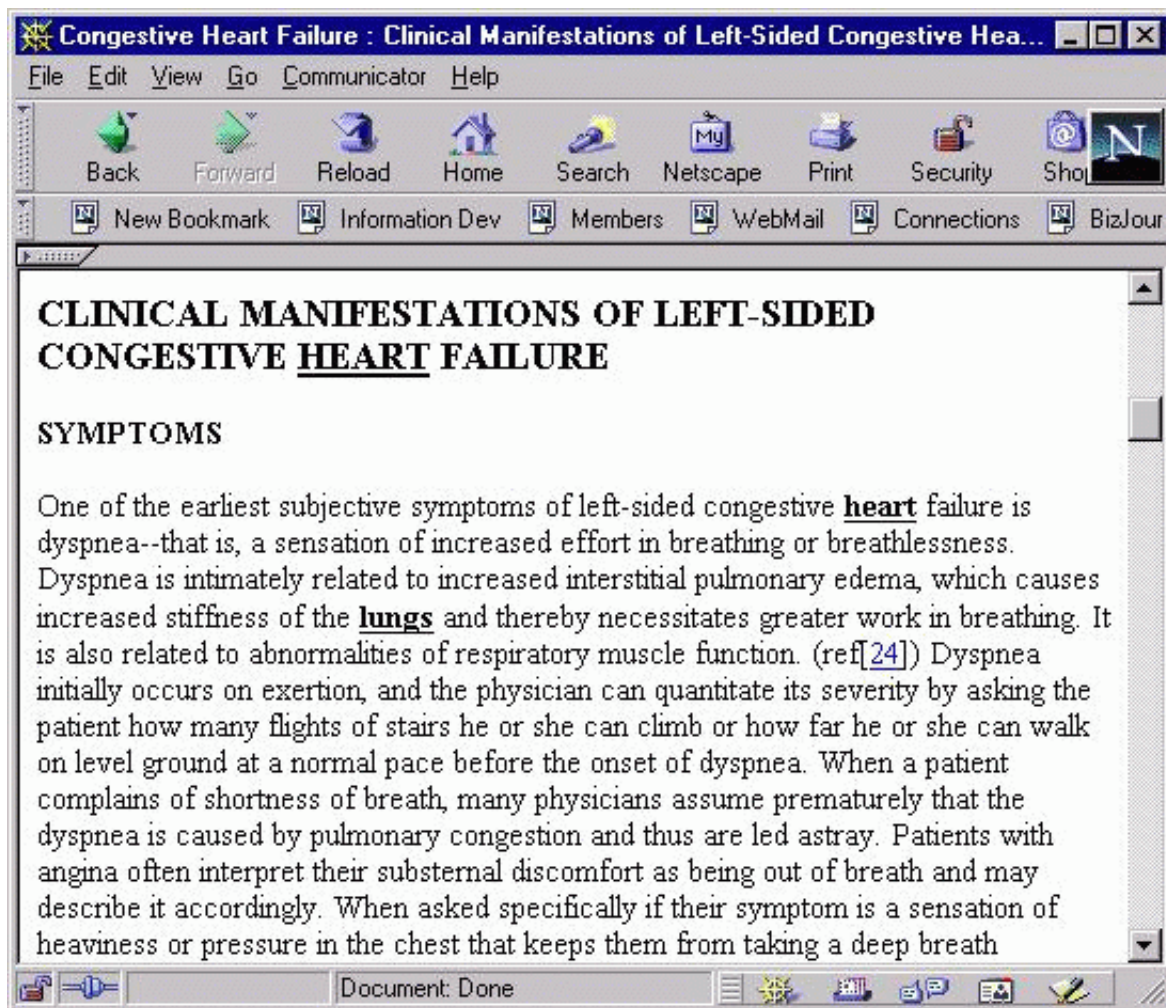
出力	プロシージャ
プレーン・テキスト形式、ハイライト表示なし	CTX_DOC.FILTER
HTML 形式のドキュメント、ハイライト表示なし	CTX_DOC.FILTER
ハイライト表示されたドキュメント、プレーン・テキスト形式	CTX_DOC.MARKUP
ハイライト表示されたドキュメント、HTML 形式	CTX_DOC.MARKUP
プレーン・テキスト形式のハイライト・オフセット情報	CTX_DOC.HIGHLIGHT
HTML 形式のハイライト・オフセット情報	CTX_DOC.HIGHLIGHT
ドキュメントのテーマ・サマリーと要旨	CTX_DOC.GIST
ドキュメントのテーマ・リスト	CTX_DOC.THEMES

関連項目： [第 4 章「ドキュメントの表示方法」](#)

ハイライト表示の例

図 1-5 は、問合せ語句 *heart* および *lungs* がハイライト表示されているドキュメントを表示している問合せアプリケーションの画面です。

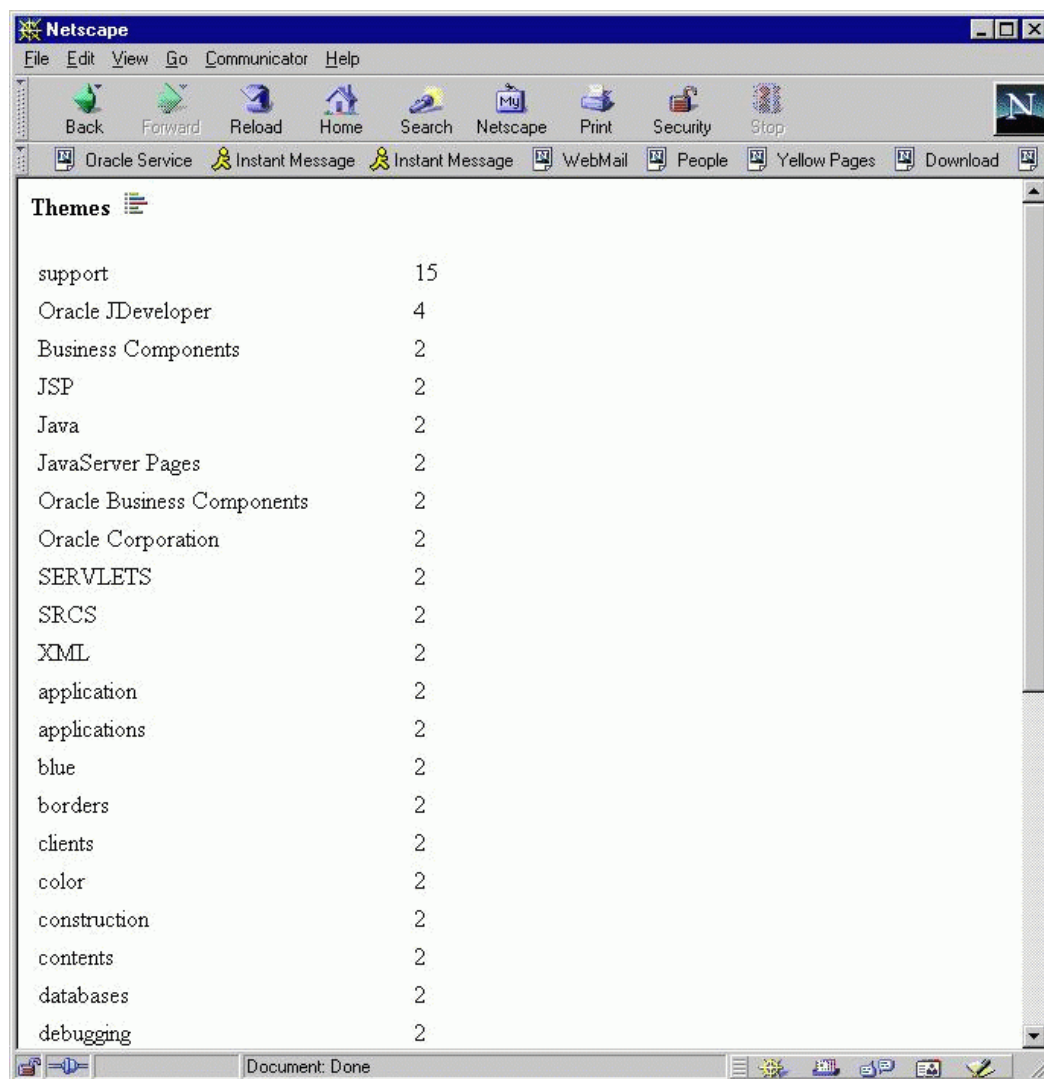
図 1-5 ハイライト表示されたドキュメントを表示する問合せアプリケーション



ドキュメントのテーマ・リストの例

図 1-6 は、ドキュメントのテーマ・リストを表示している問合せアプリケーションの画面です。

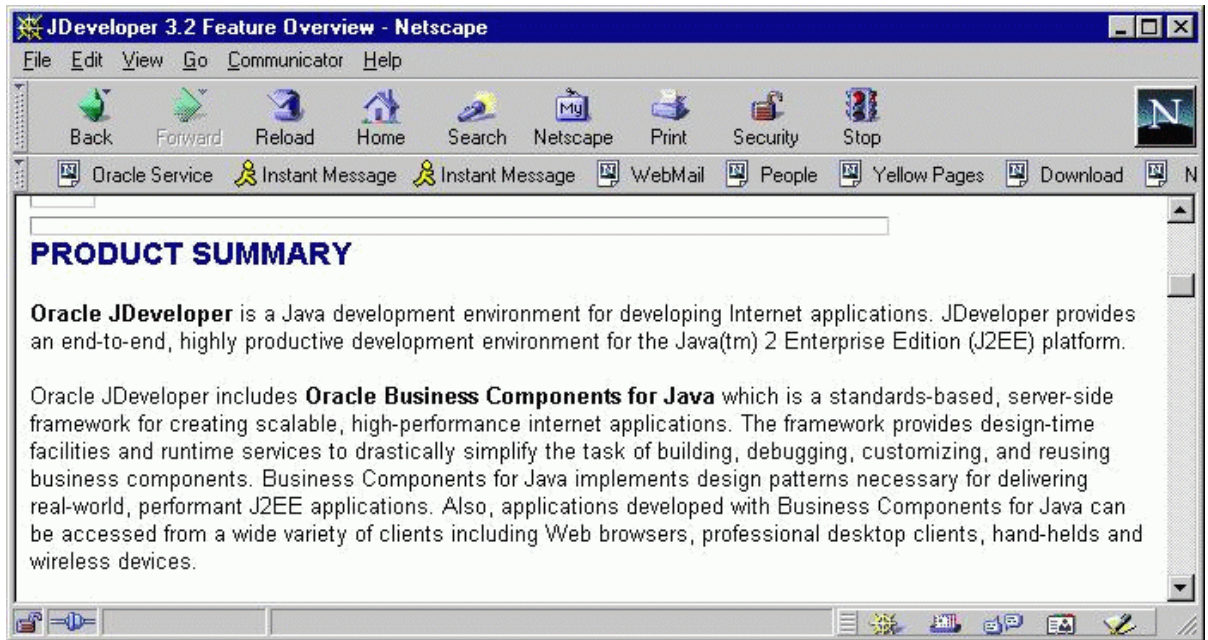
図 1-6 ドキュメントのテーマを表示する問合せアプリケーション



要旨の例

図 1-7 は、ドキュメントの要旨を表示している問合せアプリケーションの画面です。

図 1-7 ドキュメントの要旨を表示する問合せアプリケーション



索引付け

この章では、Oracle Text の索引付け機能について説明します。次の項目について説明します。

- Oracle Text の索引の概要
- 索引付け時の考慮事項
- 索引の作成
- 索引のメンテナンス
- CONTEXT 索引に関する DML 操作の管理

Oracle Text の索引の概要

Oracle Text の索引は、Oracle ドメイン索引です。問合せアプリケーションを作成するには、CONTEXT タイプの索引を作成し、CONTAINS 演算子を使用してその索引を問い合わせます。

索引は、移入済みのテキスト表から作成します。問合せアプリケーションでは、テキスト表にテキストまたはテキストの格納場所へのポインタが含まれている必要があります。テキストは通常ドキュメントの集まりですが、小さいテキスト断片の場合もあります。

複合問合せのパフォーマンスを改善する場合は、CTXCAT 索引を作成できます。使用しているアプリケーションが、日付や価格などの関連基準に基づいて、小さいドキュメントや記述的なテキスト断片を検索する複合問合せを利用する回数が多い場合は、この索引タイプを使用します。この索引は、CATSEARCH 演算子で問い合わせます。

ドキュメント分類アプリケーションの作成には、CTXRULE タイプの索引を作成します。この索引タイプでは、MATCHES 演算子を使用して、プレーン・テキスト、HTML または XML の各ドキュメントを分類できます。定義する問合せセットは、索引付け対象のテキスト表に格納します。

XMLtype 列を使用している場合は、CTXXPATH 索引により existsNode 問合せを高速に実行できます。

テキスト索引は、標準 SQL を使用して、Oracle の拡張索引の 1 タイプとして作成します。つまり、Oracle Text の索引は、Oracle 索引と同じように機能します。この索引には参照名があり、標準 SQL 文を使用して操作できます。

Oracle Text の索引を作成するメリットは、CONTAINS、CATSEARCH および MATCHES などの Oracle Text の演算子を使用したテキスト問合せの応答時間が短縮されることです。これらの演算子は、それぞれ CONTEXT、CTXCAT および CTXRULE の各索引タイプを問い合わせます。

関連項目： この章の「[索引の作成](#)」を参照してください。

CTXXPATH 索引タイプの詳細は、『Oracle9i XML データベース開発者ガイド - Oracle XML DB』を参照してください。

Oracle Text の CONTEXT 索引の構造

Oracle Text では、すべてのワードをトークンに変換してテキストを索引付けします。Oracle Text の CONTEXT 索引の一般的な構造は、逆向きの索引です。つまり、各トークンにそのトークンを含むドキュメント（行）のリストが格納されています。

たとえば、初期の索引付け操作が 1 つ終了すると、ワード DOG のエントリは次のようになります。

```
DOG DOC1 DOC3 DOC5
```

つまり、ワード DOG は、ドキュメント 1、3 および 5 を格納する行に含まれています。

詳細は、この章の「索引の最適化」を参照してください。

マージ済みのワードとテーマ索引

Oracle Text では、英語およびフランス語の場合はデフォルトで、ワード情報とともにテーマ情報も索引付けします。テーマ情報は、ABOUT 演算子を使用して問い合わせることができます。必要に応じて、テーマの索引付けを使用可能および使用禁止に設定できます。

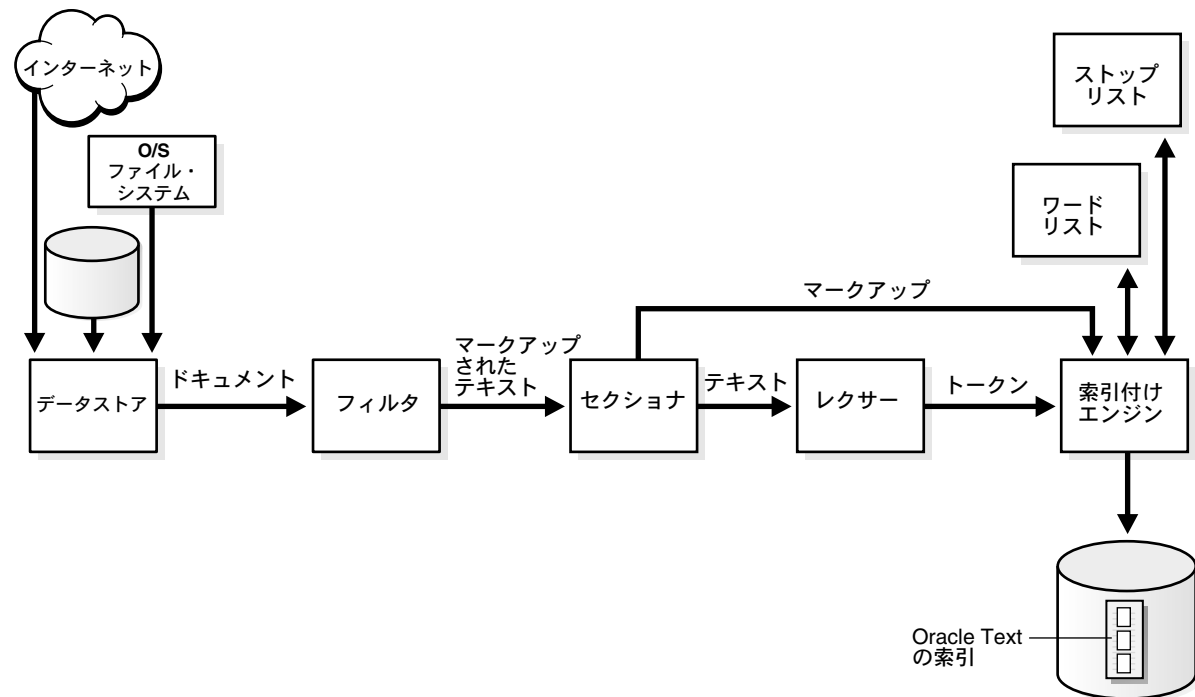
関連項目： テーマ情報の索引付けの詳細は、この章の「[プリファレンスの作成](#)」を参照してください。

Oracle Text の索引付け処理

この項では、Oracle Text の索引付け処理について説明します。索引付け処理は、CREATE INDEX 文を使用して開始します。この処理の目的は、指定したパラメータとプリファレンスに基づいて、トークンに対して Oracle Text の索引を作成することです。

[図 2-1](#) は、索引付け処理を示しています。データは、様々な索引付けオブジェクトによって操作されます。各オブジェクトは、CREATE INDEX または ALTER INDEX のパラメータ文字列で指定した索引付けプリファレンス型またはセクション・グループにそれぞれ対応しています。これらのオブジェクトについては、次の項で説明します。

図 2-1 Oracle Text の索引付け処理



データストア・オブジェクト

ストリームはデータストアから始まります。データストアでは、指定したデータストア・プリファレンスに従って、システム内に格納されているとおりにドキュメントを読み込みます。たとえば、データストアを `FILE_DATASTORE` と定義した場合、ストリームは、オペレーティング・システムからファイルを読み込むことによって開始します。ドキュメントは、インターネット上または Oracle データベースに格納することもできます。

フィルタ・オブジェクト

次に、ストリームはフィルタ内を通過します。フィルタで行われる処理の内容は、指定した `FILTER` プリファレンスによって決定されます。次のいずれかの処理がストリームに対して行われます。

- フィルタ処理は行われません。`NULL_FILTER` プリファレンス型を指定した場合に発生します。プレーン・テキスト、HTML または XML のドキュメントには、フィルタ処理は不要です。
- 形式設定されたドキュメント（バイナリ）は、マークアップされたテキストにフィルタ処理されます。`INSO_FILTER` プリファレンス型を指定した場合に発生します。

- テキストは、非データベース・キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。CHARSET_FILTER プリファレンス型を指定した場合に発生します。

セクショナ・オブジェクト

フィルタ処理後、マークアップされたテキストはセクショナを通過し、ストリームはテキスト情報とセクション情報に分割されます。セクション情報には、テキスト・ストリーム内でセクションが開始する場所と終了する場所の情報が含まれています。抽出されるセクションのタイプは、セクション・グループのタイプによって決定されます。

セクション情報は、直接索引付けエンジンに渡され、後で使用されます。テキストはレクサーに渡されます。

レクサー・オブジェクト

レクサーは、使用言語に基づいて、テキストをトークンに分解します。通常、分解されたトークンはワードです。レクサーは、トークンの抽出に、レクサー・プリファレンスに定義されたパラメータを使用します。これらのパラメータには、トークンの分割に使用する空白などの文字の定義や、テキストをすべて大文字に変換するか、あるいは大 / 小文字混合のままで残すかなどの定義が含まれています。

テーマの索引付けが使用可能な場合、レクサーはテキストを分析してテーマ・トークンを作成し、索引付けを行います。

索引付けエンジン

索引付けエンジンは、トークンを含んでいるドキュメントにそのトークンをマップする逆向きの索引を作成します。このフェーズでは、指定したストップリストを使用して索引からストップワードまたはストップテーマを排除します。また、WORDLIST プリファレンスに定義されたパラメータも使用します。このパラメータによって、システムはプリフィックス索引またはサブストリング索引（使用可能な場合）の作成方法を判断します。

パーティション表とパーティション索引

パーティション・テキスト表に CONTEXT パーティション索引を作成できます。パーティション表は、レンジでパーティション化する必要があります。ハッシュ・パーティション、コンポジット・パーティションおよびリスト・パーティションは、サポートされていません。

日付でデータをパーティション化したパーティション・テキスト表を作成できます。たとえば、使用しているアプリケーションで日付付きのニュース記事の大規模なライブラリをメンテナンスしている場合は、月別または年度別に情報をパーティション化できます。パーティション化によって、問合せ、DML およびバックアップとリカバリの操作を単一のパーティションに対して行うことができるため、大規模データベースが管理しやすくなります。

関連項目： パーティション化の詳細は、『Oracle9i データベース概要』を参照してください。

パーティション表の問合せ

パーティション表の問合せには、通常の表の問合せと同様に、SELECT 文で CONTAINS を使用します。表全体または単一のパーティションを問い合わせることができます。ただし、ORDER BY SCORE 句を使用する場合は、問合せを単一のパーティションに制限する範囲述語を組み込まないかぎり、単一のパーティションを問い合わせることをお勧めします。

オンラインでの索引の作成

継続的に更新が発生し、索引付けのために元表をロックすることが実用的でない場合は、CREATE INDEX の ONLINE パラメータを使用して、オンラインで索引を作成できます。この方法を使用すると、DML 操作が頻繁に発生するアプリケーションで、索引のために元表の更新を一時停止する必要がありません。

ただし、索引付け処理の最初と最後に元表が短時間ロックされます。

関連項目： オンラインでの索引の作成の詳細は、『Oracle Text リファレンス』を参照してください。

パラレル索引付け

Oracle Text では、CREATE INDEX を使用したパラレル索引付けがサポートされています。

非パーティション表にパラレル索引付けコマンドを発行すると、元表がパーティションに分割され、スレーブ・プロセスが起動され、各スレーブにそれぞれ別のパーティションが割り当てられます。各スレーブが、それぞれのパーティションの行を索引付けします。元表をパーティションに分割する方法は Oracle によって判断され、ユーザーが直接制御することはありません。また、実際に起動されるスレーブ・プロセス数も Oracle によって判断され、マシンの性能、システム負荷、init.ora 設定およびその他の要因に依存します。実際の並列度は、要求された並列度に一致しない可能性があります。

索引付けは I/O 集中型の操作であるため、分散ディスク・アクセスと複数 CPU を使用する場合は、パラレル索引付けが索引付けに要する時間を短縮するための最も効果的な操作となります。パラレル索引付けは、CREATE INDEX を使用した初期索引のパフォーマンスにのみ影響します。ALTER INDEX を使用した DML のパフォーマンスには影響しません。また、問合せパフォーマンスにもほとんど影響しません。

パラレル索引付けによって、初期の索引付け時間が短縮されるため、次の操作に有効です。

- データ・ステージング。使用している製品に Oracle Text の索引が組み込まれている場合に有効です。
- アプリケーションの高速な初期起動。大規模なデータ・コレクションを対象にする場合に有効です。
- アプリケーションのテスト。アプリケーションの開発時に多様な索引パラメータやスキーマのテストが必要な場合に有効です。

関連項目：

パラレル索引の作成方法の詳細は、[第 5 章「パフォーマンス・チューニング」](#)の「[索引付けのパフォーマンスに関する FAQ（よくある質問）](#)」を参照してください。

『Oracle Text リファレンス』

索引付けの制限事項

複数の索引を持つ列

1つの列に連結できるのは、1つのドメイン索引のみです。これは、Oracle の標準の動作と一致しています。ただし、単一のテキスト索引には、ワード情報の他にテーマ情報を格納できます。

ビューの索引付け

Oracle SQL の標準では、ビューに対する索引の作成はサポートされていません。したがって、複数の表にコンテンツを含むドキュメントの索引付けが必要な場合は、USER_DATASTORE オブジェクトを使用してデータ記憶域プリファレンスを作成できます。このオブジェクトを使用して、索引時に複数の表のドキュメントを合成するプロシージャを定義できます。

関連項目： USER_DATASTORE の詳細は、『Oracle Text リファレンス』を参照してください。

索引付け時の考慮事項

Oracle Text の索引を作成するには、CREATE INDEX 文を使用します。索引の作成時にパラメータ文字列を指定しないと、索引は、デフォルトのパラメータを使用して作成されます。

また、デフォルトを変更し、問合せアプリケーションにあわせて索引をカスタマイズできます。CREATE INDEX で索引をカスタマイズするために使用するパラメータとプリファレンスのタイプは、次の一般的なカテゴリに分類されます。

索引のタイプ

Oracle Text では、CREATE INDEX を使用して次の 4 つの索引タイプのいずれかを作成できます。次の表は、各タイプ、その用途およびサポートしている機能を示しています。

索引タイプ	説明	サポートされるプリファレンスとパラメータ	問合せ演算子	注意
CONTEXT	<p>テキストが大量のまとまったドキュメントで構成されている場合、テキスト検索アプリケーションを作成するには、この索引を使用します。MS Word、HTML またはプレーン・テキストなどの多様な形式のドキュメントを索引付けできます。</p> <p>CONTEXT 索引では、様々な方法で索引をカスタマイズできます。</p> <p>この索引タイプでは、元表に対する DML の後に CTX_DDL.SYNC_INDEX が必要です。</p>	<p>INDEX SET を除くすべての CREATE INDEX プリファレンスとパラメータがサポートされています。</p> <p>このサポート対象のパラメータには、索引パーティション句、形式列、キャラクタ・セット列および言語列が含まれます。</p>	<p>CONTAINS</p> <p>文法は CONTEXT 文法と呼ばれ、豊富な操作をサポートします。</p> <p>CTXCAT 文法は、問合せテンプレートとともに使用できます。</p>	<p>すべてのドキュメント・サービスと問合せサービスがサポートされています。</p> <p>パーティション・テキスト表の索引付けがサポートされています。</p>
CTXCAT	<p>複合問合せのパフォーマンスを改善するには、この索引タイプを使用します。通常は、この索引タイプを使用して、小さいドキュメントやテキスト断片を索引付けします。複合問合せのパフォーマンスを改善するために、項目名、価格および説明などの元表の他の列を索引に組み込むことができます。</p> <p>この索引タイプはトランザクションに基づき、元表に対する DML の後に自動的に更新されます。CTX_DDL.SYNC_INDEX は不要です。</p>	<p>INDEX SET</p> <p>LEXER (テーマの索引付けはサポートされていません。)</p> <p>STOPLIST</p> <p>STORAGE</p> <p>WORDLIST (日本語データの prefix_index 属性のみがサポートされています。)</p> <p>形式列、キャラクタ・セット列および言語列はサポートされていません。</p> <p>表のパーティション化および索引のパーティション化はサポートされていません。</p>	<p>CATSEARCH</p> <p>文法は CTXCAT と呼ばれ、論理操作、句問合せおよびワイルド・カード操作をサポートします。</p> <p>CONTEXT 文法は、問合せテンプレートとともに使用できます。</p>	<p>CTXCAT 索引のサイズは、索引付け対象のテキストの合計量、索引セット内の索引数および索引付けされた列の数に関連します。索引セットに索引を追加する前に、使用する問合せとリソースを慎重に考慮してください。</p> <p>CTXCAT 索引は、表および索引のパーティション化、ドキュメント・サービス (ハイライト表示、マークアップ、テーマおよび要旨) または問合せサービス (実行計画、問合せフィードバックおよびワードのブラウズ) をサポートしていません。</p>

索引タイプ	説明	サポートされるプリファレンスとパラメータ	問合せ演算子	注意
CTXRULE	ドキュメント分類アプリケーションまたはルーティング・アプリケーションの作成には、CTXRULE 索引を使用します。CTXRULE 索引は、問合せ表に作成する索引です。問合せには、分類基準とルーティング基準を定義します。	<p>問合せセットの索引付けにサポートされているのは、BASIC_LEXER 型のみです。</p> <p>問合せセットの問合せには、ABOUT、STEM、AND、NEAR、NOT および OR の各演算子を組み込むことができます。</p> <p>次の演算子はサポートされていません。ACCUM、EQUIV、WITHIN、WILDCARD、FUZZY、SOUNDEX、MINUS、WEIGHT、THRESHOLD。</p> <p>CREATE INDEX の STORAGE 句は、問合せに索引を作成する場合にサポートされています。</p> <p>セクション・グループは、MATCHES 演算子を使用してドキュメントを分類する場合にサポートされています。</p> <p>ワードリストは、問合せセットに対するステミング操作でサポートされています。</p> <p>filter、memory、datastore および populate の各パラメータは、CTXRULE 索引タイプには使用できません。</p>	MATCHES	単一のドキュメント（プレーン・テキスト、HTML または XML）は、MATCHES 演算子を使用して分類できます。この演算子は、ドキュメントを問合せのセットに変更し、CTXRULE 索引内の一致する行を検索します。
CTXXPATH	XMLType 列に対する existsNode() 問合せを高速にする必要がある場合に、この索引を作成します。	STORAGE	existsNode() とともに使用します。	<p>この索引を作成できるのは、XMLType 列に対してのみです。</p> <p>詳細は、『Oracle9i XML データベース開発者ガイド - Oracle XML DB』を参照してください。</p>

関連項目： この章の「[索引の作成](#)」を参照してください。

テキストの場所

ドキュメントのテキストは、テキスト表、ファイル・システムまたは World Wide Web のいずれかの場所に配置できます。CREATE INDEX で索引付けするときに、データストア・プリファレンスを使用してその場所を指定します。使用しているアプリケーションに従って適切なデータストアを使用します。

次の表は、データストア・プリファレンス型を使用してテキストを格納するすべての方法を説明しています。

データストア型	使用する場合
DIRECT_DATASTORE	データをテキスト列に内部的に格納する場合。各行は単一のドキュメントとして索引付けされます。 テキスト列には、VARCHAR2、CLOB、BLOB、CHAR または BFILE を使用できます。XMLType 列は、CONTEXT 索引タイプでサポートされています。
MULTI_COLUMN_DATASTORE	データをテキスト表の複数の列に格納する場合。列が連結され、各行に 1 つずつ仮想ドキュメントが作成されます。
DETAIL_DATASTORE	データをテキスト列に内部的に格納する場合。ドキュメントがディテール表のテキスト列にある 1 つ以上の行で構成され、ヘッダー情報はマスター表に格納されます。
FILE_DATASTORE	データをオペレーティング・システム・ファイルに外部的に格納する場合。ファイル名が、テキスト列の各行に 1 つずつ格納されます。
NESTED_DATASTORE	データをネストした表に格納する場合。
URL_DATASTORE	データをイントラネットまたはインターネット上にあるファイルに外部的に格納する場合。Uniform Resource Locator (URL) がテキスト列に格納されます。
USER_DATASTORE	ドキュメントが、索引付け時にユーザー定義ストアド・プロセスによって合成されます。

URL の索引付けの場合、索引付け時間とドキュメントの取出し時間が長くなります。これは、システムがネットワークからドキュメントを取り出す必要があるためです。

関連項目： この章の「[データストア例](#)」を参照してください。

ドキュメント形式とフィルタ処理

Microsoft Word や PDF などの書式設定されたドキュメントを索引付けするには、テキストにフィルタ処理する必要があります。システムが使用するフィルタ処理のタイプは、`FILTER` プリファレンス型によって決定されます。デフォルトでは、システムは `INSO_FILTER` フィルタ型を使用します。この型は、ドキュメントの形式を自動的に検出し、そのドキュメントをテキストにフィルタ処理します。

Oracle では、ほとんどの形式を索引付けできます。また、複合形式を持つドキュメントを含む列も索引付けできます。

フィルタ処理が不要な HTML

HTML やプレーン・テキストのファイルを索引付けする場合は、`INSO_FILTER` 型を使用しないでください。最も効率のよい方法として、`NULL_FILTER` プリファレンス型を使用します。

関連項目： この章の「[NULL_FILTER 例: HTML ドキュメントの索引付け](#)」を参照してください。

複合形式列のフィルタ処理

Microsoft Word、プレーン・テキストおよび HTML のドキュメントを含む列などの複合形式列がある場合は、テキスト表に形式列を組み込んで、プレーン・テキストや HTML のフィルタ処理をバイパスできます。形式列では、各行に `TEXT` または `BINARY` のタグを付けることができます。`TEXT` タグを付けた行は、フィルタ処理されません。

たとえば、HTML およびプレーン・テキストの行に `TEXT` タグを付け、Microsoft Word の行に `BINARY` タグを付けることができます。その形式列を `CREATE INDEX` の `PARAMETERS` 句で指定します。

カスタム・フィルタ処理

独自のカスタム・フィルタを作成し、ドキュメントをフィルタ処理して索引付けできます。ファイル・システムから実行する外部フィルタ、または PL/SQL や Java ストアド・プロシージャのような内部フィルタのいずれも作成できます。

外部カスタム・フィルタ処理を行う場合は、`USER_FILTER` フィルタ・プリファレンス型を使用します。

内部フィルタ処理を行う場合は、`PROCEDURE_FILTER` フィルタ型を使用します。

関連項目： この章の「[PROCEDURE_FILTER 例](#)」を参照してください。

索引付け時の行のバイパス

テキスト表内の索引付け対象外の行（イメージ・データなどを含む行）はバイパスできます。バイパスする場合は、表に形式列を作成し、IGNORE に設定します。その形式列の名前を、CREATE INDEX の PARAMETERS 句で指定します。

ドキュメントのキャラクタ・セット

索引付けエンジンは、フィルタ処理済みのテキストを、データベース・キャラクタ・セットであるとみなします。INSO_FILTER フィルタ型を使用すると、書式設定されたドキュメントは、データベース・キャラクタ・セットでテキストに変換されます。

ソースがテキストで、ドキュメント・キャラクタ・セットがデータベース・キャラクタ・セットでない場合は、INSO_FILTER フィルタ型または CHARSET_FILTER フィルタ型を使用すると、テキストを索引付け用に変換できます。

複合キャラクタ・セット列

ドキュメント・セットに、JA16EUC および JA16SJIS などの異なるキャラクタ・セットを持つドキュメントが含まれている場合は、キャラクタ・セット列を作成して、そのドキュメントを索引付けできます。この列に、ドキュメントのキャラクタ・セットの名前を行ごとに移入します。その列の名前を、CREATE INDEX 文の PARAMETERS 句で指定します。

ドキュメントの言語

Oracle では、ほとんどの言語を索引付けできます。デフォルトでは、索引付けするテキストの言語は、データベース設定で指定した言語であるとみなされます。

英語、フランス語、ドイツ語およびスペイン語のような空白で区切られた言語の索引付けには、BASIC_LEXER プリファレンス型を使用します。これらの言語の中には、代替スペル、複合語の索引付けおよび基本文字変換を使用できるものがあります。

日本語、中国語および韓国語も索引付けできます。

関連項目： これらの言語の索引付けの詳細は、『Oracle Text リファレンス』を参照してください。

BASIC_LEXER 以外の言語機能

BASIC_LEXER、日本語、中国語および韓国語の各レクサーを使用して、Oracle Text ではほとんどの言語にレクサー・ソリューションを提供しています。タイ語やアラビア語などの他言語の場合は、ユーザー定義のレクサー・インタフェースを使用して、独自のレクサー・ソリューションを作成できます。このインタフェースを使用すると、索引付けまたは問合せ中にドキュメントを処理する PL/SQL プロシージャまたは Java プロシージャを作成できます。

さらに、ユーザー定義レクサーを使用して、独自のテーマ・レクサー・ソリューションまたは言語処理エンジンを作成することもできます。

関連項目： このレクサーの詳細は、『Oracle Text リファレンス』を参照してください。

マルチ言語列の索引付け

英語、ドイツ語および日本語で書かれたドキュメントを含む列など、異なる言語のドキュメントを含むテキスト列を索引付けできます。マルチ言語列を索引付けするには、テキスト表に言語列が必要です。MULTI_LEXER プリファレンス型を使用します。

また、マルチ言語列の索引付け時に、マルチ言語ストップリストを取り込むこともできます。

関連項目： この章の「[MULTI_LEXER 例：マルチ言語表の索引付け](#)」を参照してください。

特殊文字の索引付け

BASIC_LEXER プリファレンス型を使用する場合は、ハイフンやピリオドなどの英数字以外の文字の索引付け方法を、それらの文字を含むトークンに対して指定できます。たとえば、*web-site* のようなワードを索引付けする場合、ハイフン文字 (-) を組み込むかまたは除外するかを指定できます。

これらの文字は、索引付け時に要求する動作に基づいて、BASIC_LEXER カテゴリに分類されます。索引付け用に設定したレクサーの動作は、問合せ解析用のレクサーの動作と同じです。

次に、設定できる特殊文字をいくつか示します。

printjoin 文字

索引付け時に英数字以外の文字をトークンに組み込む場合は、その文字を **printjoin** として定義します。

たとえば、索引にハイフンやアンダースコアの文字を組み込む場合は、その文字を **printjoin** として定義します。この場合、*web-site* のようなワードは、*web-site* として索引付けされます。*website* を問い合せても、*web-site* は検索されません。

関連項目： この章の「[BASIC_LEXER 例：printjoin 文字の設定](#)」を参照してください。

skipjoin 文字

英数字以外の文字を含むトークンを使用して索引付けしない場合は、その文字を **skipjoin** として定義します。

たとえば、ハイフン (-) 文字を `skipjoin` として定義した場合、ワード *web-site* は、*website* として索引付けされます。*web-site* を問い合わせると、*website* と *web-site* を含むドキュメントが検索されます。

その他の文字

その他の文字は、別のトークン化動作を制御する文字として指定できます。たとえば、トークン分割 (`startjoin`、`endjoin`、`whitespace`)、句読点識別 (`punctuation`)、数値トークン化 (`numjoin`) および改行後のワード継続 (`continuation`) などの動作が含まれます。これらのカテゴリの文字にはデフォルトがありますが、変更できます。

関連項目： `BASIC_LEXER` の詳細は、『Oracle Text リファレンス』を参照してください。

大 / 小文字を区別した索引付けおよび問合せ

デフォルトでは、すべてのテキスト・トークンが大文字に変換された後で索引付けされます。つまり、大 / 小文字を区別しない問合せになります。たとえば、3 つのワード *cat*、*CAT* および *Cat* をそれぞれ個別に問い合わせても、すべて同じドキュメントが戻ります。

デフォルトを変更し、索引レコード・トークンをテキストの表示どおりに設定できます。大 / 小文字を区別する索引を作成した場合は、ドキュメントに一致するように大 / 小文字を正確に区別して問合せを指定する必要があります。たとえば、ドキュメントに *Cat* が含まれている場合は、このドキュメントに一致するように、問合せを *Cat* と指定する必要があります。*cat* または *CAT* と指定すると、ドキュメントは戻りません。

大 / 小文字を区別する索引付けを使用可能または使用禁止にするには、`BASIC_LEXER` プリファレンスの `mixed_case` 属性を使用します。

関連項目： `BASIC_LEXER` の詳細は、『Oracle Text リファレンス』を参照してください。

各国語別の機能

索引付け時に、次の各国語別の機能を使用可能にできます。

テーマの索引付け

英語とフランス語の場合は、ドキュメントのテーマ情報を索引付けできます。ドキュメント・テーマとは、ドキュメントの主要概念です。テーマは、ABOUT 演算子で問い合わせることができます。

他の言語のナレッジ・ベースがロードされ、コンパイルされている場合は、その言語でテーマ情報を索引付けできます。

デフォルトでは、テーマは英語とフランス語で索引付けされます。テーマの索引付けは、BASIC_LEXER プリファレンス型の `index_themes` 属性を使用して、使用可能および使用禁止にできます。

関連項目： BASIC_LEXER の詳細は、『Oracle Text リファレンス』を参照してください。

第 3 章「問合せ」の「ABOUT 問合せおよびテーマ」

発音区別記号を持つ文字の基本文字変換

一部の言語には、ティルデ、ウムラウトおよびアクセントなどの発音区別記号を持つ文字が含まれています。索引付け操作で、発音区別記号を含むワードが基本文字書式に変換された場合は、一致のスコアを取得するために、問合せに発音区別記号を含める必要はありません。たとえば、スペイン語の基本文字索引を使用すると、*energía* の問合せでは、索引内の *energía* および *energia* が一致します。

ただし、基本文字の索引付けが使用禁止になっていると、*energía* を問い合わせても、一致するのは *energía* のみです。

言語の基本文字の索引付けは、BASIC_LEXER プリファレンス型の `base_letter` 属性を使用して、使用可能および使用禁止にできます。

関連項目： BASIC_LEXER の詳細は、『Oracle Text リファレンス』を参照してください。

代替スペル

ドイツ語、デンマーク語およびスウェーデン語などの言語には、複数のスペルが容認されているワードがあります。たとえば、ドイツ語では、*ä* という文字は、*ae* という文字で代替できます。*ae* 文字は、基本文字書式と呼ばれます。

これらの言語の場合、デフォルトでは基本文字書式のワードが索引付けされます。問合せ語句もそれぞれの基本文字書式に変換されます。その結果、これらのワードをいずれのスペルでも問い合わせることができます。

言語の代替スペルは、BASIC_LEXER プリファレンス型の `alternate_spelling` 属性を使用して、使用可能および使用禁止にできます。

関連項目： BASIC_LEXER の詳細は、『Oracle Text リファレンス』を参照してください。

複合語

ドイツ語とオランダ語のテキストには、複合語が含まれています。これらの言語の場合、デフォルトではコンポジット索引が作成されます。その結果、ある語句を問い合わせると、その語句を複合語の要素として含むワードが戻ります。

たとえば、ドイツ語で、語句 *Bahnhof* (列車の駅) を問い合わせると、*Bahnhof* を含むドキュメント、あるいは *Hauptbahnhof*、*Nordbahnhof* または *Ostbahnhof* など、*Bahnhof* が複合語の要素として含まれているワードを含むドキュメントが戻ります。

コンポジット索引の作成は、BASIC_LEXER プリファレンス型の `composite` 属性を使用して、使用可能および使用禁止にできます。

関連項目： BASIC_LEXER の詳細は、『Oracle Text リファレンス』を参照してください。

韓国語、日本語および中国語の索引付け

これらの言語の索引付けには、次の特殊なレクサーを使用します。

言語	レクサー
韓国語	KOREAN_MORPH_LEXER
日本語	JAPANESE_LEXER
中国語	CHINESE_VGRAM_LEXER

KOREAN_MORPH_LEXER には、索引付けを制御する独自の属性セットがあります。機能には、複合語の索引付けも組み込まれています。

関連項目： これらのレクサーの詳細は、『Oracle Text リファレンス』を参照してください。

ファジー・マッチングおよびステミング

問合せでファジー・マッチングを使用すると、類似するスペルのワードが一致します。ステミングを使用すると、同じ語幹を持つワードが一致します。

ファジー・マッチングとステミングは、**Oracle Text** が使用言語に対してこの機能をサポートしている場合は、索引で自動的に使用可能になります。

ファジー・マッチングは、デフォルトのパラメータで使用可能です。ただし、類似度スコアの下限および拡張される語句の最大数が指定されています。索引時にこのデフォルト・パラメータを変更できます。

ステミング問合せのパフォーマンスを改善するには、**BASIC_LEXER** の `index_stems` 属性を使用可能にして、ステミング索引を作成します。

関連項目：『Oracle Text リファレンス』

ワイルド・カード問合せのパフォーマンスの向上

ワイルド・カード問合せを使用すると、`%ing`、`cos%` または `%benz%` のような後方一致、前方一致および中間一致の各問合せを発行できます。通常の索引付けでは、これらの問合せは、問合せパフォーマンスを無視して、大きなワード・リストに拡張されることがあります。

ワイルド・カード問合せでは、トークン・プリフィックスとサブストリングが索引に記録されていると、応答時間が短縮されます。

デフォルトでは、トークン・プリフィックスとサブストリングは、**Oracle Text** の索引に記録されません。使用している問合せアプリケーションがワイルド・カード問合せを頻繁に使用する場合は、トークン・プリフィックスとサブストリングの索引付けを考慮してください。この索引付けを実行するには、**wordlist** プリファレンス型を使用します。ワイルド・カード検索のパフォーマンスを向上させるために必要なトレードオフは、索引が大きくなることです。

関連項目： この章の「[BASIC_WORDLIST 例: サブストリングとプリフィックスの索引付けの使用可能化](#)」を参照してください。

ドキュメントのセクション検索

HTML や XML のように、内部構造を持つドキュメントの場合は、ドキュメントのセクションを定義および索引付けできます。ドキュメントのセクションを索引付けすると、問合せの範囲を事前定義したセクションに絞り込むことができます。たとえば、問合せに対して、*Headings* として定義したセクション内の語句 *dog* を含むドキュメントすべてを検索するように指定できます。

セクションは、索引付け前に定義し、セクション・グループ・プリファレンスを使用して指定する必要があります。

Oracle Text には、システム定義の HTML および XML 用のセクションが定義されたセクション・グループが用意されています。また、索引付け時に、XML ドキュメントからセクションをシステムで自動的に作成するように指定できます。

関連項目： [第 6 章「ドキュメントのセクション検索」](#)

ストップワードとストップテーマ

ストップワードは、索引付け対象外のワードです。通常、ストップワードは、特定の言語で下位レベルの情報を提供するワードを指します。英語では、*this* や *that* のようなワードです。

デフォルトでは、特定言語の索引付け用に、ストップリストと呼ばれるストップワードのリストが用意されています。CTX_DDL パッケージを使用してこのリストを変更したり、独自のリストを作成できます。ストップリストは、CREATE INDEX のパラメータ文字列に指定します。

ストップテーマは、テーマ索引またはテーマの構成要素に使用できないワードです。ストップテーマは、CTX_DDL パッケージを使用して追加できます。

ドキュメントのテーマは、ABOUT 演算子を使用して問い合わせることができます。また、PL/SQL パッケージ CTX_DOC を使用して、プログラムで取り出すこともできます。

マルチ言語のストップリスト

言語固有のストップワードを保持するマルチ言語のストップリストを作成することもできます。マルチ言語のストップリストは、MULTI_LEXER を使用した英語、ドイツ語および日本語などの異なる言語のドキュメントを含む表の索引付けに有効です。

索引付け時に、各ドキュメントの言語列が調べられ、その言語のストップワードのみが排除されます。問合せ時に、セッション言語の設定によって、アクティブなストップワードが特定されます。これは、マルチレクサーの使用時にアクティブなレクサーが特定されるのと同様です。

索引のパフォーマンス

索引のパフォーマンスに影響する要因には、メモリー割当て、ドキュメント形式、並列度およびパーティション表などがあります。

関連項目： [第5章「パフォーマンス・チューニング」の「索引付けのパフォーマンスに関する FAQ（よくある質問）」](#)

問合せのパフォーマンスと LOB 列の格納

問合せで頻繁にアクセスされるが、ほとんど更新されない LOB 構造化列を含む表の場合は、このような列を表外に格納することで問合せのパフォーマンスを向上させることができます。

関連項目： [第5章「パフォーマンス・チューニング」の「元表の LOB 列を表外に格納すると、パフォーマンスが向上しますか？」](#)

索引の作成

Oracle Text では、CONTEXT、CTXCAT および CTXRULE という 3 つのタイプの索引を作成できます。

CONTEXT 索引の作成手順

デフォルトでは、システムはドキュメントがテキスト列に格納されるとみなします。この要件が満たされている場合は、プリファレンスを明示的に指定せずに、SQL コマンド CREATE INDEX を使用してテキスト索引を拡張可能な索引タイプ CONTEXT として作成できます。システムでは、言語、テキスト列のデータ型およびドキュメントの形式を自動的に検出し、それに従って索引プリファレンスを設定します。

関連項目： このデフォルトの詳細は、この章の「[デフォルトの CONTEXT 索引例](#)」を参照してください。

Oracle Text の索引を作成する手順は、次のとおりです。

1. デフォルトを使用しない場合は、必要に応じて、カスタムの索引プリファレンス、セクション・グループまたはストップリストを指定します。次の表は、これらの索引クラスを説明しています。

クラス	説明
データストア	ドキュメントの保存方法
フィルタ	ドキュメントのプレーン・テキストへの変換方法
レクサー	索引付け対象の言語
ワードリスト	ステミング問合せおよびファジー問合せの拡張方法
記憶域	索引データの格納方法
ストップリスト	索引付け対象外のワードまたはテーマ
セクション・グループ	ドキュメント・セクションの定義方法

関連項目： この章の「[索引付け時の考慮事項](#)」と『Oracle Text リファレンス』を参照してください。

2. 必要に応じて、独自のカスタム・プリファレンス、セクション・グループまたはストップリストを作成します。この章の「[プリファレンスの作成](#)」を参照してください。
3. SQL コマンド CREATE INDEX を使用してテキスト索引を作成し、索引名を指定し、必要に応じてプリファレンスを指定します。この章の「[索引の作成](#)」を参照してください。

プリファレンスの作成

必要に応じて、独自のカスタム索引プリファレンスを作成し、デフォルトを変更できます。ファイルの格納場所やドキュメントのフィルタ処理方法などの索引情報を指定するには、プリファレンスを使用します。プリファレンスを作成した後、属性を設定します。

データストア例

次の各項では、ダイレクト、複数列、URL およびファイルのデータストアの設定例を示します。

関連項目： データ記憶域の詳細は、『Oracle Text リファレンス』を参照してください。

DIRECT_DATASTORE の指定 次の例では、テキスト・データを格納する CLOB 列がある表を作成します。表の作成後、2 行のテキスト・データを移入し、システム定義プリファレンス CTXSYS.DEFAULT_DATASTORE を使用して表を索引付けします。

```
create table mytable(id number primary key, docs clob);

insert into mytable values(111555,'this text will be indexed');
insert into mytable values(111556,'this is a direct_datastore example');
commit;

create index myindex on mytable(docs)
  indextype is ctxsys.context
  parameters ('DATASTORE CTXSYS.DEFAULT_DATASTORE');
```

MULTI_COLUMN_DATASTORE の指定 次の例では、これらのテキスト列は、連結されて索引付けされます。3 つのテキスト列に my_multi という複数列のデータストア・プリファレンスを作成します。

```
begin
ctx_ddl.create_preference('my_multi', 'MULTI_COLUMN_DATASTORE');
ctx_ddl.set_attribute('my_multi', 'columns', 'column1, column2, column3');
end;
```

URL データ記憶域の指定 この例では、my_url という URL_DATASTORE プリファレンスを作成し、http_proxy、no_proxy および timeout の各属性を設定します。属性を設定しない場合は、デフォルトが使用されます。

```
begin
ctx_ddl.create_preference('my_url', 'URL_DATASTORE');
ctx_ddl.set_attribute('my_url', 'HTTP_PROXY', 'www-proxy.us.oracle.com');
ctx_ddl.set_attribute('my_url', 'NO_PROXY', 'us.oracle.com');
ctx_ddl.set_attribute('my_url', 'Timeout', '300');
end;
```

ファイル・データ記憶域の指定 次の例では、FILE_DATASTORE を使用してデータ記憶域プリファレンスを作成します。このプリファレンスによって、索引付け対象のファイルがオペレーティング・システムに格納されていることをシステムに知らせます。この例では、CTX_DDL.SET_ATTRIBUTE を使用して、PATH 属性をディレクトリ /docs に設定します。

```
begin
ctx_ddl.create_preference('mypref', 'FILE_DATASTORE');
ctx_ddl.set_attribute('mypref', 'PATH', '/docs');
end;
```


NULL_FILTER 例 : HTML ドキュメントの索引付け

ドキュメント・セット全体が HTML である場合は、フィルタ・プリファレンスに NULL_FILTER を使用する（フィルタ処理を行わない）ことをお勧めします。

たとえば、HTML ドキュメント・セットを索引付けするには、NULL_FILTER および HTML_SECTION_GROUP に対するシステム定義プリファレンスを次のように指定できます。

```
create index myindex on docs(htmlfile) indextype is ctxsys.context
  parameters('filter ctxsys.null_filter
             section group ctxsys.html_section_group');
```

PROCEDURE_FILTER 例

次のシグネチャを使用して定義するフィルタ・プロシージャ CTXSYS.NORMALIZE を考えてみます。

```
PROCEDURE NORMALIZE(id IN ROWID, charset IN VARCHAR2, input IN CLOB,
output IN OUT NOCOPY VARCHAR2);
```

このプロシージャをフィルタとして使用するには、次のようにフィルタ・プリファレンスを設定します。

```
begin
  ctx_ddl.create_preference('myfilt', 'procedure_filter');
  ctx_ddl.set_attribute('myfilt', 'procedure', 'normalize');
  ctx_ddl.set_attribute('myfilt', 'input_type', 'clob');
  ctx_ddl.set_attribute('myfilt', 'output_type', 'varchar2');
  ctx_ddl.set_attribute('myfilt', 'rowid_parameter', 'TRUE');
  ctx_ddl.set_attribute('myfilt', 'charset_parameter', 'TRUE');
end;
```

BASIC_LEXER 例 : printjoin 文字の設定

printjoin 文字は、索引トークンに組み込まれる英数字以外の文字を指します。*web-site* のようなワードを、*web-site* として索引付けしたい場合に使用します。

次の例では、BASIC_LEXER を使用して、printjoin 文字をハイフンとアンダースコアに設定します。

```
begin
  ctx_ddl.create_preference('mylex', 'BASIC_LEXER');
  ctx_ddl.set_attribute('mylex', 'printjoins', '_-');
end;
```


前述のように、`printjoin` 文字を設定した索引を作成するには、次の文を発行します。

```
create index myindex on mytable ( docs )
  indextype is ctxsys.context
  parameters ( 'LEXER mylex' );
```

MULTI_LEXER 例：マルチ言語表の索引付け

異なる言語のドキュメントを含む列を索引付けするには、`MULTI_LEXER` プリファレンス型を使用します。たとえば、テキスト列に英語、ドイツ語およびフランス語のドキュメントが格納されている場合は、このプリファレンス型を使用できます。

最初に、次のように、主キー、テキスト列および言語列を持つマルチ言語表を作成します。

```
create table globaldoc (
  doc_id number primary key,
  lang varchar2(3),
  text clob
);
```

表に格納されているドキュメントのほとんどが英語で、ドイツ語または日本語のドキュメントがいくつかあるとします。この3つの言語を処理するには、英語、ドイツ語および日本語に対して1つずつの、3つのサブレクサーを作成する必要があります。

```
ctx_ddl.create_preference('english_lexer','basic_lexer');
ctx_ddl.set_attribute('english_lexer','index_themes','yes');
ctx_ddl.set_attribute('english_lexer','theme_language','english');

ctx_ddl.create_preference('german_lexer','basic_lexer');
ctx_ddl.set_attribute('german_lexer','composite','german');
ctx_ddl.set_attribute('german_lexer','mixed_case','yes');
ctx_ddl.set_attribute('german_lexer','alternate_spelling','german');

ctx_ddl.create_preference('japanese_lexer','japanese_vgram_lexer');
```

マルチレクサー・プリファレンスを作成します。

```
ctx_ddl.create_preference('global_lexer','multi_lexer');
```

格納されているドキュメントのほとんどが英語であるため、`CTX_DDL.ADD_SUB_LEXER` を使用して、英語のレクサーをデフォルトに設定します。

```
ctx_ddl.add_sub_lexer('global_lexer','default','english_lexer');
```

ここで、`CTX_DDL.ADD_SUB_LEXER` プロシージャを使用して、ドイツ語および日本語のレクサーをそれぞれの言語に追加します。また、言語列が ISO 標準言語コード 639-2 で表現されている場合は、これらを代替値として追加します。

```
ctx_ddl.add_sub_lexer('global_lexer','german','german_lexer','ger');
ctx_ddl.add_sub_lexer('global_lexer','japanese','japanese_lexer','jpn');
```


次のように、PARAMETERS 句にマルチレクサー・プリファレンスおよび言語列を指定して、索引 globalx を作成します。

```
create index globalx on globaldoc(text) indextype is ctxsys.context
parameters ('lexer global_lexer language column lang');
```

BASIC_WORDLIST 例：サブストリングとプリフィックスの索引付けの使用可能化

次の例では、プリフィックスとサブストリングの索引付けに対してワードリスト・プリファレンスを設定します。プリフィックスとサブストリングのコンポーネントを索引に設定すると、ワイルド・カード問合せのパフォーマンスが向上します。

プリフィックス索引付けに対して、3～4文字の長さのトークン・プリフィックスを作成するように指定します。

```
begin
  ctx_ddl.create_preference('mywordlist', 'BASIC_WORDLIST');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_INDEX', 'TRUE');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_MIN_LENGTH', '3');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_MAX_LENGTH', '4');
  ctx_ddl.set_attribute('mywordlist', 'SUBSTRING_INDEX', 'YES');
end;
```

セクション検索のためのセクション・グループの作成

HTML や XML のように、ドキュメントに内部構造がある場合は、索引付け操作前に埋込みタグを使用してドキュメントのセクションを定義できます。この定義によって、WITHIN 演算子を使用してセクション内で問合せができます。セクションは、セクション・グループの構成要素として定義します。

例：HTML のセクションの作成

次のコードは、HTML_SECTION_GROUP 型の htmgroup というセクション・グループを定義します。その後、htmgroup に、<H1> タグで識別する heading というゾーン・セクションを作成します。

```
begin
  ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
  ctx_ddl.add_zone_section('htmgroup', 'heading', 'H1');
end;
```

関連項目： [第6章「ドキュメントのセクション検索」](#)

ストップワードおよびストップリストの使用

ストップワードは、索引付け対象外のワードです。ストップワードは、通常、英語の *this* や *that* のような下位レベルの情報を提供するワードを指します。

システムには、すべての言語に対してストップリストというストップワードのリストが用意されています。デフォルトでは、索引付け時に使用言語に対して、Oracle Text のデフォルトのストップリストが使用されます。

デフォルトのストップリスト CTXSYS.DEFAULT_STOPLIST を編集するか、次の PL/SQL プロシージャを使用して独自のリストを作成できます。

- CTX_DDL.CREATE_STOPLIST
- CTX_DDL.ADD_STOPWORD
- CTX_DDL.REMOVE_STOPWORD

CREATE INDEX の PARAMETERS 句に、カスタムのストップリストを指定します。

また、索引を作成した後で、ALTER INDEX 文でストップワードを動的に追加することもできます。

マルチ言語のストップリスト

各言語固有のストップワードを保持するマルチ言語ストップリストを作成できます。マルチ言語のストップリストは、英語、ドイツ語および日本語などの異なる言語のドキュメントを含む表を索引付けするために MULTI_LEXER を使用している場合に有効です。

マルチ言語ストップリストを作成するには、CTX_DDL.CREATE_STOPLIST プロシージャを使用し、MULTI_STOPLIST というストップリスト・タイプを指定します。CTX_DDL.ADD_STOPWORD を使用して言語固有のストップワードを追加します。

ストップテーマとストップクラス

独自のストップワードの定義に加えて、ストップテーマの定義もできます。ストップテーマは、索引付け対象外のテーマです。この機能が使用できるのは、英語のみです。

数値が索引付け対象外であることも指定できます。索引付け対象外の数値のような英数字のクラスは、ストップクラスと呼ばれます。

独自のストップワード、ストップテーマおよびストップクラスを記録するには、それらを追加するためのストップリストを 1 つ作成します。そのストップリストを CREATE INDEX の PARAMETERS 句で指定します。

ストップリスト管理のための PL/SQL プロシージャ

次のプロシージャを使用してストップリスト、ストップワード、ストップテーマおよびストップクラスを管理します。

- CTX_DDL.CREATE_STOPLIST
- CTX_DDL.ADD_STOPWORD
- CTX_DDL.ADD_STOPTHEME
- CTX_DDL.ADD_STOPCLASS
- CTX_DDL.REMOVE_STOPWORD
- CTX_DDL.REMOVE_STOPTHEME
- CTX_DDL.REMOVE_STOPCLASS
- CTX_DDL.DROP_STOPLIST

関連項目： これらのコマンドの使用方法は、『Oracle Text リファレンス』を参照してください。

索引の作成

SQL コマンド `CREATE INDEX` を使用して、拡張可能な索引として Oracle Text の索引を作成します。

次の 3 つのタイプの索引を作成できます。

- CONTEXT
- CTXCAT
- CTXRULE

CONTEXT 索引の作成

CONTEXT 索引タイプは、MS Word、HTML またはプレーン・テキストのような大量のまとまったドキュメントを索引付けする場合に最適です。CONTEXT 索引では、様々な方法で索引をカスタマイズすることもできます。

ドキュメントはテキスト表にロードする必要があります。

デフォルトの CONTEXT 索引例

次のコマンドによって、docs 表内の text 列に myindex というデフォルトの CONTEXT 索引を作成します。

```
CREATE INDEX myindex ON docs(text) INDEXTYPE IS CTXSYS.CONTEXT;
```

パラメータを明示的に指定せずに CREATE INDEX を使用すると、すべての言語に対してシステムのデフォルト動作が次のようになります。

- 索引付けされるテキストは、テキスト列に直接格納されるとみなします。テキスト列の型は、CLOB、BLOB、BFILE、VARCHAR2 または CHAR になります。
- 列型を検出し、バイナリの列型に対してフィルタ処理を使用します。ほとんどのドキュメント形式がフィルタ処理でサポートされています。列がプレーン・テキストの場合、システムはフィルタ処理を使用しません。

注意： システムで正しくドキュメントをフィルタ処理するために、使用している環境が Inso フィルタをサポートするように正しく設定されていることを確認してください。

Inso フィルタを使用するための環境の構成方法は、『Oracle Text リファレンス』を参照してください。

- 索引付けするテキストの言語は、データベース設定で指定した言語であるとみなします。
- データベース設定で指定した言語に対するデフォルトのストップリストを使用します。ストップリストは、索引付け時にシステムが無視するワードを識別します。
- 使用言語でのファジーおよびステミング問合せを有効にします（その言語に対してこの機能が使用可能な場合）。

デフォルトの索引付け動作は、ユーザー独自のプリファレンスを作成し、このカスタム・プリファレンスを CREATE INDEX のパラメータ文字列に指定することによって、いつでも変更できます。

カスタムの CONTEXT 索引例：HTML ドキュメントの索引付け

URL で指定された HTML ドキュメント・セットを索引付けする場合は、CREATE INDEX 文でシステム定義プリファレンスである NULL_FILTER を指定できます。

HTML_SECTION_GROUP を使用するセクション・グループ htmgroup および URL_DATASTORE を使用するデータストア my_url を次のように指定できます。


```
begin
  ctx_ddl.create_preference('my_url','URL_DATASTORE');
  ctx_ddl.set_attribute('my_url','HTTP_PROXY','www-proxy.us.oracle.com');
  ctx_ddl.set_attribute('my_url','NO_PROXY','us.oracle.com');
  ctx_ddl.set_attribute('my_url','Timeout','300');
end;

begin
  ctx_ddl.create_section_group('htmgroup','HTML_SECTION_GROUP');
  ctx_ddl.add_zone_section('htmgroup','heading','H1');
end;

create index myindex on docs(htmlfile) indextype is ctxsys.context
parameters('datastore my_url filter ctxsys.null_filter section group htmgroup');
```

関連項目： カスタムの CONTEXT 索引の作成例は、この章の「[プリファレンスの作成](#)」を参照してください。

CTXCAT 索引の作成

CTXCAT 索引タイプは、小さなテキスト断片や関連情報の索引付けに最適です。適切に作成されている場合は、この索引タイプを使用するほうが、CONTEXT 索引に比べて構造化問合せのパフォーマンスが向上します。

CTXCAT 索引と DML

CTXCAT 索引は、トランザクション・ベースで更新されます。元表で DML（挿入、更新および削除）を実行すると、索引が自動的に同期化されます。CONTEXT 索引と異なり、CTX_DDL.SYNC_INDEX は不要です。

注意： トリガーを起動せずに挿入を実行するアプリケーション（SQL*Loader など）では、前述の索引の自動同期化は行われません。

CTXCAT サブ索引とそのコスト

CTXCAT 索引は、索引セットの構成要素として定義したサブ索引で構成されています。1 つ以上の列にサブ索引を作成すると、複合問合せのパフォーマンスが向上します。

ただし、索引セットにサブ索引を追加すると、コストがかかります。CTXCAT 索引の作成に要する時間は、その総サイズによって異なります。CTXCAT 索引の総サイズに直接関係する要因は、次のとおりです。

- 索引付け対象の総テキスト
- 索引セットに含まれるサブ索引の数
- サブ索引を構成する元表の列数

索引セットに多数のコンポーネント索引がある場合、更新が必要な索引が増えるため、DMLのパフォーマンスは低下します。

CTXCAT 索引の作成には追加の索引付け時間とディスク領域が必要になるため、索引セットにコンポーネント索引を追加する前に、それぞれのコンポーネント索引によってアプリケーションに提供される問合せパフォーマンス上のメリットを慎重に考慮する必要があります。

CTXCAT サブ索引の作成

オンライン・オークション・サイトでは、品目の説明、価格、入札終了日などの情報を格納し、参照する必要があるため、CTXCAT 索引作成のよい例です。

図 2-2 オークション表の概要と CTXCAT 索引

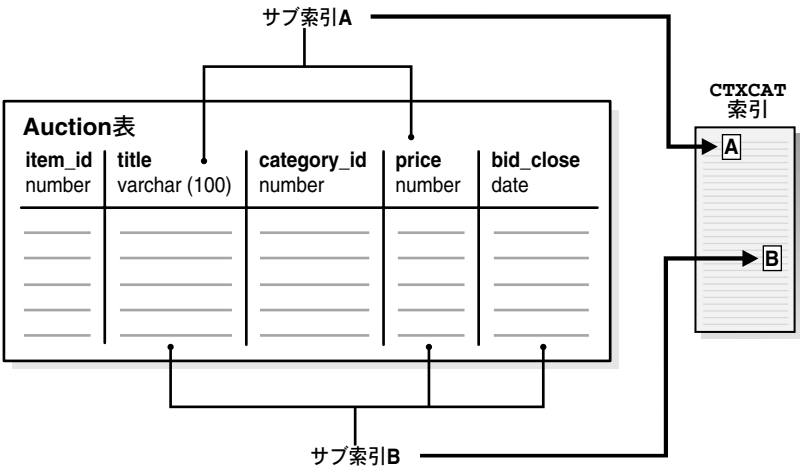


図 2-2 は、次のスキーマを使用した AUCTION という表を示しています。

```
create table auction(  
  item_id number,  
  title varchar2(100),  
  category_id number,  
  price number,  
  bid_close date);
```


サブ索引を作成するには、サブ索引を格納する索引セットを作成します。

```
begin
  ctx_ddl.create_index_set('auction_iset');
end;
```

次に、アプリケーションが発行する可能性がある構造化問合せを特定します。CATSEARCH 問合せ演算子では、必須のテキスト句とオプションの構造化句を使用します。

この例では、すべての問合せにテキスト列である title 列の句が組み込まれています。構造化句は、次のカテゴリに分類されるとします。

構造化句	問合せを満たす サブ索引の定義	カテゴリ
'price < 200'	'price'	A
'price = 150'		
'order by price'		
'price = 100 order by bid_close'	'price, bid_close'	B
'order by price, bid_close'		

構造化問合せ句カテゴリ A 構造化問合せ句には、価格列のみの式が含まれています。

```
SELECT FROM auction WHERE CATSEARCH(title, 'camera', 'price < 200')> 0;
SELECT FROM auction WHERE CATSEARCH(title, 'camera', 'price = 150')> 0;
SELECT FROM auction WHERE CATSEARCH(title, 'camera', 'order by price')> 0;
```

これらの問合せは、サブ索引 B を使用して満たすことができます。しかし、効率を上げるために、price のみにサブ索引を作成することもできます。それをここではサブ索引 A とします。

```
begin
  ctx_ddl.add_index('auction_iset','price'); /* sub-index A */
end;
```

構造化問合せ句カテゴリ B 構造化問合せ句には、price についての等式に bid_close でのソートの条件が付与されたもの、および price と bid_close の両方で（この順に）ソートの条件が付与されたものが含まれます。

```
SELECT FROM auction WHERE CATSEARCH(title, 'camera','price = 100 order by bid_
close')> 0;
SELECT FROM auction WHERE CATSEARCH(title, 'camera','order by price, bid_close')> 0;
```


これらの問合せは、次のように定義したサブ索引で満たすことができます。

```
begin
    ctx_ddl.add_index('auction_iset','price, bid_close'); /* sub-index B */
end;
```

結合された B ツリー索引と同様に、CTX_DDL.ADD_INDEX を使用して指定する列の順序は、特定の問合せを満たすために使用される索引スキンの効率性および実行可能性に影響します。たとえば、2つの構造化列 p と q に B ツリー索引が 'p,q' として指定されているとします。Oracle では、ソート 'order by q,p' を実行するために、この索引をスキャンできません。

CTXCAT 索引の作成

次の例では、前述の例を結合し、2つのサブ索引を持つ索引セット・プリファレンスを作成します。

```
begin
    ctx_ddl.create_index_set('auction_iset');
    ctx_ddl.add_index('auction_iset','price'); /* sub-index A */
    ctx_ddl.add_index('auction_iset','price, bid_close'); /* sub-index B */
end;
```

[図 2-2](#) は、サブ索引 A および B を AUCTION 表から作成する方法を示しています。各サブ索引はそれぞれテキスト列と名前付き構造化列の B ツリー索引です。たとえば、サブ索引 A は、title 列と bid_close 列の索引です。

結合されたカタログ索引は、CREATE INDEX を使用して次のように作成します。

```
CREATE INDEX auction_titlex ON AUCTION(title) INDEXTYPE IS CTXSYS.CTXCAT PARAMETERS
('index set auction_iset');
```

関連項目： CREATE INDEX を使用した CTXCAT 索引の作成方法の詳細は、『Oracle Text リファレンス』を参照してください。

CTXRULE 索引の作成

ドキュメント分類アプリケーションの作成には、CTXRULE 索引を使用します。問合せ表を作成後、問合せを索引付けします。CTXRULE 索引では、MATCHES 演算子を使用して単一のドキュメントを分類できます。

問合せ表の作成

最初に、分類を定義する問合せ表を作成します。ここでは、カテゴリ名と問合せテキストを保持する表 `myqueries` を次のように作成します。

```
CREATE TABLE myqueries (  
    queryid NUMBER PRIMARY KEY,  
    category VARCHAR2(30),  
    query VARCHAR2(2000)  
);
```

表に分類と各分類を定義する問合せを移入します。たとえば、*US Politics*、*Music* および *Soccer* などの主題の分類を考えます。

```
INSERT INTO myqueries VALUES(1, 'US Politics', 'democrat or republican');  
INSERT INTO myqueries VALUES(2, 'Music', 'ABOUT(music)');  
INSERT INTO myqueries VALUES(3, 'Soccer', 'ABOUT(soccer)');
```

CTX_CLS.TRAIN の使用方法 ルール（問合せ）の表の生成には、CTX_CLS.TRAIN プロシージャも使用できます。このプロシージャは入力としてドキュメント・トレーニング・セットを取ります。

関連項目： CTX_CLS.TRAIN の詳細は、『Oracle Text リファレンス』を参照してください。

CTXRULE 索引の作成

CTXRULE 索引を作成するには、CREATE INDEX を使用します。必要に応じて、次のように、lexer、storage、section group および wordlist の各パラメータを指定できます。

```
CREATE INDEX ON myqueries(query) INDEXTYPE IS CTXSYS.CTXRULE PARAMETERS('lexer  
lexer_pref storage storage_pref section group section_pref wordlist wordlist_pref');
```

注意： filter、memory、datastore、stoplist および [no]populate の各パラメータは、CTXRULE 索引タイプには使用できません。

ドキュメントの分類

問合せセットに作成された CTXRULE 索引を使用すれば、MATCHES 演算子を使ったドキュメント分類が可能です。

着信ドキュメントは表 news に格納されるとします。

```
CREATE TABLE news (  
    newsid NUMBER,  
    author VARCHAR2(30),  
    source VARCHAR2(30),  
    article CLOB);
```

MATCHES で BEFORE INSERT トリガーを作成すると、各ドキュメントをそれぞれの分類に基づいて、別の表 news_route にルーティングできます。

```
BEGIN  
    -- find matching queries  
    FOR c1 IN (select category  
               from myqueries  
               where MATCHES(query, :new.article)>0)  
    LOOP  
        INSERT INTO news_route(newsid, category)  
            VALUES (:new.newsid, c1.category);  
    END LOOP;  
END;
```

索引のメンテナンス

この項では、エラーや索引付け時の障害が発生した場合の索引のメンテナンスについて説明します。

索引エラーの表示

索引付け操作は失敗したり、正常に完了しないことがあります。行の索引付け操作でエラーが発生すると、システムはそのエラーを Oracle Text のビューに記録します。

索引のエラーは、CTX_USER_INDEX_ERRORS を使用して表示できます。すべての索引のエラーを表示するには、CTXSYS として CTX_INDEX_ERRORS を使用します。

たとえば、索引に最後に発生したエラーを表示するには、次の問合せを発行します。

```
SELECT err_timestamp, err_text FROM ctx_user_index_errors ORDER BY err_timestamp  
DESC;
```

エラーのビューを消去するには、次の問合せを発行します。

```
DELETE FROM ctx_user_index_errors;
```


関連項目： これらのビューの詳細は、『Oracle Text リファレンス』を参照してください。

索引の削除

CREATE INDEX を使用して索引を再作成する前に、既存の索引を削除する必要があります。

索引を削除するには、SQL の DROP INDEX コマンドを使用します。

たとえば、newsindex という索引を削除するには、次の SQL コマンドを発行します。

```
DROP INDEX newsindex;
```

索引付け操作がクラッシュしたときなど、Oracle が索引の状態を判断できない場合は、前述のコマンドでは索引を削除できません。この場合は、次のコマンドを使用します。

```
DROP INDEX newsindex FORCE;
```

関連項目： このコマンドの詳細は、『Oracle Text リファレンス』を参照してください。

失敗した索引付けの再開

ALTER INDEX コマンドを使用すると、失敗した索引作成操作を再開できます。通常は、失敗した索引を調査して修正した後、その索引作成を再開します。

索引の最適化は定期的コミットされます。したがって、最適化操作が失敗しても、すべての最適化作業は保存されています。

関連項目： ALTER INDEX コマンド構文の詳細は、『Oracle Text リファレンス』を参照してください。

例：失敗した索引の再開

次のコマンドは、2MB のメモリーを使用して newsindex の索引付け操作を再開します。

```
ALTER INDEX newsindex REBUILD PARAMETERS('resume memory 2M');
```

索引の再構築

ALTER INDEX を使用すると、有効な索引を再構築できます。新しいプリファレンスを使用して索引付けする場合に、索引を再構築することがあります。

関連項目： ALTER INDEX コマンド構文の詳細は、『Oracle Text リファレンス』を参照してください。

例：索引の再構築

次のコマンドは、索引を再構築し、レクサー・プリファレンスを `my_lexer` で置き換えます。

```
ALTER INDEX newsindex REBUILD PARAMETERS('replace lexer my_lexer');
```

プリファレンスの削除

カスタム索引プリファレンスが不要になった場合は、そのプリファレンスを削除できます。

索引プリファレンスの削除には、プロシージャ `CTX_DDL.DROP_PREFERENCE` を使用します。

プリファレンスを削除しても、そのプリファレンスから作成された索引は影響を受けません。

関連項目： `CTX_DDL.DROP_PREFERENCE` プロシージャの構文の詳細は、『Oracle Text リファレンス』を参照してください。

例

次のコードは、プリファレンス `my_lexer` を削除します。

```
begin
ctx_ddl.drop_preference('my_lexer');
end;
```

CONTEXT 索引に関する DML 操作の管理

元表の DML 操作とは、元表に対するドキュメントの挿入、更新または削除操作のことです。この項では、DML 操作時に Oracle Text の CONTEXT 索引を監視、同期化および最適化する方法を説明します。

注意： CTXCAT 索引は、トランザクション・ベースで更新されます。したがって、元表を更新すると即時に更新されます。この項で説明する手動同期化は、CTXCAT 索引には不要です。

保留中の DML の表示

元表のドキュメントを挿入、更新または削除すると、その ROWID は、その索引が同期化されるまで DML キューに保持されます。このキューを表示するには、`CTX_USER_PENDING` ビューを使用します。

たとえば、すべての索引で保留中の DML を表示するには、次の文を発行します。

```
SELECT pnd_index_name, pnd_rowid, to_char(pnd_timestamp, 'dd-mon-yyyy hh24:mi:ss')
timestamp FROM ctx_user_pending;
```

この文によって、次の形式の出力結果が戻されます。

PND_INDEX_NAME	PND_ROWID	TIMESTAMP
-----	-----	-----
MYINDEX	AAADXnAABAAAS3SAAC	06-oct-1999 15:56:50

関連項目： このビューの詳細は、『Oracle Text リファレンス』を参照してください。

索引の同期化

索引を同期化すると、元表に対する保留中のすべての挿入、更新および削除が処理されます。同期化は、PL/SQL で CTX_DDL.SYNC_INDEX プロシージャを使用して実行します。

次の例では、2MB のメモリーを使用して索引を同期化します。

```
begin
    ctx_ddl.sync_index('myindex', '2M');
end;
```

バックグラウンド DML の設定

DBMS_JOB.SUBMIT プロシージャを使用して、定期的に CTX_DDL.SYNC_INDEX を自動的に実行するように設定できます。Oracle Text には、この設定に使用できる SQL スクリプトが組み込まれています。このスクリプトの場所は、次のとおりです。

```
$ORACLE_HOME/ctx/sample/script/drjobdml.sql
```

このスクリプトを使用するには、ユーザーがその索引の所有者であり、CTX_DDL パッケージの実行権限を所有している必要があります。また、使用している Oracle 初期化ファイルに job_queue_processes パラメータが設定されている必要があります。

たとえば、索引の同期化を myindex に対して 360 分ごとに実行するように設定するには、SQL*Plus で次の文を発行します。

```
SQL> @drjobdml myindex 360
```

関連項目： CTX_DDL.SYNC_INDEX コマンド構文の詳細は、『Oracle Text リファレンス』を参照してください。

索引の最適化

索引の同期化を頻繁に行うと、CONTEXT 索引が断片化する可能性があります。索引の断片化は、問合せ応答時間に悪影響を与える場合があります。CONTEXT 索引を最適化すると、索引の断片化とサイズが減少し、問合せパフォーマンスが向上します。

索引の最適化を理解するには、索引の構造と同期化の内容を理解する必要があります。

CONTEXT 索引の構造

CONTEXT 索引は、逆向きの索引です。つまり、各ワードにそのワードを含むドキュメントのリストが格納されています。たとえば、初期の索引付け操作が 1 つ終了すると、ワード DOG のエントリは次のようになります。

```
DOG DOC1 DOC3 DOC5
```

索引の断片化

新しいドキュメントが元表に追加されると、索引は新しい行の追加によって同期化されます。たとえば、ワード *dog* を含む新しいドキュメント (DOC 7) を元表に追加し、索引を同期化すると、次のようになります。

```
DOG DOC1 DOC3 DOC5  
DOG DOC7
```

続く DML 操作によって、新しい行が次のように作成されるとします。

```
DOG DOC1 DOC3 DOC5  
DOG DOC7  
DOG DOC9  
DOG DOC11
```

新しいドキュメントの追加と索引の同期化は、索引の断片化の原因になります。特に、バックグラウンド DML の場合は索引を頻繁に同期化するため、一般的に、バッチでの同期化に比べて断片化が増加します。

バッチ処理の頻度を少なくすると、ドキュメント・リストが長くなり、索引内の行数も減ります。したがって、断片化も減少します。

CTX_DDL.OPTIMIZE_INDEX を使用して、FULL (完全) または FAST (高速) モードで索引を最適化すると、索引の断片化を減少させることができます。

ドキュメントの無効性とガベージ・コレクション

ドキュメントを元表から削除すると、Oracle Text では、そのドキュメントに削除済みのマークが設定されます。ただし、索引は即時に変更されません。

古い情報が領域を占有していると、問合せ時に余分なオーバーヘッドが発生します。したがって、索引を FULL モードで最適化して古い情報を削除する必要があります。この処理をガベージ・コレクションといいます。元表に対する更新または削除を頻繁に行う場合は、ガベージ・コレクションのために、FULL モードで最適化を行うことが必要です。

単一のトークンの最適化

索引全体の最適化以外に、単一のトークンも最適化できます。これにより、検索頻度の高い索引トークンを最適化できます。この結果、参照頻度の低いトークンの最適化に時間が取られるのを防ぎます。

たとえば、トークン DOG の更新と問合せが頻繁に行われる場合、索引内のこのトークンのみの最適化を指定できます。

トークンを最適化すると、そのトークンの問合せ応答時間が短縮できます。

索引の最適化を単一のトークンで行うには、CTX_DDL.OPTIMIZE_INDEX を使用します。

索引の断片化およびガベージ・データの表示

CTX_REPORT.INDEX_STATS プロシージャを使用すると、索引の統計レポートを作成できます。このレポートには、最適な行の断片化に関する情報、最も断片化されているトークン、および索引内のガベージ・データの量が含まれています。大規模な索引の場合はこのレポートの実行に時間がかかることがありますが、このレポートは索引を最適化する必要があるかどうかの判断に役立ちます。

関連項目： このプロシージャの使用方法は、『Oracle Text リファレンス』を参照してください。

例：索引の最適化

索引を最適化する場合は、CTX_DDL.OPTIMIZE_INDEX を使用することをお勧めします。

関連項目： CTX_DDL.OPTIMIZE_INDEX コマンド構文と使用例の詳細は、『Oracle Text リファレンス』を参照してください。

この章では、Oracle Text の問合せと関連機能を説明します。次の項目について説明します。

- [問合せの概要](#)
- [CONTEXT 文法](#)
- [CTXCAT 文法](#)
- [応答時間短縮のための最適化](#)
- [ヒット数のカウント](#)

問合せの概要

Oracle Text の基本的な問合せでは、問合せ式を入力します。式は、通常ワードで、演算子を併用する場合と併用しない場合があります。式を満たすすべてのドキュメント（事前に索引付け済み）が、各ドキュメントの関連性スコアとともに戻ります。スコアは、結果セット内のドキュメントを順序付けするために使用することができます。

Oracle Text の問合せを発行するには、SQL の SELECT 文を使用します。作成する索引のタイプに応じて、WHERE 句に CONTAINS 演算子または CATSEARCH 演算子のいずれかを使用します。これらの演算子は、PL/SQL カーソル内など、SELECT 文を使用できる状況であれば、いつでもプログラムで使用できます。

CTXRULE 索引を使用してドキュメントを分類するには、MATCHES 演算子を使用します。

CONTAINS による問合せ

CONTEXT 索引タイプを作成する場合は、CONTAINS 演算子を使用して問合せを発行する必要があります。大量のまとまったドキュメントのコレクションを索引付けするには、CONTEXT 索引タイプが最適です。

CONTAINS 演算子では、複数の演算子を使用して検索条件を定義できます。これらの演算子によって、論理、近接、ファジー、ステミング、シソーラスおよびワイルド・カードの各検索を発行できます。また、適切に構成された索引を使用すると、HTML や XML のような内部構造を持つドキュメントに対してセクション検索を発行することもできます。

CONTAINS では、ABOUT 演算子を使用して、ドキュメント・テーマを検索できます。

CONTAINS SQL 例

SELECT 文では、CONTAINS 演算子を使用して WHERE 句で問合せを指定します。また、ヒットリストのヒットごとにスコアを戻すには、SCORE 演算子を指定します。次の例では、問合せの発行方法を示します。

```
SELECT SCORE(1),title from news WHERE CONTAINS(text, 'oracle', 1) > 0;
```

次のように、ORDER BY 句を使用して、結果を最も高いスコアのドキュメントから最も低いスコアのドキュメントに順序付けることができます。

```
SELECT SCORE(1), title from news
      WHERE CONTAINS(text, 'oracle', 1) > 0
      ORDER BY SCORE(1) DESC;
```


CONTAINS PL/SQL 例

PL/SQL アプリケーションでは、カーソルを使用して問合せ結果をフェッチできます。

次の例では、CONTAINS 問合せを NEWS 表に対して発行し、ワード *oracle* を含むすべての記事を検索します。ヒットしたもののうち上位 10 個のタイトルとスコアが出力されます。

```
declare
    rowno number := 0;
begin
    for c1 in (SELECT SCORE(1) score, title FROM news
               WHERE CONTAINS(text, 'oracle', 1) > 0
               ORDER BY SCORE(1) DESC)
    loop
        rowno := rowno + 1;
        dbms_output.put_line(c1.title||': '||c1.score);
        exit when rowno = 10;
    end loop;
end;
```

この例では、カーソル FOR ループを使用して、ヒットしたもののうち上位 10 個を取り出します。SCORE 演算子の戻り値に対して、別名 *score* が宣言されています。カーソル・ドット表記法を使用して、スコアとタイトルが標準出力に出力されます。

CONTAINS による構造化問合せ

構造化問合せは複合問合せとも呼ばれ、テキスト列を問い合わせる CONTAINS 述語と、構造化データ列を問い合わせる別の述語を持つ問合せです。

構造化問合せを発行するには、SELECT 文の WHERE 条件に構造化句を指定します。

たとえば、次の SELECT 文は、1997 年 10 月 1 日以降に書かれた、ワード *oracle* を含む記事をすべて検索します。

```
SELECT SCORE(1), title, issue_date from news
       WHERE CONTAINS(text, 'oracle', 1) > 0
       AND issue_date >= ('01-OCT-97')
       ORDER BY SCORE(1) DESC;
```

注意： CONTAINS で構造化問合せを発行できる場合でも、CTXCAT 索引を作成し、CATSEARCH で問合せを発行することを検討してください。そのほうが構造化問合せのパフォーマンスが向上します。

CATSEARCH による問合せ

CTXCAT 索引タイプを作成する場合は、CATSEARCH 演算子を使用して問合せを発行する必要があります。使用しているアプリケーションで、テキスト列に短いテキスト断片や関連列にその他の関連情報を格納する場合は、CTXCAT 索引タイプが最適です。

たとえば、オンライン・オークション・サイトを提供しているアプリケーションでは、表のテキスト列に品目の説明を格納し、その他の列に日付や価格などの関連情報を格納する場合があります。CTXCAT 索引を使用すると、これらの列の 1 つ以上に B ツリー索引を作成できます。その結果、CATSEARCH 演算子を使用して CTXCAT 索引を検索すると、通常、複合問合せの問合せパフォーマンスが向上します。

CATSEARCH による問合せに使用できる演算子は、AND や OR などの論理操作に制限されています。構造化基準の定義に使用できる演算子は、>、<、=、BETWEEN および IN です。

CATSEARCH SQL 問合せ

CATSEARCH による一般的な問合せの例として、次のように、ワード *camera* を含むすべての行を検索して `bid_close` の日付順にソートする構造化句が組み込まれていることがあります。

```
SELECT FROM auction WHERE CATSEARCH(title, 'camera', 'order by bid_close desc')> 0;
```

発行できる構造化問合せのタイプは、サブ索引の作成方法によって異なります。

関連項目： [第 2 章「索引付け」の「CTXCAT 索引の作成」](#)

CATSEARCH 構造化問合せ

CATSEARCH 問合せの構造化部分を指定するには、`structured_query` パラメータを使用します。構造化式で名前を指定する列には、対応するサブ索引が必要です。

たとえば、`category_id` と `bid_close` には、AUCTION 表の CTXCAT 索引内にサブ索引があるとします。この場合は、次のような構造化問合せを発行できます。

```
SELECT FROM auction WHERE CATSEARCH(title, 'camera', 'category_id=99 order by bid_close desc')> 0;
```

CATSEARCH PL/SQL 例

CONTAINS の場合と同じように、カーソルを使用して、CATSEARCH 問合せの出力を処理できます。

MATCHES による問合せ

CTXRULE 索引タイプを作成する場合は、MATCHES 演算子を使用してドキュメントを分類する必要があります。CTXRULE 索引は、本来は分類を定義する問合せのセットに作成される索引です。

たとえば、ドキュメントの着信ストリームをドキュメントの内容に基づいて分類する必要がある場合は、カテゴリを定義する条件式のセットを作成できます。この条件式は、テキスト列の行として作成します。このタイプの表は、CTX_CLS.TRAIN プロシージャを使用して作成することもできます。

次に、CTXRULE 索引を作成するために表を索引付けします。ドキュメントの着信時に、MATCHES 演算子を使用して各ドキュメントを分類します。

関連項目： CTX_CLS.TRAIN の詳細は、『Oracle Text リファレンス』を参照してください。

MATCHES SQL 問合せ

MATCHES 問合せでは、指定したドキュメントに一致する問合せ表内のすべての行を検索します。表 querytable に CTXRULE 索引が関連付けられているとします。この場合は、次の問合せを発行できます。

```
SELECT classification FROM querytable WHERE MATCHES(text, 'Smith is a common name in the United States') > 0;
```

MATCHES PL/SQL 例

次の例では、問合せ表 profiles に CTXRULE 索引が作成されているとします。また、表 newsfeed には、分類対象の記事のセットが含まれているとします。

この例では、newsfeed 表内をループし、MATCHES 演算子を使用して各記事を分類します。結果は、results 表に格納されます。

```
PROMPT Populate the category table based on newsfeed articles
PROMPT
set serveroutput on
declare
    mypk    number;
    mytitle varchar2(1000);
    myarticles clob;
    mycategory varchar2(100);
    cursor doccur is select pk,title,articles from newsfeed;
    cursor mycur is  select category from profiles where matches(rule, myarticles)>0;
    cursor rescur is select category, pk, title from results order by category,pk;

begin
    dbms_output.enable(1000000);
    open doccur;
```



```
loop
  fetch doccur into mypk, mytitle, myarticles;
  exit when doccur%notfound;
  open mycur;
  loop
    fetch mycur into mycategory;
    exit when mycur%notfound;
    insert into results values(mycategory, mypk, mytitle);
  end loop;
  close mycur;
  commit;
end loop;
close doccur;
commit;

end;
/
```

次の例では、分類された記事をカテゴリ別に表示します。

```
PROMPT display the list of articles for every category
PROMPT
set serveroutput on
declare
  mypk    number;
  mytitle varchar2(1000);
  mycategory varchar2(100);
  cursor catcur is select category from profiles order by category;
  cursor rescur is select pk, title from results where category=mycategory order by
pk;

begin
  dbms_output.enable(1000000);
  open catcur;
  loop
    fetch catcur into mycategory;
    exit when catcur%notfound;
    dbms_output.put_line('***** CATEGORY: '||mycategory||' *****');
  open rescur;
  loop
    fetch rescur into mypk, mytitle;
    exit when rescur%notfound;
    dbms_output.put_line('** ('||mypk||'). '||mytitle);
  end loop;
  close rescur;
  dbms_output.put_line('***');
  dbms_output.put_line('*****');
```



```
end loop;  
close catcur;  
end;  
/
```

ワード問合せと句問合せ

ワード問合せは、ワードまたは句に対する問合せです。たとえば、テキスト表でワード *dog* を含むすべての行を検索するには、問合せ語句として、*dog* を指定して問合せを発行します。

ワード問合せは、SQL 演算子の `CONTAINS` と `CATSEARCH` の両方で発行できます。

問合せ式に複数のワードが空白のみ（演算子なし）で区切られて含まれている場合、そのワードの文字列は句とみなされ、問合せ時に文字列全体が検索されます。

たとえば、句 *international law* を含むすべてのドキュメントを検索するには、句 *international law* を指定して問合せを発行します。

ストップワードの問合せ

ストップワードは、索引エントリが作成されないワードです。ストップワードは、通常、それ自体は検索の対象とならない、その言語の一般的なワードです。

Oracle Text には、使用言語のデフォルトのストップワード・リストが組み込まれています。このリストは、ストップリストと呼ばれます。たとえば、英語では、ワード *this* および *that* は、デフォルトのストップリストでストップワードとして定義されています。このデフォルトのストップリストを変更したり、CTX_DDL パッケージを使用して新しいストップリストを作成できます。また、索引を作成した後で、ALTER INDEX 文でストップワードを追加することもできます。

ストップワードまたはストップワードのみで構成されている句に対する問合せは発行できません。たとえば、*this* がストップワードとして定義されている場合は、ワード *this* を問い合わせても、ヒットは戻りません。

ただし、*this boy talks to that girl* のように、ストップワードとストップワード以外のワードが含まれている句を問い合わせることはできます。これは、Oracle Text の索引では、ストップワードの索引エントリは作成しませんが、ストップワードの位置は記録しているためです。

問合せ句内にストップワードが含まれている場合、このストップワードは任意のワードに一致します。たとえば、次の問合せがあるとします。

```
'Jack was big'
```

ストップワードが *was* の場合、*Jack is big* および *Jack grew big* などの句が一致します。

ABOUT 問合せおよびテーマ

ABOUT 問合せは、ドキュメント・テーマに対する問合せです。ドキュメント・テーマは、テキスト内で詳しく展開されている概念のことです。たとえば、*US politics* の ABOUT 問合せでは、アメリカの大統領選挙や外交政策に関する情報を含むドキュメントが戻る可能性があります。戻るドキュメントには、*US politics* と正確に一致する句が含まれている必要はありません。

索引付け時に、ドキュメント・テーマはナレッジ・ベースから導出されます。このナレッジ・ベースには、一般的な知識を表すカテゴリと概念が階層式にリストされています。たとえば、ナレッジ・カタログのテーマには、*jazz music*、*football* または *Nelson Mandela* などの具体的な概念も含まれています。また、テーマには、*happiness* または *honesty* などの抽象的な概念もあります。

索引付け時に、システムでは、ドキュメント内で詳しく展開されているが、ナレッジ・ベースには存在しないドキュメント・テーマも識別し、索引付けすることができます。

ナレッジ・ベースは、業界または問合せアプリケーション固有の概念や用語を定義して補強できます。補強した場合は、追加した概念に対する ABOUT 問合せの精度が向上します。

索引内にテーマ・コンポーネントを作成すると、ABOUT 問合せのパフォーマンスが最も向上します。テーマ・コンポーネントは、英語とフランス語ではデフォルトで作成されます。

関連項目：『Oracle Text リファレンス』

ストップテーマの問合せ

ABOUT 演算子を使用すると、テーマを問い合わせることができます。ストップテーマは、索引付け対象外のテーマです。ストップテーマは、CTX_DLL パッケージを使用して追加および削除できます。また、索引を作成した後で、ALTER INDEX 文でストップテーマを追加することもできます。

問合せ式

問合せ式とは、CONTAINS 演算子または CATSEARCH 演算子の `text_query` 引数内で一重引用符で囲まれたものを指します。CONTAINS 問合せの問合せ式に組み込むことができる内容は、CATSEARCH 演算子に組み込む内容とは異なります。

CONTAINS 演算子

CONTAINS の問合せ式には、論理検索、近接検索、シソーラスを使用した検索、ファジー検索およびワイルド・カード検索を行う問合せ演算子を含めることができます。ストアド式を使った問合せも可能です。問合せ式内でグループ化文字を使用すると、演算子の優先順位を変更できます。このマニュアルでは、このような演算子を CONTEXT 文法と呼びます。

CONTAINS では、ABOUT 問合せを使用して、ドキュメント・テーマを問い合わせることもできます。

関連項目： この章の「[CONTEXT 文法](#)」を参照してください。

CATSEARCH 演算子

CATSEARCH 演算子では、`text_query` 引数を使用して問合せ式を指定し、`structured_query` 引数を使用してオプションの構造化基準を指定できます。

`text_query` 引数は、ワードと句の問合せに制限されています。AND、OR および NOT などの論理操作を使用できます。このマニュアルでは、このような演算子を CTXCAT 文法と呼びます。

CONTEXT 文法によりサポートされているさらに豊富な演算子を使用する場合は、CATSEARCH で問合せテンプレート機能を使用できます。

`structured_query` 引数では、構造化基準を指定します。次の SQL 操作を使用できます。

- `=`
- `<=`
- `>=`
- `>`
- `<`
- `IN`
- `BETWEEN`

また、ORDER BY 句を使用して出力を順序付けできます。

関連項目： この章の「[CTXCAT 文法](#)」を参照してください。

MATCHES 演算子

MATCHES 演算子は、ドキュメントを入力として受け取り、問合せ表から条件に合うすべての行を検出します。MATCHES 演算子に問合せ式は指定しません。

大 / 小文字を区別した検索

Oracle Text では、ワード問合せと ABOUT 問合せで大 / 小文字区別がサポートされています。

ワード問合せ

ワード問合せでは、デフォルトで大 / 小文字を区別していません。たとえば、語句 *dog* を問い合わせると、テキスト表のワード *dog*、*Dog* または *DOG* を含む行が検出されます。

BASIC_LEXER 索引プリファレンスの `mixed_case` 属性を使用可能にすると、大 / 小文字を区別した検索を使用可能にできます。大 / 小文字を区別する索引の場合は、大 / 小文字を正確に区別して問合せを発行する必要があります。つまり、*Dog* への問合せは、*Dog* を含むドキュメントのみと一致します。*dog* または *DOG* を含むドキュメントは、ヒットとして戻りません。

ストップワードと大 / 小文字区別 ワード問合せに大 / 小文字区別を使用可能に設定して、ストップワードとストップワード以外のワードを含む句に問合せを発行する場合は、そのストップワードの大 / 小文字の区別を正確に指定する必要があります。たとえば、*this boy talks to that girl* という句に問合せを発行したとします。*this* がストップワードの場合、句 *This boy talks to that girl* を含むテキストは戻りません。

ABOUT 問合せ

大 / 小文字区別を適切に行って問合せを実行すると、ABOUT 問合せの精度が最も向上します。これは、問合せの正規化が、大 / 小文字が区別されるナレッジ・カタログに基づいているためです。大文字 / 小文字によって意味が異なるワードの場合は、特に注意が必要です。たとえば、*turkey* は鳥の名で、*Turkey* は国名です。

ただし、ABOUT 問合せの関連結果を取得するために、大 / 小文字を正確に区別して問合せを入力する必要はありません。システムが、最適な方法で問合せを解釈します。たとえば、ORACLE という問合せを入力し、この概念がナレッジ・カタログで検索されない場合、システムは検索用の関連概念として Oracle を使用する場合があります。

問合せのフィードバック

フィードバック情報には、CONTEXT 索引で指定した問合せに対する上位語、下位語および関連語の情報が含まれています。フィードバック情報は、CTX_QUERY.HFEEDBACK プロシージャを使用してプログラムで取得します。

上位語、下位語および関連語の情報は、他の問合せ語句を問合せアプリケーションのユーザーに提示する場合に役立ちます。

戻されるフィードバック情報はナレッジ・ベースから取得され、索引中にも存在する語句のみを含みます。これによって、HFEEDBACK プロシージャから戻された語句が、現在索引付けされているドキュメント・セットに対して高いヒットを生成する可能性が高くなります。

関連項目： CTX_QUERY.HFEEDBACK の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

問合せの実行計画

実行計画情報では、CONTAINS 問合せ式の解析ツリーがグラフィカルに表示されます。実行計画情報は、CTX_QUERY.EXPLAIN プロシージャを使用してプログラムで取得できます。

実行計画情報を使用すると、問合せを実行しなくても、問合せの拡張方法や解析方法がわかります。実行計画情報を取得すると、STEM、ワイルド・カード、シソーラス、FUZZY、SOUNDEX、ABOUT などの特定の問合せの拡張方法がわかります。解析ツリーには、次の情報も表示されます。

- 実行の順序
- ABOUT 問合せの正規化
- 問合せ式の最適化
- ストップワード変換
- サポート対象言語の複合語トークンの分類

関連項目： CTX_QUERY.EXPLAIN の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

CONTEXT 文法

CONTEXT 文法は、CONTAINS のデフォルトの文法です。この文法では、演算子を使用してさらに複雑な検索を行うことができます。問合せ演算子は問合せ式で使用します。たとえば、論理演算子 AND を使用すると、2つの異なるワードを含むすべてのドキュメントを検索できます。ABOUT 演算子を使用すると、概念を検索できます。

また、セクション検索には WITHIN 演算子を、近接検索には NEAR 演算子を使用できます。さらに、問合せ式の拡張には、STEM 演算子、FUZZY 演算子およびシソーラス演算子を使用できます。

CONTAINS では、問合せテンプレート機能を使用して CTXCAT 文法を使用できます。

次の各項では、Oracle Text の演算子について説明します。

関連項目： 問合せ演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

ABOUT 問合せ

英語またはフランス語での概念の問合せには、ABOUT 演算子を使用します。通常、問合せ文字列は、検索する内容を表す概念またはテーマになります。Oracle では、そのテーマを含むドキュメントを戻します。

ワード情報とテーマ情報は、単一の索引に結合されます。テーマ問合せを発行するには、索引内にテーマ・コンポーネントが存在している必要があります。英語とフランス語の場合は、デフォルトで作成されます。

テーマ問合せは、問合せ式内で ABOUT 演算子を使用して発行します。たとえば、*politics* に関するすべてのドキュメントを取り出すには、次のように問合せを記述します。

```
SELECT SCORE(1), title FROM news
      WHERE CONTAINS(text, 'about(politics)', 1) > 0
      ORDER BY SCORE(1) DESC;
```

関連項目： ABOUT 演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

論理演算子

AND や OR などの論理演算子を使用すると、様々な方法で検索条件を制限できます。次の表では、論理演算子について説明します。

演算子	記号	説明	式の例
AND	&	各問合せ語句が 1 つ以上出現するドキュメントを検索するには、AND 演算子を使用します。 オペランドの最小値がスコアとして戻ります。	'cats AND dogs' 'cats & dogs'
OR		問合せ語句のいずれかが 1 つ以上出現するドキュメントを検索するには、OR 演算子を使用します。 オペランドの最大値がスコアとして戻ります。	'cats dogs' 'cats OR dogs'
NOT	~	ある問合せ語句が含まれているが、別の問合せ語句は含まれていないドキュメントを検索するには、NOT 演算子を使用します。	たとえば、 <i>animals</i> という語句が含まれていて、 <i>dogs</i> という語句は含まれていないドキュメントを取得するには、次の式を使用します。 'animals ~ dogs'

演算子	記号	説明	式の例
ACCUM	,	問合せ語句のいずれかが 1 つ以上出現するドキュメントを検索するには、ACCUM 演算子を使用します。ACCUM 演算子は、ドキュメントを語句の合計の重みによってランク付けします。	次の問合せは、 <i>dogs</i> 、 <i>cats</i> および <i>puppies</i> という語句を含むすべてのドキュメントを検出し、3 つの語句をすべて含むドキュメントに最も高いスコアを割り当てます。 'dogs, cats, puppies'
EQUIV	=	問合せ時に条件を満たす代替ワードを指定するには、EQUIV 演算子を使用します。	次の例は、句 <i>alsatians are big dogs</i> または <i>German shepherds are big dogs</i> のいずれかを含むすべてのドキュメントを検出します。 'German shepherds=alsatians are big dogs'

セクション検索

セクション検索は、ドキュメント・セットが HTML または XML の場合に有効です。HTML の場合は、埋込みタグを使用してセクションを定義し、次に WITHIN 演算子を使用して定義したセクションを検索します。

XML の場合は、システムでセクションを自動的に作成するように設定できます。パスを検索するには、WITHIN 演算子または INPATH 演算子で問い合わせることができます。

関連項目： [第 6 章「ドキュメントのセクション検索」](#)

NEAR 演算子による近接問合せ

NEAR 演算子を使用すると、ドキュメント内で互いに近接している語句を検索できます。

たとえば、*dog* が *cat* の 6 ワード以内にあるすべてのドキュメントを検索する場合は、次の問合せを発行します。

```
'near((dog, cat), 6)'
```

関連項目： NEAR 演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

FUZZY、STEM、SOUNDEX、ワイルド・カードおよびシソーラスの拡張演算子

ワイルド・カード、FUZZY、STEM、SOUNDEX、シソーラスなどの演算子を使用すると、問合せを長いワード・リストに拡張できます。

関連項目： これらの演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

[第5章「パフォーマンス・チューニング」の「問合せに多数の拡張を使用してもかまいませんか？」](#)

CTXCAT 文法の使用法

CONTAINS 問合せに CTXCAT 文法を使用できます。そのためには、CONTAINS の `text_query` パラメータに問合せテンプレート指定を使用します。

代替のより単純な問合せ文法が必要な場合は、CTXCAT 文法を利用できます。

関連項目： これらの演算子の使用法は、『Oracle Text リファレンス』を参照してください。

ストアド・クエリー式

CTX_QUERY.STORE_SQE プロシージャを使用すると、結果を格納せずに問合せの定義を格納できます。CONTAINS の SQE 演算子によって、その問合せの定義を参照します。このように、ストアド・クエリー式を使用すると、長い問合せ式や使用頻度の高い問合せ式を簡単に定義できます。

ストアド・クエリー式は索引に連結されません。CTX_QUERY.STORE_SQE をコールする場合、指定するのはストアド・クエリー式の名前と問合せ式のみです。

問合せの定義は、テキスト・データ・ディクショナリに格納されます。ストアド・クエリー式は、すべてのユーザーが参照できます。

関連項目： CTX_QUERY.STORE_SQE の構文の詳細は、『Oracle Text リファレンス』を参照してください。

ストアド・クエリー式の定義

ストアド・クエリー式は、次のように定義して使用します。

1. CTX_QUERY.STORE_SQE をコールして、テキスト列に対する問合せを格納します。STORE_SQE を使用して、ストアド・クエリー式の名前と問合せ式を指定します。
2. SQE 演算子を使用して、問合せ式でストアド・クエリー式をコールします。ストアド・クエリー式の結果は、通常の問合せの結果と同じ方法で戻ります。問合せは、ストアド・クエリー式のコール時に評価されます。

ストアド・クエリー式は、REMOVE_SQE を使用して削除できます。

SQE 例

次の例では、*disaster* というストアド・クエリー式を作成します。この式は、*tornado*、*hurricane* または *earthquake* のいずれかのワードを含むドキュメントを検索します。

```
begin
ctx_query.store_sqe('disaster', 'tornado | hurricane | earthquake');
end;
```

この問合せを 1 つの式で実行するには、次のように問合せを記述します。

```
SELECT SCORE(1), title from news
WHERE CONTAINS(text, 'SQE(disaster)', 1) > 0
ORDER BY SCORE(1);
```

関連項目： CTX_QUERY.STORE_SQE の構文の詳細は、『Oracle Text リファレンス』を参照してください。

CONTAINS での PL/SQL ファンクションのコール

ユーザー定義ファンクションを直接 CONTAINS 句でコールできます。ただし、そのユーザー定義ファンクションが SQL 文で指定された要件を満たしている必要があります。また、コールを実行するユーザーは、そのファンクションに対する EXECUTE 権限を所有している必要があります。

たとえば、ファンクション *french* が英語のワードに相当するフランス語を戻すとします。次の問合せを記述すると、*cat* に相当するフランス語のワードを検索できます。

```
SELECT SCORE(1), title from news
WHERE CONTAINS(text, french('cat'), 1) > 0
ORDER BY SCORE(1);
```

関連項目： ユーザー・ファンクションの作成および SQL からのユーザー・ファンクションのコールの詳細は、『Oracle9i SQL リファレンス』を参照してください。

CTXCAT 文法

CTXCAT 文法は、CATSEARCH のデフォルトの文法です。この文法は、句問合せの他に、AND や OR などの論理操作もサポートしています。

CATSEARCH の問合せ演算子は、次の構文です。

操作	構文	操作の説明
論理 AND	a b c	a、b および c を含む行を戻します。
論理 OR ()	a b c	a、b または c を含む行を戻します。
論理 NOT (-)	a - b	a を含み、b を含まない行を戻します。
空白なしのハイフン	a-b	通常の文字として処理されるハイフンです。 たとえば、ハイフンが skipjoin として定義されている場合、web-site などのワードは、単一の問合せ語句 website として処理されます。 同様に、ハイフンが printjoin として定義されている場合、web-site などのワードは、CTXCAT 問合せ言語では web-site として処理されます。
" "	"a b c"	句 "a b c" を含む行を戻します。 たとえば、"Sony CD Player" と入力すると、この一連のワードを含むすべての行を戻します。
()	(A B) C	グループ設定をカッコで囲みます。この問合せは、CONTAINS 問合せの (A &B) C に相当します。

CATSEARCH での CONTEXT 文法の使用

さらに、CATSEARCH 問合せで CONTEXT 文法を使用することもできます。そのためには、text_query パラメータに問合せテンプレート指定を使用します。

CTXCAT 索引で近接問合せ、シソーラス問合せまたは ABOUT 問合せを発行する必要がある場合に、CONTAINS 文法を使用することがあります。

関連項目： これらの演算子の使用方法は、『Oracle Text リファレンス』を参照してください。

応答時間短縮のための最適化

CONTAINS 問合せを応答時間短縮のために最適化すると、ヒットリストの最もスコアの高いドキュメントが必要な場合に、高速なソリューションとなります。

次の例では、標準出力に最初の 20 個のヒットを戻します。この例では、FIRST_ROWS (n) ヒントとカーソルを使用します。

```
declare
cursor c is
  select /*+ FIRST_ROWS(20) */ title, score(1) score
    from news where contains(txt_col, 'dog', 1) > 0 order by score(1) desc;
begin
  for c1 in c
  loop
    dbms_output.put_line(c1.score||':'||substr(c1.title,1,50));
    exit when c%rowcount = 21;
  end loop;
end;
/
```

関連項目： [第 5 章「パフォーマンス・チューニング」の「応答時間短縮のための問合せの最適化」](#)

問合せの応答時間に影響するその他の要因

問合せヒントの使用以外にも、次のような要因が問合せ応答時間に影響します。

- 表統計の収集
- メモリー割当て
- ソート
- 元表に存在する LOB 列
- パーティション化
- 並列性
- 問合せで語句を拡張する回数

関連項目： [第 5 章「パフォーマンス・チューニング」の「問合せのパフォーマンスに関する FAQ（よくある質問）」](#)

ヒット数のカウント

CONTAINS 述語のみが指定された問合せから戻されたヒット数をカウントするには、PL/SQL の CTX_QUERY.COUNT_HITS または SQL SELECT 文の COUNT(*) を使用できます。

おおまかなヒット件数が必要な場合は、CTX_QUERY.COUNT_HITS を予測モード (EXACT パラメータを FALSE に設定) で使用できます。応答時間の点では、これが最も高速なカウント方法です。

構造化述語が含まれた問合せから戻されるヒット数をカウントするには、SELECT 文の COUNT(*) ファンクションを使用します。

SQL によるヒット数のカウント例

ワード *oracle* を含むドキュメント数を検索するには、SQL の COUNT ファンクションによる問合せを次のように発行します。

```
SELECT count(*) FROM news WHERE CONTAINS(text, 'oracle', 1) > 0;
```

構造化述語によるヒット数のカウント

構造化述語による問合せで戻されるドキュメント数を検索するには、COUNT(*) を次のように使用します。

```
SELECT COUNT(*) FROM news WHERE CONTAINS(text, 'oracle', 1) > 0 and author = 'jones';
```

PL/SQL によるヒット数のカウント例

ワード *oracle* を含むドキュメント数を検索するには、COUNT_HITS を次のように使用します。

```
declare count number;
begin
    count := ctx_query.count_hits(index_name => my_index, text_query => 'oracle',
                                   exact => TRUE);
    dbms_output.put_line('Number of docs with oracle:');
    dbms_output.put_line(count);
end;
```

関連項目： CTX_QUERY.COUNT_HITS の構文の詳細は、『Oracle Text リファレンス』を参照してください。

ドキュメントの表示方法

この章では、ドキュメントの表示方法を説明します。次の項目について説明します。

- [問合せ語句のハイライト表示](#)
- [テーマのリスト、要旨およびテーマ・サマリーの取得](#)

問合せ語句のハイライト表示

Oracle Text の問合せアプリケーションでは、テキスト問合せの場合は問合せ語句をハイライト表示し、ABOUT 問合せの場合はテーマをハイライト表示して、選択したドキュメントを表示できます。

ハイライト表示に関連する出力では、マークアップ形式のドキュメント、プレーン・テキスト形式のドキュメント（フィルタ処理済みの出力）およびドキュメントのハイライト・オフセット情報の 3 タイプを生成できます。

この 3 タイプの出力は、PL/SQL パッケージ CTX_DOC（ドキュメント・サービス）に含まれている 3 つの異なるプロシージャによって生成されます。また、各出力タイプごとにプレーン・テキスト形式と HTML 形式の出力を取得できます。

テキストのハイライト表示

テキストのハイライト表示の場合は、問合せを発行すると、その問合せを満たすドキュメント内のワードがハイライト表示されます。プレーン・テキストまたは HTML もハイライト表示できます。

テーマのハイライト表示

ABOUT 問合せの場合、CTX_DOC パッケージのプロシージャによって、ABOUT 問合せを最も適切に表しているワードまたは句がハイライト表示またはマークアップされます。

CTX_DOC のハイライト表示プロシージャ

CTX_DOC には、次の 3 つのハイライト表示プロシージャがあります。

- CTX_DOC.HIGHLIGHT
- CTX_DOC.MARKUP
- CTX_DOC.FILTER

HIGHLIGHT プロシージャ

ハイライト・オフセット情報は、ドキュメント表示用の独自のカスタム・ルーチンを記述する場合に有効です。

ハイライト・オフセット情報を取得するには、CTX_DOC.HIGHLIGHT プロシージャを使用します。このプロシージャは、問合せとドキュメントを使用して、プレーン・テキスト形式または HTML 形式のいずれかのハイライト・オフセット情報を戻します。

オフセット情報を使用すると、ドキュメントを任意のハイライト形式で表示できます。たとえば、CTX_DOC.MARKUP から取得する標準のプレーン・テキストのマークアップではなく、様々なフォント・タイプまたは色で、ドキュメントを表示できます。

関連項目： CTX_DOC.HIGHLIGHT の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

MARKUP プロシージャ

CTX_DOC.MARKUP プロシージャは、ドキュメント参照と問合せを使用して、マークアップされたドキュメントを戻します。出力は、マークアップされたプレーン・テキストかマークアップされた HTML のいずれかになります。

HTML ナビゲーションのマークアップ順序をカスタマイズできます。

関連項目： CTX_DOC.MARKUP の詳細は、『Oracle Text リファレンス』を参照してください。

FILTER プロシージャ

ドキュメントが Microsoft Word などの固有の形式で格納されている場合は、FILTER プロシージャ CTX_DOC.FILTER を使用して、プレーン・テキスト形式または HTML 形式のドキュメントのいずれかを取得できます。

関連項目： CTX_DOC.FILTER の詳細は、『Oracle Text リファレンス』を参照してください。

テーマのリスト、要旨およびテーマ・サマリーの取得

次の表では、テーマのリスト、要旨およびテーマ・サマリーについて説明します。

表 4-1

出力タイプ	説明
テーマのリスト	ドキュメントの主要概念のリスト。 リストの各テーマが単一のワードまたは句である場合、あるいはリストの各テーマが親テーマの階層リストである場合は、テーマのリストを生成できます。
要旨	ドキュメントの概要を最も適切に表すドキュメント内のテキスト。
テーマ・サマリー	ドキュメントの特定のテーマを最も適切に表すドキュメント内のテキスト。

この出力を取得するには、提供されている CTX_DOC パッケージのプロシージャを使用します。このパッケージを使用すると、次の操作を実行できます。

- 主キーに加えて ROWID でもドキュメントを識別します。
- パフォーマンス向上のため、結果をメモリー内に格納します。

テーマのリスト

テーマのリストは、ドキュメントの主要概念のリストです。テーマのリストを生成するには、CTX_DOC.THEMES プロシージャを使用します。

関連項目： CTX_DOC.THEMES のコマンド構文の詳細は、『Oracle Text リファレンス』を参照してください。

メモリー内テーマ

次の例では、ドキュメント 1 のトップ 10 のテーマを生成し、それを the_themes というメモリー内表に格納します。その後、表内をループし、ドキュメントのテーマを表示します。

```
declare
  the_themes ctx_doc.theme_tab;

begin
  ctx_doc.themes('myindex','1',the_themes, numthemes=>10);
  for i in 1..the_themes.count loop
    dbms_output.put_line(the_themes(i).theme||':'||the_themes(i).weight);
  end loop;
end;
```

結果表のテーマ

テーマ表は、次のように作成します。

```
create table ctx_themes (query_id number,
                        theme varchar2(2000),
                        weight number);
```

単一テーマ リストの各要素が単一テーマであるテーマ・リストを取得するには、次の文を発行します。

```
begin
  ctx_doc.themes('newsindex','34','CTX_THEMES',1,full_themes => FALSE);
end;
```

全テーマ リストの各要素が親テーマの階層リストであるテーマのリストを取得するには、次の文を発行します。

```
begin
  ctx_doc.themes('newsindex','34','CTX_THEMES',1,full_themes => TRUE);
end;
```


要旨およびテーマ・サマリー

要旨は、ドキュメントの概要を最も適切に表すドキュメント内のテキストです。テーマ・サマリーは、ドキュメントの単一テーマを最も適切に表すドキュメント内のテキストです。

要旨およびテーマ・サマリーを生成するには、CTX_DOC.GIST プロシージャを使用します。プロシージャのコール時に、要旨またはテーマ・サマリーのサイズを指定できます。

関連項目： CTX_DOC.GIST のコマンド構文の詳細は、『Oracle Text リファレンス』を参照してください。

メモリー内要旨

次の例では、10 個以内の段落のデフォルト・サイズでない全体の要旨を生成します。結果は CLOB ロケータの中のメモリーに格納されます。次のコードでは、戻された CLOB ロケータの割当てを使用後に解除しています。

```
declare
  gklob clob;
  amt number := 40;
  line varchar2(80);

begin
  ctx_doc.gist('newsindex','34','gklob',1,glevel => 'P',pov => 'GENERIC',
numParagraphs => 10);
  -- gklob is NULL when passed-in, so ctx-doc.gist will allocate a temporary
  -- CLOB for us and place the results there.

  dbms_lob.read(gklob, amt, 1, line);
  dbms_output.put_line('FIRST 40 CHARS ARE:'||line);
  -- have to de-allocate the temp lob
  dbms_lob.freetemporary(gklob);
end;
```

結果表の要旨

要旨表は、次のように作成します。

```
create table ctx_gist (query_id number,
                      pov      varchar2(80),
                      gist      CLOB);
```


次の例では、ドキュメント 34 に対して、デフォルト・サイズの段落レベルの要旨を戻します。

```
begin
ctx_doc.gist('newsindex','34','CTX_GIST',1,'PARAGRAPH', pov =>'GENERIC');
end;
```

次の例では、10 個の段落のデフォルト・サイズでない要旨を生成します。

```
begin
ctx_doc.gist('newsindex','34','CTX_GIST',1,'PARAGRAPH', pov =>'GENERIC',
numParagraphs => 10);
end;
```

次の例では、段落数がドキュメントの合計段落数の 10 パーセントの要旨を生成します。

```
begin
ctx_doc.gist('newsindex','34','CTX_GIST',1, 'PARAGRAPH', pov =>'GENERIC', maxPercent
=> 10);
end;
```

テーマ・サマリー

次の例では、テキストキー 34 を持つドキュメントのテーマ *insects* に対するテーマ・サマリーを戻します。デフォルト・サイズの要旨が戻ります。

```
begin
ctx_doc.gist('newsindex','34','CTX_GIST',1, 'PARAGRAPH', pov => 'insects');
end;
```

パフォーマンス・チューニング

この章では、問合せおよび索引付けのパフォーマンスを改善する方法について説明します。次の項目について説明します。

- [統計を使用した問合せの最適化](#)
- [応答時間短縮のための問合せの最適化](#)
- [スループット向上のための問合せの最適化](#)
- [パラレル問合せ](#)
- [ブロック操作による問合せのチューニング](#)
- [問合せのパフォーマンスに関する FAQ（よくある質問）](#)
- [索引付けのパフォーマンスに関する FAQ（よくある質問）](#)
- [索引の更新に関する FAQ（よくある質問）](#)

統計を使用した問合せの最適化

統計を使用して問合せを最適化する場合は、問合せ表および索引に関して収集された統計を使用して、その問合せを最も効率的に処理できる実行計画が選択されます。一般に、問合せパフォーマンスの改善が必要な場合は、元表に関する統計を収集することをお勧めします。

オブティマイザは、次のパラメータに基づいて最適な実行計画を選択します。

- CONTAINS 述語の選択性
- 同じ問合せに含まれるその他の述語の選択性
- CONTAINS 述語を処理したときの CPU コストと I/O コスト

次の各項では、統計を拡張可能問合せオブティマイザで使用方法を説明します。統計を使用して最適化を行うと、CONTAINS 述語の選択性およびコストをより正確に見積ることができるため、より適切な実行計画が選択されます。

統計の収集

デフォルトでは、コストベースのオブティマイザによって問合せに対する最適な実行計画が決定されます。オブティマイザを使用して最適なコストを見積るために、問い合わせる表の統計を計算できます。そのためには、次の文を発行します。

```
ANALYZE TABLE <table_name> COMPUTE STATISTICS;
```

また、次のように表のサンプルの統計を見積ることができます。

```
ANALYZE TABLE <table_name> ESTIMATE STATISTICS 1000 ROWS;
```

または

```
ANALYZE TABLE <table_name> ESTIMATE STATISTICS 50 PERCENT;
```

統計の収集は、DBMS_STATS.GATHER_TABLE_STATS プロシージャを使用してパラレルに行うこともできます。

```
begin
    DBMS_STATS.GATHER_TABLE_STATS('owner', 'table_name',
                                   estimate_percent=>50,
                                   block_sample=>TRUE,
                                   degree=>4) ;
end ;
```

これらの文は、table_name に対応付けられたすべてのオブジェクトの統計を収集します。オブジェクトには、その表の列とその表に対応付けられたすべての索引（B ツリー、ビットマップまたはテキスト・ドメイン）も含まれます。

表の統計を再収集するには、ANALYZE コマンドを必要な回数発行するか、DBMS_STATS パッケージを使用します。

テキスト・ドメイン索引の統計を収集することによって、コストベースのオプティマイザでは次の見積りを行うことができます。

- CONTAINS 述語の選択性の見積り
- テキスト索引を使用したときの I/O コストと CPU コスト（ドメイン索引を使用して CONTAINS 述語を処理するときのコスト）の見積り
- CONTAINS の起動ごとの I/O コストと CPU コストの見積り

CONTAINS 述語の選択性が分かっていると、構造化問合せなど、複数の述語が含まれた問合せを行う場合に役立ちます。このように、コストベースのオプティマイザでは、ドメイン索引を使用して CONTAINS を評価するか、あるいは CONTAINS 述語をポスト・フィルタとして適用するかを、より適切に判断できます。

関連項目：

ANALYZE コマンドの詳細は、『Oracle9i SQL リファレンス』および『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

DBMS_STATS パッケージの詳細は、『Oracle9i 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』を参照してください。

例

次の構造化問合せについて考えます。

```
select score(1) from tab where contains(txt, 'freedom', 1) > 0 and author = 'King'
and year > 1960;
```

author 列は VARCHAR2 型で、year 列は NUMBER 型であるとします。また、author 列には B ツリー索引があるとします。

さらに、構造化された author 述語は、CONTAINS 述語および year 述語よりも選択性が高いと想定します。つまり、構造化述語 (author = 'King') は、year 述語および CONTAINS 述語と比較して、かなり少ない数の行（たとえば、それぞれ 1000 行および 1500 行に対して 5 行）を戻すと想定します。

このような状況では、最初に構造化述語 (author = 'King') に B ツリー索引のレンジ・スキャンを行い、次に ROWID で表アクセスした後、B ツリーの表アクセスから戻された行に他の 2 つの述語を適用することで、この問合せをより効率的に行うことができます。

注意： テキスト索引の統計が収集されていない場合、コストベース・オプティマイザは、CONTAINS 述語の選択性と索引コストが低いものとみなします。

統計の再収集

索引を同期化した後、単一の索引の統計を再収集して、コストの見積りを更新できます。

同期化の前に元表が再解析されている場合は、同期化の後に表全体を再解析しなくても、索引を解析するのみで十分です。

そのためには、次のいずれかの文を発行します。

```
ANALYZE INDEX <index_name> COMPUTE STATISTICS;
```

または

```
ANALYZE INDEX <index_name> ESTIMATE STATISTICS SAMPLE 1000 ROWS;
```

または

```
ANALYZE INDEX <index_name> ESTIMATE STATISTICS SAMPLE 50 PERCENT;
```

統計の削除

表に対応付けられた統計を削除するには、次の文を発行します。

```
ANALYZE TABLE <table_name> DELETE STATISTICS;
```

1 つの索引の統計を削除するには、次の文を発行します。

```
ANALYZE INDEX <index_name> DELETE STATISTICS;
```

応答時間短縮のための問合せの最適化

デフォルトでは、問合せはスループット向上のために最適化されます。最適化の結果、問合せは最短時間ですべての行を戻します。

ただし、多くの場合（特に Web アプリケーションの場合）、問合せを応答時間が短縮されるように最適化する必要があります。これは大きなヒットリストからできるだけ短い時間で、最初の数個のヒットのみを取得する必要があるためです。

次の各項では、応答時間が短縮されるように CONTAINS 問合せを最適化する方法を説明します。

- [FIRST_ROWS\(n\)](#) による [ORDER BY](#) 問合せの応答時間の短縮
- [ローカル・パーティション CONTEXT](#) 索引を使用した応答時間の短縮
- [スコア順のローカル・パーティション索引](#)を使用した応答時間の短縮

問合せの応答時間に影響するその他の要因

次のような要因が問合せ応答時間に影響します。

- 表統計の収集
- メモリー割当て
- ソート
- 元表に存在する LOB 列
- パーティション化
- 並列性
- 問合せで語句を拡張する回数

関連項目： この章の「[問合せのパフォーマンスに関する FAQ（よくある質問）](#)」を参照してください。

FIRST_ROWS(n) による ORDER BY 問合せの応答時間の短縮

FIRST_ROWS (n) ヒントはリリース 9.0.1 の新機能です。ORDER BY 問合せの最初のいくつかの行が必要な場合は、FIRST_ROWS のかわりに、完全にコストベースであるこの新しいヒントを使用することをお勧めします。

注意： このヒントはコストベースであるため、使用する前に表に関する統計を収集しておくことをお勧めします。この章の「[統計の収集](#)」を参照してください。

FIRST_ROWS (n) は、最初の n 行をできるだけ速く受け取る必要がある場合に使用します。たとえば、次の PL/SQL ブロックでは、カーソルを使用して問合せの最初の 10 ヒットを取り出し、FIRST_ROWS (n) ヒントを使用して応答時間が短縮されるように最適化しています。


```
declare
cursor c is

select /* FIRST_ROWS(10) */ article_id from articles_tab
      where contains(article, 'Oracle')>0 order by pub_date desc;

begin
  for i in c
  loop
    insert into t_s values(i.pk, i.col);
    exit when c%rowcount > 11;
  end loop;
end;
/
```

カーソル `c` は `test` というワードを含むソートされた ROWID を戻す SELECT 文です。コードがカーソル内をループして、最初の 10 行を取り出します。取り出された行は、一時表 `t_s` に格納されます。

`FIRST_ROWS (n)` ヒントを指定した場合、`Oracle` はテキスト索引に対して（可能な場合）ROWID をスコア順で戻すように指示します。

このヒントがない場合は、テキスト索引が `CONTAINS` 述語を満たすすべての行をソートされていない状態で戻した後で、`Oracle` が ROWID をソートします。このように結果セット全体を取り出すと時間がかかります。

この問合せでは、最初の 10 ヒットのみが必要なため、ヒントを使用することでパフォーマンスが向上します。

注意： `FIRST_ROWS (n)` ヒントは、問合せで最初の数個のヒットのみが必要な場合に使用します。結果セット全体が必要な場合は、パフォーマンスの低下につながるため、このヒントは使用しないでください。

FIRST_ROWS ヒントについて

また、`FIRST_ROWS` ヒントを使用して応答時間を短縮することもできます。

`FIRST_ROWS (n)` の場合と同じように、応答時間が短縮されるように問合せが最適化されている場合は、`Oracle` は最初の数行を最短時間で戻します。

たとえば、このヒントは次のように使用できます。

```
select /*+ FIRST_ROWS */ pk, score(1), col from ctx_tab
      where contains(txt_col, 'test', 1) > 0 order by score(1) desc;
```


ただし、このヒントはルールベースのみです。つまり、Oracle では、ORDER BY 句を満たす索引を常に選択します。この結果、CONTAINS 句の選択性が非常に高い問合せでは、パフォーマンスが最善にはならないことがあります。そのような場合は、完全にコストベースの FIRST_ROWS (n) ヒントを使用することをお勧めします。

ローカル・パーティション CONTEXT 索引を使用した応答時間の短縮

データをパーティション化し、ローカル・パーティション索引を作成すると、問合せのパフォーマンスが向上します。パーティション表では、各パーティションに独自の索引表セットがあります。実際には複数の索引がありますが、各索引からの結果が必要に応じて組み合わせられ、最終的な結果セットが生成されます。

LOCAL キーワードを使用して、CONTEXT 索引を次のように作成します。

```
CREATE INDEX index_name ON table_name (column_name)
INDEXTYPE IS ctxsys.context LOCAL
PARAMETERS ('...');
```

パーティション化された表および索引を使用すると、次のような問合せのパフォーマンスが向上します。

- パーティション・キー列に対する範囲検索
- パーティション・キー列での ORDER BY

パーティション・キー列に対する範囲検索

これは、パーティション・キーでもある列の特定の値範囲に検索を限定する問合せです。たとえば、日付範囲に対する次の問合せについて考えます。

```
SELECT storyid FROM storytab WHERE CONTAINS(story, 'oliver')>0 and pub_date BETWEEN
'1-OCT-93' AND '1-NOV-93';
```

日付範囲が制限されている場合は、1 つのパーティションを検索するだけで問合せが満たされる可能性があります。

パーティション・キー列での ORDER BY

これは、最初の N 個のヒットのみが必要で、ORDER BY 句がパーティション・キーを指定する問合せです。次のように、price 列に対する ORDER BY 問合せで最初の 20 ヒットをフェッチする場合を考えてみます。

```
SELECT * FROM (
    SELECT itemid FROM item_tab WHERE CONTAINS(item_desc, 'cd player')>0 ORDER BY
    price)
WHERE ROWNUM < 20;
```


この例では、表は `price` によりパーティション化されており、最初のパーティションからのヒットを取得すれば問合せが満たされる可能性があります。

スコア順のローカル・パーティション索引を使用した応答時間の短縮

ローカル・パーティション索引に対して `FIRST_ROWS` ヒントを使用すると、特にスコア順にした場合にパフォーマンスが大幅に低下することがあります。これは、結果をソートする前に、全パーティションにまたがる問合せの全ヒットを取得する必要があるためです。

パフォーマンスの低下を回避するには、`FIRST_ROWS` ヒントを使用するときにインライン・ビューを使用します。`FIRST_ROWS` ヒントを使用すると、次のような条件下で、ローカル・パーティション表の問合せパフォーマンスを向上させることができます。

- `SCORE()` 句による順序を含むテキスト問合せ自体がインライン・ビューとして表されている場合
- インライン・ビュー内のテキスト問合せに `FIRST_ROWS` ヒントまたは `DOMAIN_INDEX_SORT` ヒントが含まれている場合
- インライン・ビューに対する問合せに、ビューからフェッチする行数を制限する `ROWNUM` 述語が含まれている場合

たとえば、次のテキスト問合せがあり、パーティション表 `doc_tab` に対してローカル・テキスト索引が作成されているとします。

```
select doc_id, score(1) from doc_tab
       where contains(doc, 'oracle', 1)>0
       order by score(1) desc;
```

ここで、上位 20 行のみを取り出す場合は、この問合せを次のように再作成します。

```
select * from
  (select /*+ FIRST_ROWS */ doc_id, score(1) from doc_tab
   where contains(doc, 'oracle', 1)>0 order by score(1) desc)
where rownum < 21;
```

関連項目： 問合せオブティマイザの詳細および `FIRST_ROWS` などのヒントの使用方法は、『*Oracle9i データベース・パフォーマンス・チューニング・ガイド*および*リファレンス*』を参照してください。

`EXPLAIN PLAN` コマンドの詳細は、『*Oracle9i データベース・パフォーマンス・チューニング・ガイド*および*リファレンス*』および『*Oracle9i SQL リファレンス*』を参照してください。

スループット向上のための問合せの最適化

問合せがスループット向上のために最適化された場合は、すべてのヒットが最短時間で戻されます。これはデフォルトの動作です。

次の各項では、スループット向上のために明示的に最適化する方法を説明します。

CHOOSE および ALL ROWS モード

デフォルトでは、問合せは CHOOSE モードおよび ALL_ROWS モードでスループットが向上するように最適化されます。問合せがスループット向上のために最適化された場合は、すべての行が最短時間で戻されます。

FIRST_ROWS モード

FIRST_ROWS モードでは、可能な場合、Oracle オプティマイザは、テキスト・ドメイン索引にスコア順にソートされた行を戻させることにより、応答時間を短縮するように最適化を行います。これは、FIRST_ROWS ヒントを使用する場合のデフォルトの動作です。

FIRST_ROWS でスループットがさらに向上するように最適化するには、DOMAIN_INDEX_NO_SORT ヒントを使用します。スループットの向上とは、すべての行を最短時間で問合せに取り込むことです。

次の例では、スコア順にソートした行を戻すためにテキスト・ドメイン索引を使用せずにスループットを向上させています。かわりに、CONTAINS 述語を満たすすべての行が索引から取り出された後で、Oracle によって行がソートされます。

```
select /*+ FIRST_ROWS DOMAIN_INDEX_NO_SORT */ pk, score(1), col from ctx_tab
       where contains(txt_col, 'test', 1) > 0 order by score(1) desc;
```

関連項目： 問合せオプティマイザの詳細および FIRST_ROWS や CHOOSE などのヒントの使用方法は、『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

パラレル問合せ

Oracle では、ローカルな CONTEXT 索引に対するパラレル問合せがサポートされています。つまり、Oracle では、索引の並列度と様々なシステム属性に基づいて、索引処理のために起動されるパラレル問合せスレーブの数が決定されます。各パラレル問合せスレーブが、1 つ以上の索引パーティションを処理します。これは、パラレルに作成されたローカル索引のデフォルトの問合せ動作です。

パラレル問合せは、一般に、大規模なデータ・コレクションと複数の CPU を備え、同時ユーザー数が少ない DSS や分析システムに適しています。

ただし、同時ユーザー数が多く負荷の高いシステムでは、パラレル問合せによって全体的な問合せスループットが低下することがあります。さらに、次のような、パーティション・キー列順の上位 N 個のテキスト問合せでは、通常、パラレル問合せを使用するとパフォーマンスが低下します。

```
select * from (  
    select story_id from stories_tab where contains(...) > 0 order by  
    publication_date desc)  
    where rownum <= 10;
```

ALTER INDEX コマンドを次のように使用して、パラレル索引の操作後にパラレル問合せを使用禁止にできます。

```
Alter index <text index name> NOPARALLEL;  
Alter index <text index name> PARALLEL 1;
```

次のように指定すると、パラレル問合せを使用可能にするか、並列度を上げることができます。

```
Alter index <text index name> parallel < parallel degree >;
```


ブロック操作による問合せのチューニング

複数の述語を持つ問合せを発行すると、実行計画でブロック操作が行われる場合があります。たとえば、次の複合問合せについて考えます。

```
select docid from mytab where contains(text, 'oracle', 1) > 0
AND colA > 5
AND colB > 1
AND colC > 3;
```

すべての述語が非選択的で、colA、colB および colC がビットマップ索引を持つと想定します。Oracle のコストベースのオプティマイザは、次の実行計画を選択します。

```
TABLE ACCESS BY ROWIDS
  BITMAP CONVERSION TO ROWIDS
    BITMAP AND
      BITMAP INDEX COLA_BMX
      BITMAP INDEX COLB_BMX
      BITMAP INDEX COLC_BMX
    BITMAP CONVERSION FROM ROWIDS
      SORT ORDER BY
        DOMAIN INDEX MYINDEX
```

BITMAP AND はブロック操作であるため、Oracle では、BITMAP AND 操作を実行する前に、Oracle Text のドメイン索引から戻された ROWID とスコアのペアを一時的に保存する必要があります。

Oracle では、これらの ROWID とスコアのペアをメモリーに保存しようとします。ただし、これらの ROWID とスコアのペアを含む結果セットのサイズが SORT_AREA_SIZE 初期化パラメータの値を超える場合は、これらの結果がディスク上の一時セグメントに排出されます。

ディスクに結果を保存すると余分なオーバーヘッドが発生するため、次のように ALTER SESSION を使用して SORT_AREA_SIZE パラメータを増やすことによって、パフォーマンスが向上します。

```
alter session set SORT_AREA_SIZE = <new memory size in bytes>;
```

たとえば、バッファを約 8MB に設定するには、次の文を発行します。

```
alter session set SORT_AREA_SIZE = 8300000;
```

関連項目： SORT_AREA_SIZE の詳細は、『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』および『Oracle9i データベース・リファレンス』を参照してください。

問合せのパフォーマンスに関する FAQ（よくある質問）

この項では、問合せのパフォーマンスに関してよくある質問と、それに対する回答を提供します。

問合せのパフォーマンスとは何を意味しますか？

回答：問合せのパフォーマンスは、一般的に次の 2 つの基準で判断します。

- 応答時間。個々の問合せに対する応答を取得するまでの時間です。
- スループット。任意の時間内に実行可能な問合せの数（たとえば、1 秒当たりの問合せ数）です。

この 2 つは関係していますが、同一ではありません。通常、負荷の高いシステムでは高いスループットが必要ですが、比較的負荷の低いシステムでは応答時間を最短にすることが必要になります。また、アプリケーションによっては、問合せの全ヒットをユーザーに戻すことが必要なものもあれば、順序付けられた結果セットの最初の 20 ヒットのみを戻すことが必要なものもあります。この 2 つの状況を区別することが重要です。

テキスト問合せのうち、最速のタイプはどれですか？

回答：最速タイプの問合せは、次の条件を満たす問合せです。

- 単一の CONTAINS 句。
- WHERE 句内に他の条件がない。
- ORDER BY 句が存在しない。
- 結果の最初のページのみが戻される（たとえば、最初の 10 ヒットまたは 20 ヒット）。

表に関する統計を収集する必要がありますか？

回答：はい。表に関する統計を収集しておくことで、Oracle がコストベースの分析を実行できます。これにより、問合せに最も効率的な実行計画を Oracle が選択できます。

関連項目： この章の「[統計を使用した問合せの最適化](#)」を参照してください。

データのサイズは問合せにどのように影響しますか？

回答：テキスト索引が ROWID を戻す速度は、データの実際のサイズには影響されません。テキスト問合せの速度は、索引表からフェッチする必要のある行数、要求されるヒット数、問合せにより生成されるヒット数、およびソートの有無に関係します。

データの形式は問合せにどのように影響しますか？

回答：ドキュメントの形式（ASCII プレーン・テキスト、HTML または Microsoft Word）は、問合せ速度には影響しません。ドキュメントは、問合せ時ではなく索引付け時にプレーン・テキストにフィルタ処理されます。

データがクリーンであるかどうか、問合せに影響します。スペルチェック済みで編集作業が行われた出版用のテキストは、スペルミスや略語の多い電子メールなどの非公式のテキストと比べて、合計語彙数がかなり少なくなる傾向にあります（したがって、索引表のサイズも小さくなります）。指定した索引メモリー設定では、余分なテキストがあるとメモリー使用量が多くなり、クリーンなテキストと比べて断片化される行が増え、問合せの応答時間に悪影響を及ぼす可能性があります。

機能的検索と索引付き検索とは、何を意味しますか？

回答：カーネルがテキスト索引に対して問合せを行う方法は 2 つあります。1 つ目の方法では、カーネルはテキスト索引に対して特定のテキスト検索を満たすすべての ROWID を問い合わせます。これが最も一般的な方法です。これらの ROWID はまとめて戻されます。2 つ目の方法では、カーネルは個々の ROWID をテキスト索引に渡し、その特定の ROWID が特定のテキスト条件を満たすかどうかを問い合わせます。

2 つ目の方法は機能的検索と呼ばれ、選択性の高い構造化句がある場合に実行されます。このため、テキスト索引に対してチェックが必要な ROWID の数はわずかです。機能的検索が使用される検索の例は、次のとおりです。

```
SELECT ID, SCORE(1), TEXT FROM MYTABLE
      WHERE START_DATE = '21 Oct 1992'          <- 選択性が高い
      AND CONTAINS (TEXT, 'commonword') > 0;    <- 選択性が低い
```

構造化列（たとえば、日時や価格など）で ORDER BY を行い、テキスト問合せが非選択的な場合にも、機能的検索が使用されます。

問合せで使用される表はどれですか？

回答：すべての問合せで、索引トークン表が参照されます。この表の名前は、DR\$indexname\$I の形式です。この表には、トークンのリスト（TOKEN_TEXT 列）とトークンが発生する行とワードの位置に関する情報（TOKEN_INFO 列）が含まれます。

行情報は内部 DOCID 値として格納されます。この値は外部 ROWID 値として変換する必要があります。このために使用される表は、検索タイプにより異なります。機能的検索の場合は、\$K 表（DR\$indexname\$K）が使用されます。これは、DOCID と ROWID の各ペアに対する行を含む、単純な索引構成表（IOT）です。

索引付き検索の場合は、\$R 表（DR\$indexname\$R）が使用されます。この表の BLOB 列には ROWID の完全なリストが保持されます。

このため、SQL トレースを調べて \$K 表または \$R 表を検索すれば、機能的検索と索引付き検索のどちらが使用されているかが容易に判断できます。

注意： これらの内部索引表は、リリースごとに変更されることがあります。アプリケーションではこれらの表に直接アクセスしないことをお勧めします。

テキスト条件のみの検索速度は、結果をソートすることにより低下しますか？

回答： はい。低下します。

ソートしない場合、Oracle は検索したままの結果を返せます。通常、アプリケーションでは結果を一度に 1 ページずつ表示するため、この方が高速です。

スコアによる ORDER BY 順の問合せを速く行うには、どうすればよいですか？

回答： 関連性スコア (SCORE(n)) によるソートは、FIRST_ROWS(n) ヒントを使用すると、非常に高速になります。この場合、Oracle はテキスト索引表からフェッチするときに高速の内部ソートを実行します。

このような問合せの例を次に示します。

```
SELECT /*+ FIRST_ROWS(10) */ ID, SCORE(1), TEXT FROM MYTABLE
WHERE CONTAINS (TEXT, 'searchterm', 1) > 0
ORDER BY SCORE(1) DESC;
```

これを効率的に行うには、1 つの CONTAINS 以外の条件を WHERE 句に含めないでください。

どのメモリー設定が問合せに影響しますか？

回答： 問合せを行うには、大規模なシステム・グローバル領域 (SGA) を取得する必要があります。SGA 関連のパラメータは、Oracle 初期化ファイルに設定できます。これらのパラメータは動的に設定することもできます。

SORT_AREA_SIZE パラメータは、ORDER BY 問合せのソートで使用可能なメモリーを制御します。構造化列で頻繁に ORDER BY を行う場合は、このパラメータのサイズを増やす必要があります。

関連項目：

SGA 関連パラメータの設定方法の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

メモリー割当てと SORT_AREA_SIZE パラメータの設定方法の詳細は、『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

元表の LOB 列を表外に格納すると、パフォーマンスが向上しますか？

回答: はい。一般に、SELECT 文は元表から複数の列を選択します。Oracle では列をメモリーにフェッチするため、元表の LOB などの長い列は表外に格納の方が効率的です。これは、その列があまり更新されないが頻繁に選択される場合、特に有効です。

LOB を表外に格納するとき、問合せ中にメモリーにフェッチする必要があるのは LOB ロケータのみです。表外に格納すると、元表の有効なサイズが減少し、Oracle で表全体をメモリーにキャッシュしやすくなります。これにより、元表から列を選択するコストが低減し、テキスト問合せが高速になります。

さらに、メモリーにキャッシュされる元表が小さくなれば、問合せ中により多くの索引表データをキャッシュすることができるため、パフォーマンスが向上します。

複数の列に対する CONTAINS 問合せを高速にするには、どうすればよいですか？

回答: 最も高速の問合せタイプは、WHERE 句に CONTAINS 句が 1 つあるのみで、その他の条件が含まれていない問合せです。

複数の CONTAINS を含む次の問合せについて考えます。

```
SELECT title, isbn FROM booklist
WHERE CONTAINS (title, 'horse') > 0
AND CONTAINS (abstract, 'racing') > 0;
```

セクション検索と WITHIN 演算子を次のように使用すると、同じ結果が得られます。

```
SELECT title, isbn FROM booklist
WHERE CONTAINS (alltext,
'horse WITHIN title AND racing WITHIN abstract')>0;
```

2 つ目の問合せの方が高速です。このような問合せを使用するには、索引付けのためにすべてのデータを 1 つのテキスト列にコピーする必要があります。各列のデータはセクション・タグで囲みます。これは、索引付けの前に PL/SQL プロシージャを使用して実行するか、索引付け時に USER_DATASTORE データストアを使用して行い、構造化列とテキスト列を 1 つのドキュメントに統合します。

問合せに多数の拡張を使用してもかまいませんか？

回答：問合せに使用される個別のワードごとに、索引表から少なくとも 1 行をフェッチする必要があります。このため、拡張の数はできるだけ少なくします。

ワイルド・カード、シソーラス、ステミングおよびファジー・マッチングなどの拡張は、作業上必要でないかぎり、使用しないようにします。一般的に、多少の拡張は許容されますが（たとえば最大 20 など）、問合せでは 100 以上の拡張は避けるようにしてください。問合せ式の拡張の数を判断するには、問合せフィードバック・メカニズムを使用できます。

さらに、ワイルド・カードおよびステミングの問合せの場合は、プリフィックス、サブストリングまたはステム索引を作成すると、問合せ時から索引付け時への語句拡張コストをなくすことができます。問合せのパフォーマンスは上がりますが、索引付けの時間が長くなり、ディスク領域が大きくなります。

プリフィックス索引およびサブストリング索引により、ワイルド・カードのパフォーマンスが向上します。プリフィックスとサブストリングの索引付けは、**BASIC_WORDLIST** プリファレンスを使用して使用可能にします。次の例では、プリフィックスとサブストリングの索引付けに対してワードリスト・プリファレンスを設定します。プリフィックス索引付けについては、3～4 文字の長さのトークン・プリフィックスを作成するように指定します。

```
begin
  ctx_ddl.create_preference('mywordlist', 'BASIC_WORDLIST');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_INDEX', 'TRUE');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_MIN_LENGTH', '3');
  ctx_ddl.set_attribute('mywordlist', 'PREFIX_MAX_LENGTH', '4');
  ctx_ddl.set_attribute('mywordlist', 'SUBSTRING_INDEX', 'YES');
end;
```

ステム索引付けは、**BASIC_LEXER** プリファレンスを使用して使用可能にします。

```
begin
  ctx_ddl.create_preference('mylex', 'BASIC_LEXER');
  ctx_ddl.set_attribute ('mylex', 'index_stems', 'ENGLISH');
end;
```


ローカル・パーティション索引はどのような場合に便利ですか？

回答：ローカル・パーティション CONTEXT 索引は、パーティション表に対して作成できます。つまり、パーティション表上では、各パーティションに独自の索引表セットがあることを意味します。実際には複数の索引がありますが、各索引からの結果が必要に応じて組み合わせられ、最終的な結果セットが生成されます。

この索引は、LOCAL キーワードを使用して作成します。

```
CREATE INDEX index_name ON table_name (column_name)
INDEXTYPE IS ctxsys.context LOCAL
PARAMETERS ('...');
```

パーティション表およびローカル索引を使用すると、次のタイプの CONTAINS 問合せのパフォーマンスが向上します。

- [パーティション・キー列に対する範囲検索](#)

これは、パーティション・キー列の特定の値範囲に検索を限定する問合せです。

- [パーティション・キー列での ORDER BY](#)

これは、最初の N 個のヒットのみが必要で、ORDER BY 句がパーティション・キーを指定する問合せです。

関連項目： この章の「[ローカル・パーティション CONTEXT 索引を使用した応答時間の短縮](#)」を参照してください。

問合せはパラレルに実行する方がよいですか？

回答：場合によります。パラレルに作成された索引の場合にはパラレル問合せがデフォルト動作ですが、通常、負荷の高いシステムでは問合せの全体的なスループットが低下します。

一般に、パラレル問合せは、大規模なデータ・コレクションと複数の CPU を備え、同時ユーザー数が少ない DSS や分析システムに適しています。

関連項目： この章の「[パラレル問合せ](#)」を参照してください。

テーマには索引を付けた方がよいですか？

回答：テーマ情報に CONTEXT 索引で索引付けすると、時間がかかり、索引のサイズも大きくなります。ただし、テーマ索引を使用すると、ナレッジ・ベースが使用され（使用可能な場合）、ABOUT 問合せの精度が上がります。アプリケーションで ABOUT 問合せを頻繁に使用する場合は、索引に対してテーマ・コンポーネントを作成してみる価値はあります。ただし、索引の作成時間と格納領域が余分に必要になります。

関連項目： [第 3 章「問合せ」](#) の「[ABOUT 問合せおよびテーマ](#)」

CTXCAT 索引はどのような場合に作成するとよいですか？

回答: CTXCAT 索引は、テキストが小さなチャンク（最大で数行）になっていて、特定の構造化基準（通常は数値または日付）に従って、検索で結果セットを制限またはソートする必要がある場合（あるいはその両方）に、最も効率的に機能します。

たとえば、オンラインのオークション・サイトについて考えます。各競売対象品目には、短い説明、現在の入札価格、オークションの開始日と終了日が含まれています。この場合、説明に「アンティーク・キャビネット」が含まれていて、現在の入札価格が \$500 未満の全レコードを表示するとします。ユーザーが新しく提示される商品に特に興味がある場合は、結果をオークション開始日でソートする必要があります。

このような検索では、CONTEXT 索引に対して CONTAINS 構造化問合せを使用することは必ずしも効率的ではありません。この問合せでは、構造化句と CONTAINS 句に応じて、応答時間が大きく異なります。これは、構造化句と CONTAINS 句の共通部分またはテキスト問合せの順序付けが問合せ中に計算されるためです。

価格や日付などの構造化情報を CTXCAT 索引内に含めると、検索条件にかかわらず、問合せ応答時間は常に最適な範囲内にとどまります。これは、テキストと構造化問合せ間の相互作用が、索引付け時に事前に計算されるためです。この結果、問合せ応答時間が最適になります。

CTXCAT 索引が適さないのは、どのような場合ですか？

回答: 索引作成に要する時間と領域に違いがあります。CTXCAT 索引は CONTEXT 索引と比べて作成に時間がかかり、使用するディスク領域もかなり多くなります。ディスク領域に余裕がない場合は、CTXCAT 索引が適切かどうかを慎重に考慮する必要があります。

問合せ演算子に関しては、CATSEARCH 問合せでは問合せテンプレートを使用して、より豊富な CONTEXT 文法を使用できるようになっています。CATSEARCH 問合せ文法のみを使用しなければならないという古い制限はなくなりました。

使用可能なオプティマイザ・ヒントは何ですか。また、その機能はどのようなものですか？

回答: オプティマイザ・ヒント INDEX(table column) を通常の方法で使用すると、テキスト索引または B ツリー索引を使用した問合せを実行できます。

NO_INDEX(table column) ヒントを使用すると、特定の索引を使用禁止にすることもできます。

さらに、テキスト問合せでは FIRST_ROWS(n) ヒントが特殊な意味を持ち、問合せで最初の n 個のヒットが必要な場合に使用できます。FIRST_ROWS ヒントを ORDER BY SCORE(n) DESC とともに使用すると、Oracle に対して、ソート済みの集合をテキスト索引から受け入れ、それ以上のソートを実行しないように指示されます。

関連項目： この章の「[応答時間短縮のための問合せの最適化](#)」を参照してください。

索引付けのパフォーマンスに関する FAQ（よくある質問）

この項では、索引付けのパフォーマンスに関してよくある質問と、それに対する回答を提供します。

索引付けにはどのくらいの時間が必要ですか？

回答：テキストの索引付けは、リソース集中型の処理です。索引付けの速度は関連するハードウェアの性能に依存します。

判断基準としては、平均のドキュメント・サイズが 5KB で、次のハードウェア構成とパラレル構成の場合、Oracle Text では 1 秒間に約 200 のドキュメントに索引を付けられます。

- 400MHz の Sun Sparc CPU × 4
- 4GB の RAM
- EMC symmetrix (24 個のストライプ・ディスク)
- 並列度 5 (5 個のパーティション)
- 1 索引プロセス当たり 600MB の索引メモリー
- 平均 5KB の XML 形式のニュース・ドキュメント
- USER_DATASTORE

ドキュメント形式、データの場所、ユーザー定義データストアのコール、フィルタ、レクサーなどのその他の要因も、索引付けの速度に影響します。

どの索引メモリー設定を使用すればよいですか？

回答：索引メモリーは、システム・パラメータ DEFAULT_INDEX_MEMORY および MAX_INDEX_MEMORY を使用して設定できます。また、CREATE INDEX 時に memory パラメータを指定して、索引メモリーを実行時に設定することもできます。

DEFAULT_INDEX_MEMORY 値は、ページングが発生しない範囲で、できるだけ高く設定するようにします。

SORT_AREA_SIZE システム・パラメータを増やして、索引付けのパフォーマンスを向上させることもできます。

経験則では、索引メモリー設定を大きくすると（数百メガバイト程度）、索引付けの速度が上がり、最終的な索引の断片化が減少します。ただし、高く設定しすぎると、メモリーのページングが発生し、索引付けの速度が低下します。

索引の並列作成では、各プロセスがそれぞれ索引付け用のメモリーを必要とします。巨大な表を扱う場合には、索引作成時と検索時とで、システム・グローバル領域（SGA）のチューニング方法を変えます。問合せでは、システム・グローバル領域（SGA）のブロック・バッファ・キャッシュにできるだけ多くの情報をキャッシュするようにします。このため、ブロック・バッファ・キャッシュには大量のメモリーを割り当てる必要があります。ただし、この設定は索引付けには影響を与えないため、索引作成時には、SGA のサイズを減らして、索引付け処理中の索引メモリー設定を大きくする方が賢明です。

SGA のサイズは Oracle 初期化ファイルで設定します。

関連項目：

Oracle Text のシステム・パラメータの詳細は、『Oracle Text リファレンス』を参照してください。

SGA 関連パラメータの設定方法の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

メモリー割当てと SORT_AREA_SIZE パラメータの設定方法の詳細は、『Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

索引付けはどの程度のディスク・オーバーヘッドが必要ですか？

回答：オーバーヘッド（索引表に必要な領域の量）は、元のテキスト量の 50% から 200% まで様々です。一般に、テキストの総量が多ければ多いほどオーバーヘッドは小さくなりますが、多数の小さなレコードの方が、少数の大きなレコードよりもオーバーヘッドの消費が大きくなります。また、クリーンなデータ（出版済みのテキストなど）は、電子メールや討論記録などの未処理のデータと比べて、必要なオーバーヘッドは小さくて済みます。これは、未処理のデータにはスペルミスや略語などの特異なワードが数多く含まれている可能性が高いからです。

テキストのみの索引は、テキストとテーマを組み合わせた索引よりも小さくなります。プリフィックス索引およびサブストリング索引では、索引がかなり大きくなります。

データの形式によって索引付けにどのような影響がありますか？

回答：Microsoft Word ファイルなどの書式化されたドキュメントの場合は、ドキュメント内に含まれている実際のテキストに比べてドキュメント・サイズが大きくなる傾向にあるため、記憶域オーバーヘッドはかなり低くて済みます。1GB の Word ドキュメントに必要な索引領域は 50MB のみであるのに対して、1GB のプレーン・テキストの場合は 500MB 必要になることがあります。これは、後者のファイルには前者よりも 10 倍のプレーン・テキストが含まれている可能性があるためです。

索引付けに要する時間については、これほど単純ではありません。索引付け対象のテキスト量が減れば明らかな影響は出ますが、索引付けに要する時間を見積もるためには、INSO フィルタまたはその他のユーザー定義フィルタを使用してドキュメントをフィルタ処理する時間を相殺することが必要になります。

索引付けはパラレルに実行できますか？

回答：はい、索引付けはパラレルに実行できます。データが大量にあり、CPU も複数ある場合は、パラレル索引付けにより、索引付けのパフォーマンスが向上します。

索引を作成するときに、次のように **PARALLEL** キーワードを使用します。

```
CREATE INDEX index_name ON table_name (column_name)
INDEXTYPE IS ctxsys.context PARAMETERS ('...') PARALLEL 3;
```

これで、リソースに応じて最大 3 つの索引付けプロセスを使用して索引が作成されます。

注意： 以前のリリースのように、パラレル索引付けのためにパーティション表を作成する必要はなくなりました。

注意： ローカル索引をパラレルで作成する（実際にはシリアルに実行される）と、後続の問合せはデフォルトでパラレルに処理されます。非パーティション索引をパラレルに作成しても、問合せはパラレルに処理されません。

パラレル問合せにより、特に負荷の高いシステムでは、問合せのスループットが低下します。このため、索引作成後はパラレル問合せを使用禁止にすることをお勧めします。この場合は、**ALTER INDEX NOPARALLEL** を使用します。

ローカル・パーティション索引をパラレルに作成するには、どうすればよいですか？

回答：ローカル索引をパラレルに作成すると、索引付けのパフォーマンスが向上します。

ただし、現時点では、**CREATE INDEX** で **PARALLEL** パラメータを使用して、ローカル・パーティション表をパラレルに作成することはできません。作成しようとすると、このパラメータは無視され、索引付けは逐次処理されます。

ローカル索引をパラレルに作成するには、使用禁止の索引を最初に作成してから、**DBMS_PCLXUTIL.BUILD_PART_INDEX** ユーティリティを実行します。

次の例では、元表にパーティションが 3 つあります。使用禁止のローカル・パーティション索引を最初に作成してから、**DBMS_PCLXUTIL.BUILD_PART_INDEX** ユーティリティを実行します。このユーティリティにより、3 つのパーティションがパラレルに作成されます（パーティション間並列性）。また、各パーティション内では、索引の作成が並列度 2 によりパラレルに実行されます（パーティション内並列性）。


```
create index tdrbip02bx on tdrbip02b(text)
indextype is ctxsys.context local (partition tdrbip02bx1,
                                     partition tdrbip02bx2,
                                     partition tdrbip02bx3)

unusable;

exec dbms_pclxutil.build_part_index(3,2,'TDRBIP02B','TDRBIP02BX',TRUE)
```

索引付けの進捗を確認するには、どうすればよいですか？

回答：CTX_OUTPUT.START_LOG プロシージャを使用すると、索引付けプロセスからの出力を記録できます。ファイル名は、通常 \$ORACLE_HOME/ctx/log に書き込まれますが、CTX_ADM.SET_PARAMETER で LOG_DIRECTORY パラメータを使用してディレクトリを変更できます。

関連項目： このプロシージャの使用方法は、『Oracle Text リファレンス』を参照してください。

索引の更新に関する FAQ（よくある質問）

この項では、索引の更新と関連するパフォーマンスの問題に関してよくある質問と、それに対する回答を提供します。

新規レコードまたは更新済みレコードに索引を付ける頻度は、どれくらいがよいですか？

回答：必要に応じて異なります。CTX_DLL.SYNC_INDEX を使用して再索引付けを実行する頻度が低いほど、索引の断片化が少なくなり、索引の最適化を実行する頻度も低くて済みます。

ただし、これではデータが次第に古くなるため、ユーザーにとっては不便になる可能性があります。

ほとんどのシステムでは、毎日の夜間の索引付けで十分です。つまり、作成されてから 1 日未満のデータは検索できないということです。毎時、10 分ごとまたは 5 分ごとに更新を行うシステムもあります。

関連項目： CTX_DDL.SYNC_INDEX の使用方法は、『Oracle Text リファレンス』を参照してください。

[第 2 章「索引付け」の「CONTEXT 索引に関する DML 操作の管理」](#)

索引の断片化は、どのようにするとわかりますか？

回答：最善の方法は、いくつかの問合せに要する時間を測定してから、索引の最適化を実行し、その後で同じ問合せの時間を再度測定することです（SGA を消去するために、その都度データベースを再起動する必要があります）。問合せの速度が大幅に上がっている場合は、最適化が有効であったことを示しています。そうでない場合は、次回は待機時間が長くなる可能性があります。

索引の断片化の分析には、CTX_REPORT.INDEX_STATS を使用することもできます。

関連項目： CTX_REPORT パッケージの使用の詳細は、『Oracle Text リファレンス』を参照してください。

[第 2 章「索引付け」の「索引の最適化」](#)

メモリー割当ては索引の同期化に影響しますか？

回答：はい。通常の索引付けへの影響と同じです。同期操作では索引付け対象のレコード数が大幅に少ない場合が多いため、通常は索引付け用メモリーに数百メガバイトを準備する必要はありません。

ドキュメントのセクション検索

この章では、Oracle Text の問合せアプリケーションでドキュメントのセクションを使用する方法を説明します。

次の項目について説明します。

- [ドキュメントのセクション検索](#)
- [HTML のセクション検索](#)
- [XML のセクション検索](#)

ドキュメントのセクション検索

セクション検索を使用すると、テキスト問合せをドキュメント内のテキストのブロックに絞り込むことができます。セクション検索は、HTML や XML のドキュメントのように、ドキュメントに内部構造がある場合に有効です。

また、テキストを文レベルと段落レベルで検索できます。

セクション検索の使用可能化

- ドキュメント・コレクションのセクション検索を使用可能にする手順は、次のとおりです。
1. セクション・グループを作成します。
 2. セクションを定義します。
 3. ドキュメントを索引付けします。
 4. WITHIN、INPATH または HASPATH の各演算子を使用してセクションを検索します。

セクション・グループの作成

セクション検索を使用可能にするには、セクション・グループを定義します。システム定義のセクション・グループのいずれかを使用して、セクション・グループのインスタンスを作成します。ドキュメント・コレクションに適したセクション・グループを選択します。

セクション・グループを使用して、所有しているドキュメント・セットのタイプを指定し、タグ構造を明示的に示します。たとえば、HTML タグ付きのドキュメントを索引付けするには、HTML_SECTION_GROUP を使用します。同様に、XML タグ付きのドキュメントを索引付けするには、XML_SECTION_GROUP を使用します。

次の表は、使用可能な各種のセクション・グループを示しています。

セクション・グループ・プリファレンス	説明
NULL_SECTION_GROUP	これはデフォルトです。どのセクションも定義しないか、または SENTENCE セクションか PARAGRAPH セクションのみを定義します。
BASIC_SECTION_GROUP	開始および終了タグが <A> および という形式のセクションを定義します。 注意： このグループ・タイプでは、片方みのカッコ、コメント・タグおよび属性などの入力サポートしません。このような入力には、HTML_SECTION_GROUP を使用します。
HTML_SECTION_GROUP	HTML ドキュメントを索引付けし、HTML ドキュメントにセクションを定義します。

セクション・グループ・プリファレンス	説明
XML_SECTION_GROUP	XML ドキュメントを索引付けし、XML ドキュメントにセクションを定義します。
AUTO_SECTION_GROUP	<p>XML ドキュメントの開始タグ / 終了タグに対して自動的にゾーン・セクションを作成します。XML タグから導出されるセクション名は、XML 内と同様に大 / 小文字が区別されます。</p> <p>属性セクションは、属性を持つ XML タグに対して自動的に作成されます。属性セクションは、attribute@tag という形式でネーミングされます。</p> <p>停止セクション、空のタグ、処理の指示およびコメントは、索引付けされません。</p> <p>自動セクション・グループには次の制限事項が適用されます。</p> <ul style="list-style-type: none">■ ゾーン、フィールドまたは特殊セクションは、自動セクション・グループに追加できません。■ 自動セクション化は、XML ドキュメント・タイプ（ルート要素）を索引付けしません。ただし、ドキュメント・タイプに停止セクションを定義することはできます。■ 索引付けされたタグの長さは、プリフィックスおよび名前空間を含めて 64 文字以下です。これより長いタグは索引付けされません。
PATH_SECTION_GROUP	<p>XML ドキュメントを索引付けします。このタイプは、AUTO_SECTION_GROUP と同じように動作します。</p> <p>相違点は、このセクション・グループを使用すると、INPATH および HASPATH 演算子でパス検索を実行できることです。タグまたは属性名の間合せでは、大 / 小文字が同様に区別されます。</p>
NEWS_SECTION_GROUP	RFC 1036 に基づいて、ニュース・グループ形式のドキュメントのセクションを定義します。

CTX_DDL パッケージを使用してセクション・グループを作成し、セクション・グループの構成要素としてセクションを定義します。たとえば、HTML ドキュメントを索引付けするには、HTML_SECTION_GROUP を使用してセクション・グループを作成します。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
end;
```


セクションの定義

セクションは、セクション・グループの構成要素として定義します。次の例では、HTML <H1> タグ内の全テキストに対して **heading** というゾーン・セクションを定義します。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_zone_section('htmgroup', 'heading', 'H1');
end;
```

注意： AUTO_SECTION_GROUP または PATH_SECTION_GROUP を使用して、XML ドキュメント・コレクションを索引付けする場合は、システムが索引付け時にセクションの定義を行います。したがって、セクションを明示的に定義する必要はありません。

関連項目： セクションの詳細は、この章の「[セクションのタイプ](#)」を参照してください。

XML のセクション検索の詳細は、この章の「[XML のセクション検索](#)」を参照してください。

ドキュメントの索引付け

ドキュメントの索引付け時に、セクション・グループを CREATE INDEX の PARAMETERS 句に指定します。

```
create index myindex on docs(htmlfile) indextype is ctxsys.context
parameters('filter ctxsys.null_filter section group htmgroup');
```

WITHIN 演算子によるセクション検索

ドキュメントが索引付けされている場合は、WITHIN 演算子を使用して、セクション内を問い合わせることができます。たとえば、ドキュメントのヘッダー内にワード **Oracle** を含むすべてのドキュメントを検索するには、次の問合せを発行します。

```
'Oracle WITHIN heading'
```

関連項目： WITHIN 演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

INPATH および HASPATH 演算子によるパス検索

PATH_SECTION_GROUP を使用した場合、システムは自動的に XML セクションを作成します。WITHIN 演算子を使用して問合せを発行できる他、INPATH 演算子と HASPATH 演算子を使用してパス検索を発行できます。

関連項目： これらの演算子の使用方法は、この章の「[XML のセクション検索](#)」を参照してください。

INPATH 演算子の詳細は、『Oracle Text リファレンス』を参照してください。

セクションのタイプ

セクション・タイプはすべて、ドキュメント内のテキストのブロックです。ただし、セクションを区切る方法や索引内での記録方法にそれぞれ相違があります。セクションは、次のいずれかです。

- ゾーン・セクション
- フィールド・セクション
- 属性セクション (XML ドキュメントの場合)
- 特殊セクション (文または段落)

ゾーン・セクション

ゾーン・セクションは、ドキュメント内の開始タグと終了タグで区切られたテキストの本体です。開始タグと終了タグの位置は、索引内に記録されています。したがって、タグに挟まれたワードはそのセクション内に存在するとみなされます。ゾーン・セクションのインスタンスすべてに、開始タグと終了タグが付いている必要があります。

たとえば、<TITLE> と </TITLE> の 2 つのタグに挟まれたテキストは、ゾーン・セクションとして次のように定義できます。

```
<TITLE>Tale of Two Cities</TITLE>
It was the best of times...
```

ゾーン・セクションは、ドキュメント内でネスト、オーバーラップおよび繰返しができます。

ゾーン・セクションの間合せ時に、全セクション内の語句を検索するには、WITHIN 演算子を使用します。定義済みのセクション内に間合せ語句を含むドキュメントが戻ります。

ゾーン・セクションは、HTML および XML ドキュメントのセクションの定義に最適です。ゾーン・セクションを定義するには、CTX_DDL.ADD_ZONE_SECTION を使用します。

たとえば、次のように **booktitle** というセクションを定義したとします。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_zone_section('htmgroup', 'booktitle', 'TITLE');
end;
```


索引付けの後、セクション *booktitle* 内の語句 *Cities* を含むすべてのドキュメントを次のように検索できます。

```
'Cities WITHIN booktitle'
```

(*dog and cat*) *WITHIN booktitle* のような複数の問合せ語句の場合は、*booktitle* セクションの同じインスタンス内に *cat* と *dog* を含むドキュメントが検出されます。

繰返しゾーン・セクション ゾーン・セクションは繰返しが可能です。各繰返しは別々のセクションとして処理されます。たとえば、<H1> が heading セクションを示す場合は、次のように、同じドキュメント内で繰り返すことができます。

```
<H1> The Brown Fox </H1>
<H1> The Gray Wolf </H1>
```

これらのゾーン・セクションが *Heading* という名前の場合、問合せ *Brown WITHIN Heading* はこのドキュメントを戻します。ただし、(*Brown and Gray*) *WITHIN Heading* という問合せはこのドキュメントを戻しません。

ゾーン・セクションのオーバーラップ ゾーン・セクションは互いにオーバーラップできます。たとえば、 と <I> が 2 つの異なるゾーン・セクションを示す場合、これらはドキュメント内で次のようにオーバーラップできます。

```
plain <B> bold <I> bold and italic </B> only italic </I> plain
```

ネストされたゾーン・セクション ゾーン・セクションは（それ自体も含めて）、次のようにネストできます。

```
<TD> <TABLE><TD>nested cell</TD></TABLE></TD>
```

WITHIN 演算子を使用して、セクション内のセクションのテキストを検索するための問合せを記述できます。たとえば、**BOOK1**、**BOOK2** および **AUTHOR** のゾーン・セクションが、ドキュメント *doc1* および *doc2* で次のように出現するとします。

doc1:

```
<book1> <author>Scott Tiger</author> This is a cool book to read.<book1>
```

doc2:

```
<book2> <author>Scott Tiger</author> This is a great book to read.<book2>
```

次のようにネストされた問合せを実行します。

```
'Scott within author within book1'
```

この問合せは *doc1* のみを戻します。

フィールド・セクション

フィールド・セクションは、ゾーン・セクションと同じように、開始タグと終了タグで区切られたテキストの範囲です。フィールド・セクションがゾーン・セクションと異なる点は、その範囲がドキュメントの残りの部分とは別に索引付けされることです。

フィールド・セクションは異なる方法で索引付けされるため、大量のドキュメントが索引付けされている場合は、ゾーン・セクションに比べて問合せパフォーマンスが向上します。

フィールド・セクションは、ドキュメント内のセクションに1回のみ出現するニュース・ヘッダーのフィールドなどに最適です。また、フィールド・セクションは、ドキュメントの残りの部分で参照できます。

ゾーン・セクションとは異なり、フィールド・セクションには次の制限事項があります。

- フィールド・セクションはオーバーラップできません。
- フィールド・セクションは繰返しができません。
- フィールド・セクションはネストできません。

参照可能なフィールド・セクションと参照不能なフィールド・セクション デフォルトでは、フィールド・セクションは、ドキュメントの残りの部分とは別のサブドキュメントとして索引付けされます。フィールド・セクションは、前後のテキストから参照不能です。したがって、WITHIN 句でそのセクション名を明示的に指定することによってのみ問い合わせることができます。

フィールド・セクション内のテキストをドキュメント全体の一部として索引付けする場合は、フィールド・セクションを参照可能にできます。参照可能なフィールド・セクション内のテキストは、WITHIN 演算子を使用するかどうかにかかわらず問い合わせることができます。

次の例では、参照不能なフィールド・セクションと参照可能なフィールド・セクションの相違点を示します。

次のコードは、BASIC_SECTION_GROUP 型のセクション・グループ basicgroup を定義します。次に、<A> タグに対して Author というフィールド・セクションを basicgroup に作成します。また、参照可能フラグを FALSE に設定して、参照不能なセクションを作成します。

```
begin
ctx_ddl_create_section_group('basicgroup', 'BASIC_SECTION_GROUP');
ctx_ddl.add_field_section('basicgroup', 'Author', 'A', FALSE);
end;
```

Author フィールド・セクションは参照不能であるため、Author セクション内のテキストを検索するには、次のように WITHIN 演算子を使用する必要があります。

```
'(Martin Luther King) WITHIN Author'
```


WITHIN 演算子を使用せずに *Martin Luther King* を問い合せても、フィールド・セクション内のこの語句のインスタンスは戻りません。WITHIN を指定せずにフィールド・セクション内のテキストを問い合わせる場合は、セクション作成時に次のように参照可能フラグを TRUE に設定する必要があります。

```
begin
ctx_ddl.add_field_section('basicgroup', 'Author', 'A', TRUE);
end;
```

ネストされたフィールド・セクション フィールド・セクションはネストできません。たとえば、あるフィールド・セクションを <TITLE> で始まるように定義し、別のフィールド・セクションを <FOO> で始まるように定義した場合、この 2 つのセクションを、次のようにネストすることはできません。

```
<TITLE> dog <FOO> cat </FOO> </TITLE>
```

ネストされたセクションを使用するには、これらをゾーン・セクションとして定義します。

繰返しフィールド・セクション 繰返しフィールド・セクションは使用できますが、WITHIN 問合せはこれらを 1 つのセクションとして処理します。次に、ドキュメントの繰返しフィールド・セクションの例を示します。

```
<TITLE> cat </TITLE>
<TITLE> dog </TITLE>
```

問合せ (*dog and cat*) *within title* は、これらのワードが別のセクション内に存在している場合でも、ドキュメントを検出します。

WITHIN 問合せで繰返しセクションを区別するには、これらをゾーン・セクションとして定義します。

属性セクション

属性セクションを定義すると、XML の属性テキストを問い合わせることができます。また、システムで自動的に XML 属性を定義し、索引付けすることもできます。

関連項目： この章の「[XML のセクション検索](#)」を参照してください。

特殊セクション

特殊セクションは、タグで認識されるものではありません。現在サポートされている特殊セクションは、文と段落のみです。特殊セクションを使用すると、文または段落内のワードの組合せを検索できます。

文および段落の境界はレクサーにより決定されます。たとえば、BASIC_LEXER は、文および段落のセクション境界を次のように認識します。

表 6-1

特殊セクション	境界
SENTENCE	WORD/PUNCT/WHITESPACE
	WORD/PUNCT/NEWLINE
PARAGRAPH	WORD/PUNCT/NEWLINE/WHITESPACE
	WORD/PUNCT/NEWLINE/NEWLINE

レクサーが境界を認識できない場合、文または段落のセクションには索引が付けられません。

特殊セクションを追加するには、CTX_DDL.ADD_SPECIAL_SECTION プロシージャを使用します。たとえば、次のコードによって、HTML ドキュメントの文内での検索が使用可能になります。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_special_section('htmgroup', 'SENTENCE');
end;
```

文検索の他にゾーン検索を使用可能にするために、ゾーン・セクションをグループに追加することもできます。次の例では、ゾーン・セクション **Headline** をセクション・グループ **htmgroup** に追加します。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_special_section('htmgroup', 'SENTENCE');
ctx_ddl.add_zone_section('htmgroup', 'Headline', 'H1');
end;
```


HTML のセクション検索

HTML には、セクション検索に使用できるタグ付きテキスト形式の内部構造があります。たとえば、<H1> タグに対して **heading** というセクションを定義できます。これによって、ドキュメント・セット全体のこのタグ内でのみ語句を検索できます。

問合せには、WITHIN 演算子を使用します。**heading** セクション内に問合せ語句を含むすべてのドキュメントを検出します。したがって、**heading** セクション内にワード **oracle** を含むすべてのドキュメントを検索する場合は、次の問合せを発行します。

```
'oracle within heading'
```

HTML のセクションの作成

次のコードは、HTML_SECTION_GROUP 型の htmgroup というセクション・グループを定義します。その後、htmgroup に、<H1> タグで識別する heading というゾーン・セクションを作成します。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_zone_section('htmgroup', 'heading', 'H1');
end;
```

ドキュメントを次のように索引付けできます。

```
create index myindex on docs(htmlfile) indextype is ctxsys.context
parameters('filter ctxsys.null_filter section group htmgroup');
```

セクション・グループ htmgroup で索引付けした後、次のように問合せを発行して heading セクション内を問い合わせることができます。

```
'Oracle WITHIN heading'
```


HTML の Meta タグの検索

HTML ドキュメントの場合は、<META> タグの NAME/CONTENT のペアにセクションを作成できます。セクションの作成時に、検索を CONTENT 内のテキストに制限できます。

例：<META> タグに対するセクションの作成

次のような META タグを持つ HTML ドキュメントがあるとします。

```
<META NAME="author" CONTENT="ken">
```

NAME 値が **author** である META タグについて、すべての CONTENT 属性を索引付けるゾーン・セクションを作成します。

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
ctx_ddl.add_zone_section('htmgroup', 'author', 'meta@author');
end
```

セクション・グループ htmgroup で索引付けした後、次のようにドキュメントを問い合わせることができます。

```
'ken WITHIN author'
```

XML のセクション検索

HTML ドキュメントと同様、XML ドキュメントにもタグ付きテキストがあります。このタグ付きテキストを使用してテキストのブロックを定義すると、セクション検索を実行できます。セクションの内容は、WITHIN 演算子または INPATH 演算子を使用して検索できます。

XML の検索では、次の操作を実行できます。

- 自動セクション
- 属性検索
- ドキュメント・タイプ別のセクション
- パス・セクション検索

自動セクション

セクション・グループ `AUTO_SECTION_GROUP` を使用して、XML ドキュメントからセクションを自動的に作成するように索引付け操作を設定できます。システムは、XML タグに対してゾーン・セクションを作成します。属性セクションは、属性を持つタグに対して作成され、`tag@attribute` という形式でネーミングされます。

たとえば、次のコマンドは、`AUTO_SECTION_GROUP` を使用して XML ファイルを含む列に索引 *myindex* を作成します。

```
CREATE INDEX myindex ON xmldocs(xmlfile) INDEXTYPE IS ctxsys.context PARAMETERS
('datastore ctxsys.default_datastore filter ctxsys.null_filter section group
ctxsys.auto_section_group');
```

属性の検索

XML 属性テキストは、次のいずれかの方法で検索できます。

- `CTX_DDL.ADD_ATTR_SECTION` を使用して属性セクションを作成し、次に `XML_SECTION_GROUP` を使用して索引付けします。索引付け時に、`AUTO_SECTION_GROUP` を使用すると、属性セクションは自動的に作成されます。属性セクションは、`WITHIN` 演算子で問い合わせることができます。
- `PATH_SECTION_GROUP` を使用して索引付けし、`INPATH` 演算子で属性テキストを問い合わせます。

属性セクションの作成

次のように、`TITLE` 属性を持つ `BOOK` タグを定義する XML ファイルがあるとします。

```
<BOOK TITLE="Tale of Two Cities">
  It was the best of times.
</BOOK>
```

タイトル属性を属性セクションとして定義するには、次のように `XML_SECTION_GROUP` を作成し、属性セクションを定義します。

```
begin
ctx_ddl.create_section_group('myxmlgroup', 'XML_SECTION_GROUP');
ctx_ddl.add_attr_section('myxmlgroup', 'booktitle', 'book@title');
end;
```

索引付けは、次のように行います。

```
CREATE INDEX myindex ON xmldocs(xmlfile) INDEXTYPE IS ctxsys.context PARAMETERS
('datastore ctxsys.default_datastore filter ctxsys.null_filter section group
myxmlgroup');
```


XML 属性セクション *booktitle* は、次のように問い合わせることができます。

```
'Cities within booktitle'
```

INPATH 演算子による属性の検索

属性テキストは、INPATH 演算子を使用して検索できます。そのためには、XML ドキュメント・セットを PATH_SECTION_GROUP で索引付けする必要があります。

関連項目： この章の「[パス・セクション検索](#)」を参照してください。

ドキュメント・タイプ別のセクションの作成

異なるドキュメント・タイプに対して宣言された <book> タグを持つ XML ドキュメント・セットがあるとします。各ドキュメント・タイプに対して個別の book セクションを作成する必要がある場合を考えてみます。

mydocname1 が XML ドキュメント・タイプ（ルート要素）として、次のように宣言されているとします。

```
<!DOCTYPE mydocname1 ... [...
```

mydocname1 の中で、要素 <book> が宣言されています。このタグに対し、タグのドキュメント・タイプを区別する mybooksec1 という名前のセクションを次のように作成できます。

```
begin
  ctx_ddl.create_section_group('myxmlgroup', 'XML_SECTION_GROUP');
  ctx_ddl.add_zone_section('myxmlgroup', 'mybooksec1', 'mydocname1(book)');
end;
```

mydocname2 が別の XML ドキュメント・タイプ（ルート要素）として、次のように宣言されているとします。

```
<!DOCTYPE mydocname2 ... [...
```

mydocname2 の中で、要素 <book> が宣言されています。このタグに対し、タグのドキュメント・タイプを区別する mybooksec2 という名前のセクションを次のように作成できます。

```
begin
  ctx_ddl.create_section_group('myxmlgroup', 'XML_SECTION_GROUP');
  ctx_ddl.add_zone_section('myxmlgroup', 'mybooksec2', 'mydocname2(book)');
end;
```

セクション mybooksec1 内で問合せを行うには、WITHIN を次のように使用します。

```
'oracle within mybooksec1'
```


パス・セクション検索

XML ドキュメントには、次のような親子タグの構造を設定できます。

```
<A> <B> <C> dog </C> </B> </A>
```

この例のタグ C は、タグ A の子であるタグ B の子です。

Oracle Text では、PATH_SECTION_GROUP を使用してパス検索を実行できます。このセクション・グループを使用すると、問合せ内に直接の親子関係を指定できます。たとえば、要素 B の子である要素 C にある語句 *dog* を含むすべてのドキュメントを検索できます。

PATH_SECTION_GROUP を使用すると、属性値検索および属性の等価性のテストも実行できます。

この機能に関連する新しい演算子は、次のとおりです。

- INPATH
- HASPATH

PATH_SECTION_GROUP による索引の作成

パス・セクション検索を使用可能にするには、PATH_SECTION_GROUP を使用して XML ドキュメント・セットを索引付けします。

次のようにプリファレンスを作成します。

```
begin
ctx_ddl.create_section_group('xmlpathgroup', 'PATH_SECTION_GROUP');
end;
```

次のように索引を作成します。

```
CREATE INDEX myindex ON xmldocs(xmlfile) INDEXTYPE IS ctxsys.context PARAMETERS
('datastore ctxsys.default_datastore filter ctxsys.null_filter section group
xmlpathgroup');
```

索引を作成するときに、INPATH 演算子と HASPATH 演算子を使用できます。

トップレベルのタグ検索

トップレベルのタグ <A> に、語句 *dog* が含まれているすべてのドキュメントを検索するには、次の問合せを行います。

```
dog INPATH (/A)
```

または

```
dog INPATH(A)
```


任意レベルのタグ検索

任意レベルの `<A>` タグに語句 *dog* が含まれているすべてのドキュメントを検索するには、次の問合せを行います。

```
dog INPATH (//A)
```

この問合せは、次のドキュメントを検出します。

```
<A>dog</A>
```

および

```
<C><B><A>dog</A></B></C>
```

直接の親子関係の検索

トップレベルの要素 *A* の直接の子である要素 *B* に語句 *dog* が含まれているすべてのドキュメントを検索するには、次の問合せを行います。

```
dog INPATH (A/B)
```

この問合せは、次の XML ドキュメントを検出します。

```
<A><B>My dog is friendly.</B></A>
```

ただし、次のドキュメントは検出しません。

```
<C><B>My dog is friendly.</B></C>
```

タグ値のテスト

タグの値をテストできます。たとえば、次の問合せがあるとします。

```
dog INPATH (A[B="dog"])
```

この問合せは、次のドキュメントを検出します。

```
<A><B>dog</B></A>
```

ただし、次のドキュメントは検出しません。

```
<A><B>My dog is friendly.</B></A>
```


属性の検索

属性の内容を検索できます。たとえば、次の問合せがあるとします。

```
dog INPATH (//A/@B)
```

この問合せは、次のドキュメントを検出します。

```
<C><A B="snoop dog"> </A> </C>
```

属性値のテスト

属性の値をテストできます。たとえば、次の問合せがあるとします。

```
California INPATH (//A[@B = "home address"])
```

この問合せは、次のドキュメントを検出します。

```
<A B="home address">San Francisco, California, USA</A>
```

ただし、次のドキュメントは検出しません。

```
<A B="work address">San Francisco, California, USA</A>
```

パスのテスト

HASPATH 演算子を使用して、パスの存在をテストできます。たとえば、次の問合せがあるとします。

```
HASPATH (A/B/C)
```

この問合せは、ドキュメントを検出し、スコア 100 を戻します。

```
<A><B><C>dog</C></B></A>
```

この問合せでは、*dog* は参照されません。

HASPATH によるセクションの等価性のテスト

HASPATH 演算子を使用すると、セクションの等価性をテストできます。たとえば、次の問合せがあるとします。

```
dog INPATH A
```

この問合せは、次のドキュメントを検出します。

```
<A>dog</A>
```

さらに、次のドキュメントも検出します。

```
<A>dog park</A>
```


問合せを、語句 *dog* のみに制限し、それ以外の語句を検索しないようにする場合は、**HASPATH** 演算子によるセクションの等価性のテストを使用できます。たとえば、次の問合せがあるとして。

```
HASPATH (A="dog")
```

この問合せは、最初のドキュメントを検索し、スコア 100 を戻します。2 番目のドキュメントは検索しません。

関連項目： INPATH 演算子と HASPATH 演算子の使用方法の詳細は、『Oracle Text リファレンス』を参照してください。

シソーラスの使用

この章では、問合せアプリケーションをシソーラスを使用して改善する方法を説明します。
次の項目について説明します。

- シソーラスの概要
- シソーラスの用語の定義
- 問合せアプリケーションでのシソーラスの使用
- 提供されるナレッジ・ベース

シソーラスの概要

問合せアプリケーションのユーザーが、特定のトピックに関する情報を検索するとき、そのトピックについて書かれたドキュメント内で使用されているワードがわからない場合があります。

Oracle Text では、ワードと句のシノニムおよび階層関係を定義する大 / 小文字を区別するシソーラスまたは大 / 小文字を区別しないシソーラスを作成できます。その結果、シソーラスで定義した類似語または関連語が含まれるように問合せを拡張して、関連テキストを含むドキュメントを取り出すことができます。

シソーラスは、アプリケーション開発者が作成し、システムにロードできます。

注意： Oracle Text のシソーラスの形式および機能性は、ISO-2788 および ANSI Z39.19 (1993) の両標準に準拠しています。

シソーラスの作成とメンテナンス

シソーラスとシソーラスのエントリは、CTXAPP ロールを持つすべての Oracle Text ユーザーが作成、変更および削除できます。

CTX_THES パッケージ

シソーラスをプログラムでメンテナンスおよびブラウズするには、PL/SQL パッケージ CTX_THES を使用できます。このパッケージを使用すると、語句および階層関係のブラウズ、語句の追加と削除、およびシソーラス・リレーションの追加と削除を実行できます。

シソーラス演算子

また、CONTAINS 句でシソーラス演算子を使用すると、ロードしたシソーラスに基づいて問合せ語句を拡張できます。たとえば、SYN 演算子を使用すると、次のように *dog* などの語句をそのシノニムに拡張できます。

'syn(dog)'

ctxload ユーティリティ

ctxload ユーティリティは、シソーラスをプレーン・テキスト・ファイルからシソーラス表にロードする場合、およびシソーラスをシソーラス表から出力（ダンプ）ファイルにダンプする場合に使用できます。

ctxload で作成されたシソーラスのダンプ・ファイルは、出力したり、他のアプリケーションの入力として使用できます。ダンプ・ファイルは、シソーラス表にシソーラスをロードする場合にも使用できます。このことは、既存のシソーラスを新しいシソーラスを作成するための基礎として使用する場合に便利です。

大 / 小文字を区別するシソーラス

大 / 小文字を区別するシソーラスでは、語句（ワードおよび句）は、入力したとおり正確に格納されます。たとえば、語句を大 / 小文字の混合で入力した場合（CTX_THES パッケージまたはシソーラスのロード・ファイルのいずれかを使用）、シソーラスでは、そのエントリを大 / 小文字の混合で格納します。

注意： 大 / 小文字を区別するシソーラスから取得した問合せ拡張を利用するには、索引も大 / 小文字を区別する必要があります。

シソーラスのロード時に、**-thescase** パラメータを使用して、大 / 小文字を区別してロードするように指定できます。

CTX_THES.CREATE_THESAURUS によるシソーラスの作成時に、大 / 小文字を区別して作成するように指定できます。

また、大 / 小文字を区別するシソーラスを問合せで指定した場合、シソーラス検索では、問合せ語句は問合せで入力したとおりに正確に使用されます。したがって、大 / 小文字を区別するシソーラスを使用する問合せでは、問合せ拡張における精度が向上し、検索が容易になります。ただし、索引も大 / 小文字を区別している必要があります。

たとえば、大 / 小文字を区別するシソーラスを作成する場合、異なる意味を持つ語句、たとえば *Turkey*（国名）と *turkey*（鳥の種類）には、異なるエントリが使用されます。このシソーラスを使用すると、*Turkey* への問合せは、*Turkey* に対応付けられているエントリのみを含むように拡張されます。

大 / 小文字を区別しないシソーラス

大 / 小文字を区別しないシソーラスでは、語句は、入力時の文字に関係なく、すべて大文字で格納されます。

ctxload プログラムは、デフォルトでは大 / 小文字を区別せずにシソーラスをロードします。

CTX_THES.CREATE_THESAURUS によるシソーラスの作成時、シソーラスは、デフォルトでは大 / 小文字を区別せずに作成されます。

また、大 / 小文字を区別しないシソーラスを問合せで指定した場合、シソーラス検索では、問合せ語句はすべて大文字に変換されます。その結果、Oracle Text では、大 / 小文字の混合で表記されたときに異なる意味を持つ語句を区別できません。

たとえば、大 / 小文字を区別しないシソーラスを作成する場合、2つの異なる意味を持つ語句 *TURKEY*（国名と鳥の種類）には、異なるエントリが使用されます。このシソーラスを使用すると、*Turkey* または *turkey* のいずれに対する問合せも、シソーラス検索では *TURKEY* に変換された後、2つの意味に対応付けられているすべてのエントリを含むように拡張されます。

デフォルトのシソーラス

問合せでシソーラス名を指定しない場合、デフォルトで、シソーラス演算子は、*DEFAULT* という名前のシソーラスを使用します。ただし、Oracle Text には、デフォルトのシソーラスはありません。

したがって、シソーラス演算子に対してデフォルトのシソーラスを使用する場合は、*DEFAULT* という名前のシソーラスを作成する必要があります。このシソーラスの作成には、Oracle Text がサポートしている次のシソーラス作成方法を使用できます。

- CTX_THES.CREATE_THESAURUS (PL/SQL)
- ctxload

関連項目： ctxload および CTX_THES パッケージの使用方法的詳細は、『Oracle Text リファレンス』を参照してください。

提供されるシソーラス

Oracle Text には、デフォルトのシソーラスはありません。ただし、Oracle Text では、シソーラスを ctxload ロード・ファイルの形式で提供しています。このファイルを使用して汎用の英語シソーラスを作成できます。

このシソーラス・ロード・ファイルを使用すると、Oracle Text のデフォルトのシソーラスを作成できます。あるいは、このファイルを特定の主題または主題の範囲にあわせたシソーラスを作成するための基礎として使用することもできます。

関連項目： ctxload および CTX_THES パッケージの使用方法的詳細は、『Oracle Text リファレンス』を参照してください。

提供されるシソーラスの構造と内容

提供されるシソーラスは、Roget's Thesaurus（ロジェの分類語彙辞典）のような従来のシソーラスに類似しています。同義の語句や語義的に関連する語句のリストを備えています。

提供されるシソーラスは語句を階層形式に編成し、下位語とその上位語との実世界での実用的な関係を定義して付加価値を付けています。

また、階層の異なる領域の語句間で相互参照もできます。

提供されるシソーラスの場所

このシソーラス・ロード・ファイルの正確な名前と場所は、オペレーティング・システム固有です。ただし、通常ファイル名は、dr0thsus（適切なテキスト・ファイルの拡張子が付きます）で、ファイルは次のディレクトリ構造内に置かれます。

```
<Oracle_home_directory>
  <interMedia_Text_directory>
    sample
      thes
```

関連項目： Oracle Text のディレクトリ構造の詳細は、使用しているオペレーティング・システム固有の Oracle9i のインストール・マニュアルを参照してください。

シソーラスの用語の定義

シソーラスを使用して、シノニム、関連語および階層関係を作成できます。次の各項で、例を示します。

シノニムの定義

コンピュータ・サイエンス用語のシソーラスがあるとします。語句 *XML* のシノニムを *Extensible Markup Language* と定義します。これによって、この語句のいずれを問い合せても、同じドキュメントが戻ります。

```
XML
  SYN Extensible Markup Language
```

次のように SYN 演算子を使用すると、XML をそのシノニムにまで拡張できます。

```
'SYN(XML)'
```

次のように拡張されます。

```
'XML, Extensible Markup Language'
```


階層関係の定義

ドキュメント・セットがニュース記事で構成されている場合は、シソーラスを使用して、地理用語の階層を定義できます。たとえば、米国のカリフォルニア州の地理階層を示す次の階層があるとします。

```
California
  NT Northern California
    NT San Francisco
    NT San Jose
  NT Central Valley
    NT Fresno
  NT Southern California
    NT Los Angeles
```

NT 演算子を使用すると、カリフォルニアに対する問合せを次のように拡張できます。

```
'NT(California)'
```

次のように拡張されます。

```
'California, Northern California, San Francisco, San Jose, Central Valley, Fresno,
Southern California, Los Angeles'
```

結果のヒットリストには、米国のカリフォルニア州の地域や都市に関連するすべてのドキュメントが表示されます。

問合せアプリケーションでのシソーラスの使用

カスタム・シソーラスを定義すると、問合せをよりインテリジェントに処理できます。使用しているアプリケーションのユーザーには、トピックを表現するワードがわからない場合があるため、予想される問合せ語句にシノニムまたは下位語を定義できます。シソーラス演算子を使用すると、問合せをシソーラス語句に拡張できます。

カスタム・シソーラスを使用して、問合せをよりインテリジェントに処理できるように問合せアプリケーションを拡張するには、次の 2 つの方法があります。

- カスタム・シソーラスをロードし、シソーラス演算子を使用して問合せを発行します。
- カスタム・シソーラスを使用してナレッジ・ベースを補強し（英語のみ）、ABOUT 演算子を使用して問合せを拡張します。

どちらのアプローチにも、それぞれメリットとデメリットがあります。

カスタム・シソーラスのロードおよびシソーラス問合せの発行

カスタム・シソーラスを作成する手順は、次のとおりです。

1. シソーラスを作成します。この章の「[シソーラスの用語の定義](#)」を参照してください。
2. `ctxload` を使用してシソーラスをロードします。たとえば、次の例では、`tech_doc` というシソーラスを `tech_thesaurus.txt` というインポート・ファイルからインポートします。

```
ctxload -user jsmith/123abc -thes -name tech_doc -file tech_thesaurus.txt
```

3. シソーラス演算子を使用して問合せを行います。たとえば次のようにすると、XML と、`tech_doc` で定義した XML のシノニムを含むすべてのドキュメントを検索できます。

```
'SYN(XML, tech_doc)'
```

メリット

この方法を使用するメリットは、索引付け後にシソーラスを変更できることです。

制限事項

この方法では、問合せでシソーラス拡張演算子を使用する必要があります。問合せが長いと、シソーラスの拡張で余分なオーバーヘッドが発生し、問合せ速度が遅くなります。

カスタム・シソーラスによるナレッジ・ベースの補強

カスタム・シソーラスを既存のナレッジ・ベースのブランチに追加できます。ナレッジ・ベースは、テーマの索引付け、ABOUT 問合せおよびドキュメント・サービスでのドキュメント・テーマの抽出などに使用する概念の階層ツリーです。

既存のナレッジ・ベースを新しいシソーラスで補強する場合は、シノニムや下位語に暗黙的に拡張する ABOUT 演算子を使用して問い合わせます。この問合せにシソーラス演算子は使用しません。

カスタム・シソーラスで既存のナレッジ・ベースを補強する手順は、次のとおりです。

1. カスタム・シソーラスを作成し、新規語句を既存のナレッジ・ベースの語句にリンクします。この章の「[シソーラスの用語の定義](#)」および「[新規語句の既存語句へのリンク](#)」を参照してください。
2. `ctxload` を使用してシソーラスをロードします。この章の「[ctxload によるシソーラスのロード](#)」を参照してください。
3. ロードしたシソーラスを `ctxkbtcc` コンパイラを使用してコンパイルします。この項の後述の「[ロード済みのシソーラスのコンパイル](#)」を参照してください。
4. ドキュメントを索引付けします。デフォルトでは、システムが索引のテーマ・コンポーネントを作成します。

5. ABOUT 演算子を使用して問合せを行います。たとえば、語句 **politics** に関連するシノニムまたは下位語を含むすべてのドキュメントを検索するには、次の問合せを発行します。

```
'about (politics)'
```

メリット

カスタム・シソーラスを索引付けの前に既存のナレッジ・ベースにコンパイルすると、ABOUT 演算子を使用することにより問合せが高速かつ単純になります。また、ドキュメント・サービスで、テーマ・サマリーと要旨を作成する際にカスタマイズされた情報を利用できます。

制限事項

ABOUT 演算子を使用するには、索引内にテーマ・コンポーネントが存在していることが必要です。この場合、必要なディスク領域が多少増加します。ドキュメントを索引付けする前にシソーラスを定義する必要があります。シソーラスを変更した場合は、シソーラスを再コンパイルし、ドキュメントを再度索引付けする必要があります。

新規語句の既存語句へのリンク

ナレッジ・ベースに語句を追加する場合は、テーマの検証で最適な結果が得られるように、新規語句をナレッジ・ベースのカテゴリの 1 つにリンクすることをお勧めします。

関連項目： 提供される英語のナレッジ・ベースの詳細は、『Oracle Text リファレンス』を参照してください。

新規語句が既存のカテゴリから完全に分離している場合は、新規語句から検証されるテーマが少なくなります。その結果、ABOUT 問合せの精度が低下し、要旨およびテーマのハイライト表示の品質も低下します。

既存語句を新規語句の上位語にすることによって、新規語句を既存語句にリンクします。

例：新規語句の既存語句へのリンク 医学用語の階層を含む医学シソーラス **medthes** を購入したとします。このシソーラスの 4 つの最上位語は、次のとおりです。

- Anesthesia and Analgesia（麻酔および無痛）
- Anti-Allergic and Respiratory System Agents（抗アレルギー薬および呼吸器系薬）
- Anti-Inflammatory Agents, Antirheumatic Agents, and Inflammation Mediators（抗炎症薬、抗リウマチ薬および炎症伝達物質）
- Antineoplastic and Immunosuppressive Agents（抗腫瘍薬および免疫抑制薬）

これらの語句を既存のナレッジ・ベースにリンクするには、医学シソーラスに次のエントリを追加して、新規語句を既存の *health and medicine* ブランチにマップします。

```
health and medicine
  NT Anesthesia and Analgesia
  NT Anti-Allergic and Respiratory System Agents
  NT Anti-Inflamammatory Agents, Antirheumatic Agents, and Inflammation Mediators
  NT Antineoplastic and Immunosuppressive Agents
```

ctxload によるシソーラスのロード

医学シソーラスが med.thes というファイルにあるとします。次のように、ctxload を使用して、シソーラスを medthes としてロードします。

```
ctxload -thes -thescase y -name medthes -file med.thes -user ctxsys/ctxsys
```

ロード済みのシソーラスのコンパイル

ロードされたシソーラス medthes をナレッジ・ベースにリンクするには、次のように ctxkbtcc を使用します。

```
ctxkbtcc -user ctxsys/ctxsys -name medthes
```

提供されるナレッジ・ベース

Oracle Text では、英語とフランス語のナレッジ・ベースを提供します。提供されるナレッジ・ベースには、テーマ分析の実行に使用する情報が含まれています。テーマ分析には、テーマの索引付け、ABOUT 問合せおよび CTX_DOC パッケージによるテーマの抽出が含まれます。

ナレッジ・ベースは、概念とカテゴリの階層ツリーです。次の 6 つの主要ブランチがあります。

- science and technology
- business and economics
- government and military
- social environment
- geography
- abstract ideas and concepts

関連項目： カテゴリ階層の分類は、『Oracle Text リファレンス』を参照してください。

提供されるナレッジ・ベースは階層形式で、上位語、下位語および関連語情報が含まれています。この点で、シソーラスに似ています。したがって、新規語句を既存語句にリンクして業界固有のシソーラスで既存のナレッジ・ベースを補強することで、テーマ分析の精度を向上させることができます。

関連項目： この章の「[カスタム・シソーラスによるナレッジ・ベースの補強](#)」を参照してください。

また、言語固有のシソーラスをナレッジ・ベースにコンパイルすると、テーマ機能を別の言語に拡張できます。

関連項目： この章の「[言語固有のナレッジ・ベースの追加](#)」を参照してください。

ナレッジ・ベースのキャラクタ・セット

ナレッジ・ベースは、任意のシングルバイト・キャラクタ・セットで格納できます。提供されるナレッジ・ベースは、キャラクタ・セット WE8ISO8859P1 を使用しています。拡張ナレッジ・ベースは、US7ASCII などの別のキャラクタ・セットで格納できます。

言語固有のナレッジ・ベースの追加

シングルバイトの空白で区切られた言語の独自のナレッジ・ベースをロードすると、スペイン語など、英語やフランス語以外の言語にテーマ機能を拡張できます。

テーマ機能には、テーマの索引付け、ABOUT 問合せ、テーマのハイライト表示および CTX_DOC を使用したテーマ、要旨およびテーマ・サマリーの生成などの機能が含まれます。

テーマ機能は、ユーザー定義ナレッジ・ベースを追加して拡張します。たとえば、スペイン語シソーラスからスペイン語のナレッジ・ベースを作成できます。

言語固有のナレッジ・ベースをロードする手順は、次のとおりです。

1. ユーザー定義のカスタム・シソーラスを `ctxload` を使用してロードします。
2. 言語部分がターゲット言語になるように、`NLS_LANG` を設定します。キャラクタ・セット部分は、シングルバイト・キャラクタ・セットに設定する必要があります。
3. ロード済みのシソーラスを `ctxkbtc` を使用して次のようにコンパイルします。

```
ctxkbtc -user ctxsys/ctxsys -name my_lang_thes
```

このコマンドは、言語固有のナレッジ・ベースをロード済みのシソーラスからコンパイルします。索引付けおよび ABOUT 問合せ時のテーマ分析にこのナレッジ・ベースを使用するには、`NLS_LANG` 言語を `BASIC_LEXER` プリファレンスの `THEME_LANGUAGE` 属性として指定します。

制限事項

ナレッジ・ベースの追加には、次の制限事項が適用されます。

- Oracle が提供するナレッジ・ベースは、英語とフランス語のみです。それ以外の言語に対しては、独自のシソーラスを用意する必要があります。
- ナレッジ・ベースは、シングルバイト・キャラクタ・セットを持つ言語に対してのみ追加できます。マルチバイト・キャラクタ・セットのみで表現される言語に対して、ナレッジ・ベースを作成することはできません。データベースが、UTF-8 などのマルチバイト・ユニバーサル・キャラクタ・セットの場合は、そのシソーラスのコンパイル時に、NLS_LANG パラメータを互換性のあるシングルバイト・キャラクタ・セットに設定する必要があります。
- ナレッジ・ベースの追加によって、空白で区切られた言語が最適に機能します。
- NLS 言語ごとに 1 つのナレッジ・ベースを持つことができます。
- 上位語、下位語および関連語などの階層問合せフィードバック情報の取得操作は、英語とフランス語以外の言語では機能しません。その他の言語は、ナレッジ・ベースがユーザーのシソーラスからのみ導出されます。したがって、使用しているシソーラスから直接階層情報を取得することをお勧めします。

関連項目： テーマの索引付け、ABOUT 問合せ、CTX_DOC パッケージの使用
方法および提供される英語のナレッジ・ベースの詳細は、『Oracle Text
リファレンス』を参照してください。

この章では、Oracle Text の管理について説明します。次の項目について説明します。

- [Oracle Text のユーザーとロール](#)
- [DML キュー](#)
- [CTX_OUTPUT パッケージ](#)
- [サーバー](#)
- [管理ツール](#)

Oracle Text のユーザーとロール

すべてのユーザーが Oracle Text の索引を作成して CONTAINS 問合せを発行できる一方、Oracle Text では、管理用の CTXSYS ユーザーおよびアプリケーション開発者用の CTXAPP ロールを提供しています。

CTXSYS ユーザー

CTXSYS ユーザーは、インストール時に作成されます。管理者は、CTXSYS ユーザーで Oracle Text のユーザーを管理します。

CTXSYS ユーザーは、次の作業を実行できます。

- システム定義プリファレンスの変更
- 他のユーザーのプリファレンスの削除および変更
- PL/SQL パッケージ CTX_ADM のプロシージャ・コールによるシステム・パラメータの設定
- すべてのシステム定義ビューへの問合せ
- CTXAPP ロールを持つユーザーのすべての作業の実行

CTXAPP ロール

CTXAPP ロールは、システム定義のロールです。ユーザーは次の作業を実行できます。

- Oracle Text のプリファレンスの作成および削除
- Oracle Text の PL/SQL パッケージの使用

すべてのユーザーは、Oracle Text の索引の作成およびテキスト問合せの発行を行うことができます。CTXAPP ロールによって、ユーザーはプリファレンスを作成し、PL/SQL パッケージを使用できます。

ユーザーへのロールおよび権限の付与

システムでは、標準 SQL モデルを使用してロールをユーザーに付与します。テキスト・ロールをユーザーに付与するには、GRANT 文を使用します。

また、アプリケーション開発者に Oracle Text の PL/SQL パッケージにあるプロシージャをコールする許可を与えるには、各ユーザーに対して Oracle Text のパッケージの EXECUTE 権限を明示的に付与する必要があります。

DML キュー

元表のドキュメントに対して挿入、更新または削除がある場合、DML キューでは、索引付けを待機しているドキュメントの要求を格納します。CTX_DDL.SYNC_INDEX を使用して索引を同期化すると、その要求はキューから削除されます。

保留中の DML 要求は、CTX_PENDING ビューと CTX_USER_PENDING ビューを使用して問い合わせることができます。

DML エラーは、CTX_INDEX_ERRORS ビューまたは CTX_USER_INDEX_ERRORS ビューを使用して問い合わせることができます。

関連項目： これらのビューの詳細は、『Oracle Text リファレンス』を参照してください。

CTX_OUTPUT パッケージ

索引付け要求およびドキュメント・サービス要求のロギングには、PL/SQL パッケージ CTX_OUTPUT を使用します。

関連項目： このパッケージの詳細は、『Oracle Text リファレンス』を参照してください。

サーバー

ドキュメントの索引付けと問合せの発行には、標準 SQL を使用します。バッチ DML の実行にサーバーは不要です。CONTEXT 索引は、CTX_DDL.SYNC_INDEX プロシージャを使用して同期化できます。

関連項目： 索引付けおよび索引の同期化の詳細は、[第 2 章「索引付け」](#)を参照してください。

管理ツール

Oracle Text Manager は、Oracle Enterprise Manager と統合された Java アプリケーションで、別 CD で用意されています。

Text Manager を使用して、管理者は、プリファレンス、ストップリスト、セクションおよび索引を作成できます。また、このツールによって、管理者は DML を実行できます。

関連項目： Oracle Text Manager の詳細は、このツールで提供されるオンライン・ヘルプを参照してください。

CONTEXT 問合せアプリケーション

この付録では、CONTEXT 索引タイプを使用した、簡単な Web 検索アプリケーションの作成方法を説明します。次の項目について説明します。

- [Web 問合せアプリケーションの概要](#)
- [PSP Web アプリケーション](#)
- [JSP Web アプリケーション](#)

Web 問合せアプリケーションの概要

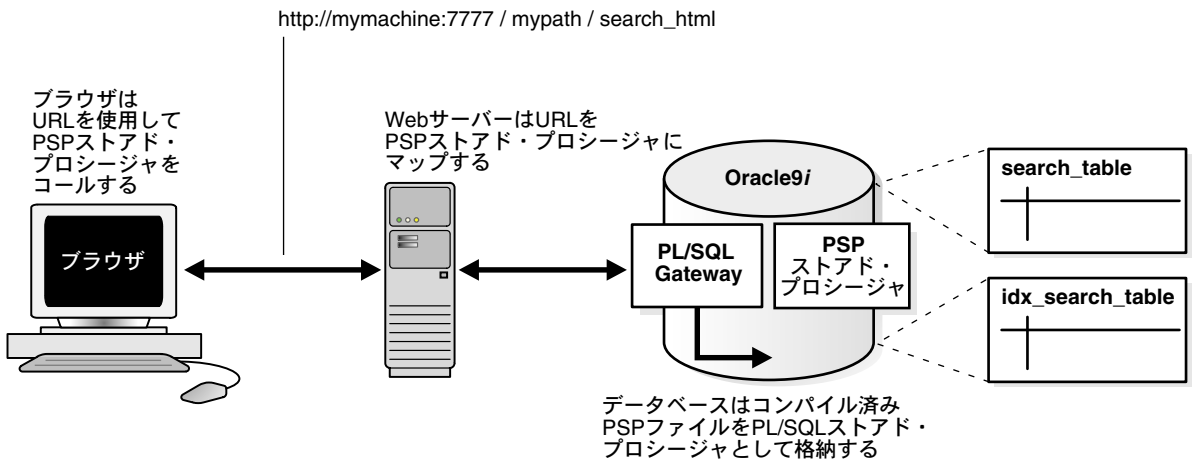
Oracle Text の一般的な使用法は、Web サイト上の HTML ファイルを索引付けし、ユーザーに検索機能を提供することです。この付録のサンプル・アプリケーションでは、データベースに格納されている HTML ファイルのセットを索引付けして、Oracle に接続している Web サーバーを使用して検索サービスを提供します。

このアプリケーションには、2 つのバージョンがあります。PL/SQL Server Pages (PSP) を使用するバージョンと JavaServer Pages (JSP) を使用するバージョンです。この付録では、両方のバージョンについて説明します。

PSP Web アプリケーション

このアプリケーションは、PL/SQL Server Pages に基づいています。図 A-1 は、ブラウザが、Web サーバーを経由して Oracle9i の PSP ストアド・プロシージャをコールする方法を示しています。

図 A-1



Web アプリケーションの前提条件

このアプリケーションには、次の要件があります。

- Oracle データベース（リリース 8.1.6 以上）が起動され、実行中であること。
- Oracle PL/SQL Gateway が実行中であること。
- Apache などの Web サーバーが起動され、実行中であり、Oracle9i サーバーに要求を送信するように適切に構成されていること。

Web アプリケーションの作成

この項では、PSP Web アプリケーションの作成方法を説明します。

手順 1 テキスト表の作成

HTML ファイルを格納するテキスト表を作成する必要があります。この例では、次のように `search_table` という表を作成します。

```
create table search_table (tk numeric primary key, title varchar2(2000), text clob);
```

手順 2 HTML ドキュメントのテキスト表へのロード（SQL*Loader を使用）

テキスト表は、HTML ファイルを使用してロードする必要があります。この例では、制御ファイル `loader.ctl` を使用して、`loader.dat` で指定したファイルをロードします。SQL*Loader のコマンドは、次のとおりです。

```
% sqlldr userid=scott/tiger control=loader.ctl
```

手順 3 CONTEXT 索引の作成

HTML ファイルを索引付けするには、次のように CONTEXT 索引をテキスト列に作成します。HTML の索引付けであるため、この例では、フィルタ処理が不要な `NULL_FILTER` プリファレンス型と `HTML_SECTION_GROUP` 型を使用します。

```
create index idx_search_table on search_table(text)
  indextype is ctxsys.context parameters
  ('filter ctxsys.null_filter section group CTXSYS.HTML_SECTION_GROUP');
```

手順 4 Oracle9i 上で search_htmlservices パッケージのコンパイル

アプリケーションには、選択したドキュメントが表示されることが必要です。そのためには、`search_table` の CLOB からドキュメントを読み込み、その結果を出力して表示する必要があります。これは、`search_htmlservices` パッケージのプロシージャをコールして実行します。ファイル `search_htmlservices.sql` をコンパイルする必要があります。これは、SQL*Plus のプロンプトで、次のように実行できます。

```
SQL> @search_htmlservices.sql
```

```
Package created.
```


手順 5 search_html PSP ページのコンパイル (loadpsp を使用)

検索ページを呼び出すには、ブラウザから [search_html.psp](#) をコールします。search_html は、Oracle9i の loadpsp コマンドライン・プログラムを使用して、次のようにコンパイルします。

```
% loadpsp -replace -user scott/tiger search_html.psp
"search_html.psp": procedure "search_html" created.
```

関連項目： PSP の使用方法の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

手順 6 Web サーバーの構成

クライアントの PSP 要求を URL として受け入れるように、Web サーバーを構成する必要があります。Web サーバーは、クライアントの要求を Oracle9i サーバーに転送し、サーバー出力をブラウザに戻します。[図 A-1](#) を参照してください。

Oracle WebDB 2.x Web リスナー、または Apache Web サーバーが組み込まれている Oracle iAS を使用できます。詳細は、使用している Web サーバーのマニュアルを参照してください。

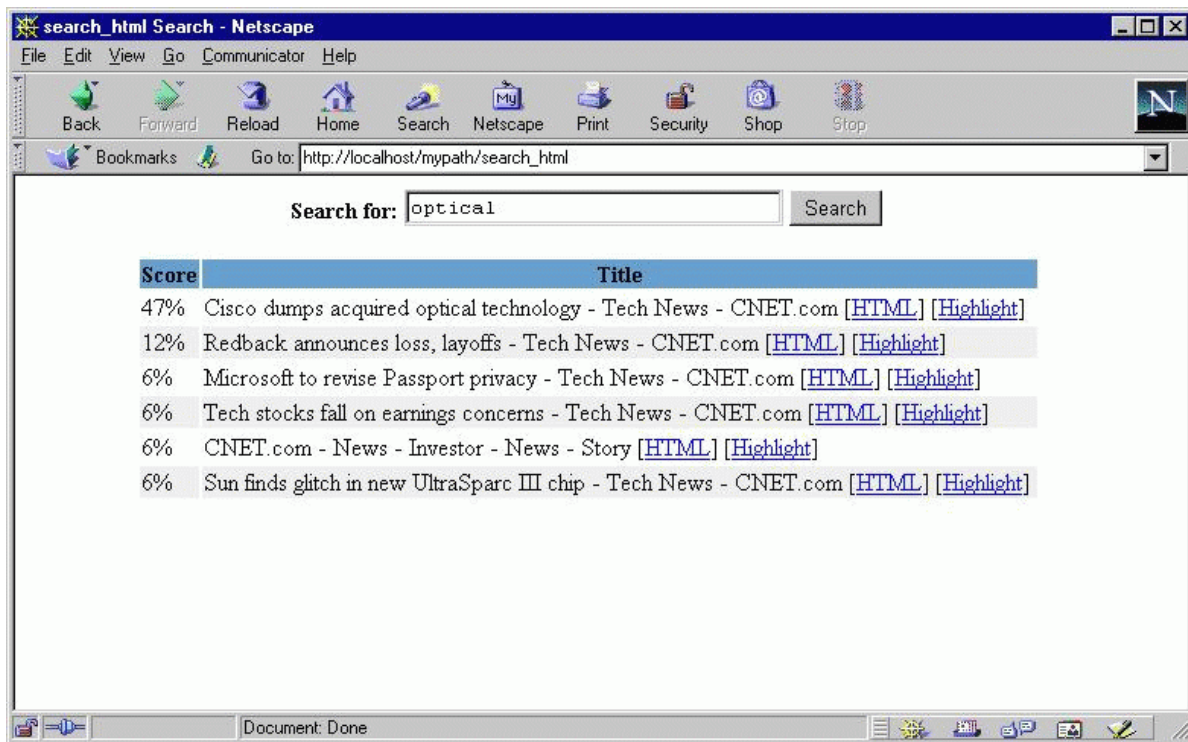
手順 7 ブラウザからの問合せの発行

URL を使用して、ブラウザから問合せアプリケーションにアクセスできます。URL を Web サーバーで構成します。URL の例は、次のようになります。

```
http://mymachine:7777/mypath/search_html
```

アプリケーションでは、ブラウザに問合せのエントリ・ボックスを表示し、問合せ結果を HTML リンクのリストとして戻します。[図 A-2 「Web 問合せアプリケーションの画面」](#) を参照してください。

図 A-2 Web 問合せアプリケーションの画面



PSP のサンプル・コード

この項では、Web アプリケーション・サンプルの作成に使用するコードを記載します。次のファイルのコードを記載します。

- [loader.cti](#)
- [loader.dat](#)
- [search_htmlservices.sql](#)
- [search_html.psp](#)

loader.ctl

```
LOAD DATA
  INFILE 'loader.dat'
  INTO TABLE search_table
  REPLACE
  FIELDS TERMINATED BY ','
  (tk          INTEGER,
   title       CHAR,
   text_file   FILLER CHAR,
   text        LOBFILE(text_file) TERMINATED BY EOF)
```

loader.dat

```
1; Sun finds glitch in new UltraSparc III chip;0-1003-200-5507959.html
2; Redback announces loss, layoffs ;0-1004-200-5424681.html
3; Cisco dumps acquired optical technology ;0-1004-200-5510096.html
4; Microsoft to revise Passport privacy ;0-1005-200-5508903.html
5; Tech stocks fall on earnings concerns;0-1007-200-5506210.html
6; CNET.com - News - Investor - News - Story ;0-9900-1028-5510548-0.html
7; Chicago Tribune JUSTICES HEAR ARGUMENTS ;0_2669_SAV-0103290318_FF.html
8; Massive new effort to combat African AIDS is planned ;WEST04.html
9; U.S. Had Biggest Growth in 1990s ;census_2000.html
10; Congress Discusses Napster Issues ;congress_napster.html
11; Washington And China Face Off in Spy Plane Drama ;crash_china_dc_35.html
12; American Arrive To Study in Cuba ;cuba_us_medical_students_1.html
13; Hubble Spots Most-Distant Supernova ;distant_supernova.html
14; Survey: U.S. Has 90 Percent Chance of Recession;economy_forecast_dc_1.html
15; House Votes To Repeal Estate Tax ;estate_tax.html
16; EU Condemns Bush on Global Warming ;eu_global_warming.html
17; Foot-and-Mouth Vaccinations on Hold ;foot_and_mouth.html
18; Foot-and-Mouth Vaccinations on Hold ;foot_and_mouth_7.html
19; Cancer Research Project Links Millions of PCs ;health_cancer_dc_1.html
20; Company Says Early HIV Vaccine Data Are Promising ;hiv.html
21; Yahoo! Sports: SOW - Maradona Faces New Paternity Suit ;maradona.html
22; Israel, Palestinians Hold High-Level Talks ;mideast_leadall_dc.html
23; Evidence Mounts Against Milosevic ;milosevic_slain_rivals.html
24; Philippines Files Charges Against Estrada ;philippines_estrada_dc.html
25; Power Woes Affecting Calif. Economy ;power_woes.html
26; Dissidents Ask UN Rights Body to Condemn China ;rights_china_dc_2.html
27; South Africa to Act on Basis HIV Causes AIDS ;safrica_aids_dc_1.html
28; Shaggy Found Inspiration For Success In Jamaica ;shaggy_found.html
29; Solar Flare Eruptions Likely ;solar_flare.html
30; Plane Crash Kills Sudanese Officers ;sudan_plane_crash.html
31; SOUNDSCAN REPORT: Recipe for An Aspiring Top Ten;urban_groove_1.html
```


search_htmlservices.sql

```
set define off

create or replace package search_htmlServices as

    procedure showHTMLDoc (p_id in numeric);

    procedure showDoc (p_id in numeric, p_query in varchar2);

end;
/
show errors;

create or replace package body search_htmlServices as

    procedure showHTMLDoc (p_id in numeric) is
        v_clob_selected    CLOB;
        v_read_amount      integer;
        v_read_offset      integer;
        v_buffer            varchar2(32767);
    begin

        select text into v_clob_selected from search_table where tk = p_id;
        v_read_amount := 32767;
        v_read_offset := 1;
    begin
        loop
            dbms_lob.read(v_clob_selected,v_read_amount,v_read_offset,v_buffer);
            http.print(v_buffer);
            v_read_offset := v_read_offset + v_read_amount;
            v_read_amount := 32767;
        end loop;
    exception
    when no_data_found then
        null;
    end;
end showHTMLDoc;

    procedure showDoc (p_id in numeric, p_query in varchar2) is

        v_clob_selected    CLOB;
        v_read_amount      integer;
        v_read_offset      integer;
```



```
v_buffer          varchar2(32767);
v_query           varchar(2000);
v_cursor          integer;

begin
  http.p('<html><title>HTML version with highlighted terms</title>');
  http.p('<body bgcolor="#ffffff">');
  http.p('<b>HTML version with highlighted terms</b>');

  begin
    ctx_doc.markup (index_name => 'idx_search_table',
                    textkey    => p_id,
                    text_query => p_query,
                    restab     => v_clob_selected,
                    starttag   => '<i><font color=red>',
                    endtag     => '</font></i>');

    v_read_amount := 32767;
    v_read_offset := 1;
    begin
      loop
        dbms_lob.read(v_clob_selected,v_read_amount,v_read_offset,v_buffer);
        http.print(v_buffer);
        v_read_offset := v_read_offset + v_read_amount;
        v_read_amount := 32767;
      end loop;
    exception
      when no_data_found then
        null;
      end;

    exception
      when others then
        null; --showHTMLdoc(p_id);
      end;
  end showDoc;
end;
/
show errors

set define on
```


search_html.psp

[illegible]


```
        select /*+ FIRST_ROWS */ rowid, tk, title, score(1) scr
        from search_table
        where contains(text, query,1) >0
        order by score(1) desc
    )
loop
    v_results := v_results + 1;
    if v_results = 1 then

%>

        <center>
        <table border="0">
            <tr bgcolor="#6699CC">
                <th>Score</th>
                <th>Title</th>
            </tr>

<%      end if; %>
        <tr bgcolor="#<%= color %>">
            <td> <%= doc.scr %> </td>
            <td> <%= doc.title %>
                [<a href="search_htmlServices.showHTMLDoc?p_id=<%= doc.tk %>">HTML</a>]
                [<a href="search_htmlServices.showDoc?p_id=<%= doc.tk %>&p_query=<%=
query %>">Highlight</a>]
            </td>
        </tr>

<%
        if (color = 'ffffff') then
            color := 'eeeeee';
        else
            color := 'ffffff';
        end if;

        end loop;
    %>

        </table>
        </center>

<%
    end if;
%>
</body></html>
```


JSP Web アプリケーション

この項では、JSP アプリケーションについて説明します。

Web アプリケーションの前提条件

このアプリケーションには、次の要件があります。

- Oracle データベース（リリース 8.1.6 以上）が起動され、実行中であること。
- Apache などの Web サーバーが起動され、実行中であり、Oracle9i サーバーに要求を送信するように適切に構成されていること。

JSP のサンプル・コード : search_html.jsp

```
<%@ page import="java.sql.* , oracle.jsp.dbutil.*" %>
<jsp:useBean id="name" class="oracle.jsp.jml.JmlString" scope="request" >
<jsp:setProperty name="name" property="value" param="query" />
</jsp:useBean>

<%
    String connStr="jdbc:oracle:thin:@localhost:1521:betadev";

    java.util.Properties info = new java.util.Properties();

    Connection conn = null;
    ResultSet rset = null;
    Statement stmt = null;

    if (name.isEmpty()) { %>

        <html>
        <title>search1 Search</title>
        <body>
        <center>
            <form method=post>
            Search for:
            <input type=text name=query size=30>
            <input type=submit value="Search">
            </form>
        </center>
        <hr>
        </body>
        </html>
```



```
<%
}
else {
%>

<html>
  <title>Search</title>
  <body>
    <center>
      <form method=post action="search_html.jsp">
        Search for:
        <input type=text name="query" value=<%= name.getValue() %> size=30>
        <input type=submit value="Search">
      </form>
    </center>

    <%
      try {

        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver() );
        info.put ("user", "ctxdemo");
        info.put ("password","ctxdemo");
        conn = DriverManager.getConnection(connStr,info);

        stmt = conn.createStatement();
        String theQuery = request.getParameter("query");

        String myQuery = "select /*+ FIRST_ROWS */ rowid, tk, title, score(1) scr
from search_table where contains(text, '"+theQuery+"' ,1 ) > 0 order by score(1)
desc";
        rset = stmt.executeQuery(myQuery);

        String color = "ffffff";
        int myTk = 0;
        String myTitle = null;
        int myScore = 0;
        int items = 0;
        while (rset.next()) {
          myTk = (int)rset.getInt(2);
          myTitle = (String)rset.getString(3);
          myScore = (int)rset.getInt(4);
          items++;

          if (items == 1) {
            %>
```



```
        <center>
            <table border="0">
                <tr bgcolor="#6699CC">
                    <th>Score</th>
                    <th>Title</th>
                </tr>
            <%    } %>

            <tr bgcolor="#<%= color %>">
                <td> <%= myScore %></td>
                <td> <%= myTitle %>
            </td>
            </tr>

        <%
            if (color.compareTo("ffffff") == 0)
                color = "eeeeee";
            else
                color = "ffffff";

        }
    } catch (SQLException e) {
        <%
            <b>Error: </b> <%= e %><p>
        <%
            } finally {
                if (conn != null) conn.close();
                if (stmt != null) stmt.close();
                if (rset != null) rset.close();
            }
        <%
            </table>
            </center>
            </body></html>
        <%
    }
    %>
```

CATSEARCH 問合せアプリケーション

この付録では、CATSEARCH 索引タイプを使用した、簡単な Web 検索アプリケーションの作成方法について説明します。次の項目について説明します。

- [CATSEARCH Web 問合せアプリケーションの概要](#)
- [JSP Web アプリケーション](#)

CATSEARCH Web 問合せアプリケーションの概要

CTXCAT 索引タイプは、短い説明テキストとそれに関連付けられた構造化データを含む商品カタログに最適です。この付録では、ユーザーが書籍のタイトルと価格を検索できるブラウザ・ベースの書店カタログの作成方法を説明します。

このアプリケーションは、JavaServer Pages (JSP) で作成されています。

JSP Web アプリケーション

このアプリケーションは Java Server Pages を基にしており、次の要件があります。

- Oracle データベース（リリース 8.1.7 以上）が起動され、実行中であること。
- Apache などの Web サーバーが起動され、実行中であり、Oracle9i サーバーに要求を送信するように適切に構成されていること。

JSP Web アプリケーションの作成

このアプリケーションは、書籍のタイトルと価格を検索できるオンライン書店をモデル化したものです。

手順 1 表の作成

書籍のタイトル、発行者、価格などの書籍情報を格納する表を作成する必要があります。SQL*Plus で次を実行します。

```
sqlplus>create table book_catalog (  
        id          numeric,  
        title       varchar2(80),  
        publisher   varchar2(25),  
        price       numeric );
```

手順 2 SQL*Loader を使用したデータのロード

書籍データは、オペレーティング・システムのコマンドラインから SQL*Loader を使用して、次のようにロードします。

```
sqlldr userid=ctxdemo/ctxdemo control=loader.ctl
```


手順 3 索引セットの作成

SQL*Plus から索引セットを次のように作成します。

```
sqlplus>begin
    ctx_ddl.create_index_set('bookset');
    ctx_ddl.add_index('bookset','price');
    ctx_ddl.add_index('bookset','publisher');
end;
/
```

手順 4 索引の作成

SQL*Plus から CTXCAT 索引を次のように作成します。

```
sqlplus>create index book_idx on book_catalog (title)
    indextype is ctxsys.ctxcat
    parameters('index set bookset');
```

手順 5 catsearch を使用した単純な検索

新しく作成した索引を SQL*Plus で次のようにテストします。

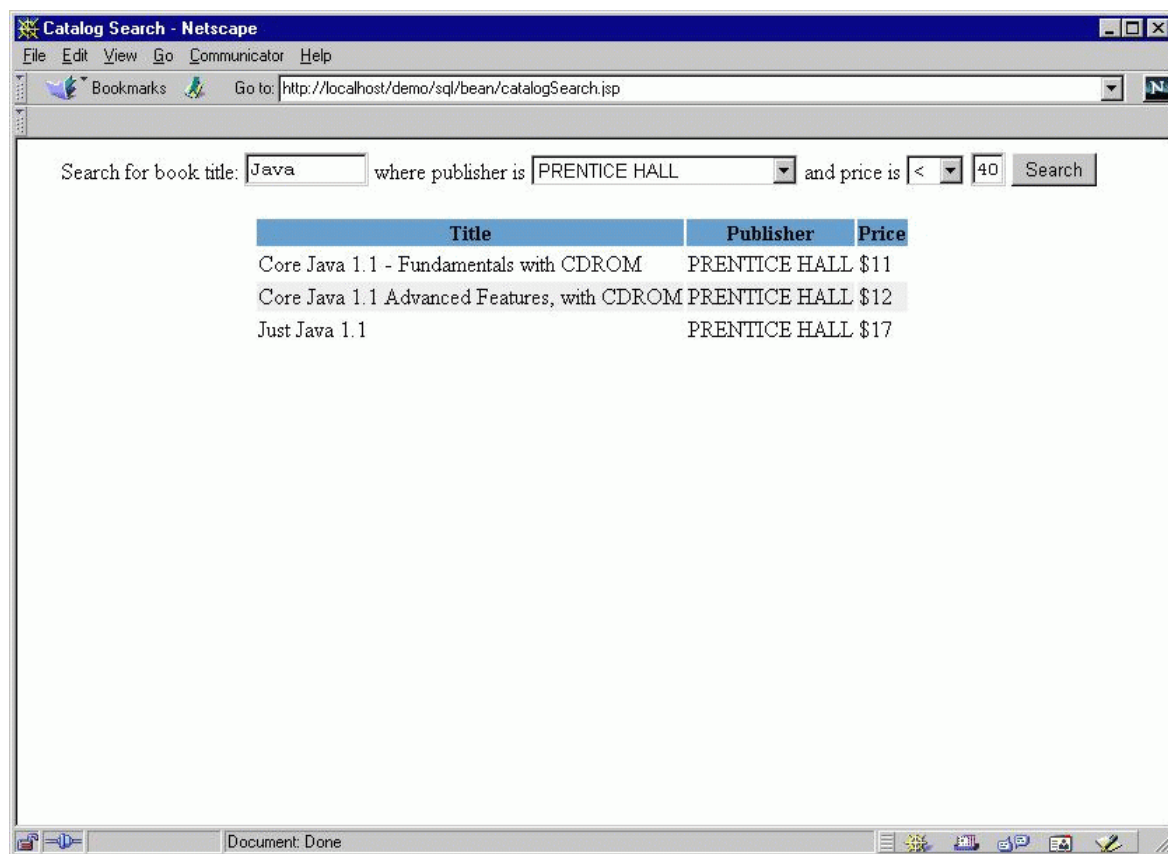
```
sqlplus>select id, title from book_catalog
    where catsearch(title,'Java','price > 10 order by price') > 0;
```

手順 6 Web サイトの jsp ディレクトリに対する catalogSearch.jsp ファイルのコピー

このコピーを行うと、ブラウザからアプリケーションにアクセスできます。URL は、`http://localhost:port/path/catalogSearch.jsp` を指定する必要があります。

アプリケーションがブラウザに問合せ入力ボックスを表示し、問合せ結果を HTML リンクのリストとして戻します。図 B-1「Web 問合せアプリケーションの画面」を参照してください。

図 B-1 Web 問合せアプリケーションの画面



JSP のサンプル・コード

この項では、Web アプリケーション・サンプルの作成に使用するコードを記載します。次のファイルのコードを記載します。

- [loader.ctl](#)
- [loader.dat](#)
- [catalogSearch.jsp](#)

loader.ctl

```
INFILE 'loader.dat'
  INTO TABLE book_catalog
  REPLACE
  FIELDS TERMINATED BY ';'
  (id, title, publisher, price)
```

loader.dat

```
1,A History of the Sciences, MACMILLAN REFERENCE,50
2,Robust Recipes Inspired by the Rustic Foods of France, Italy, and
America,MACMILLAN REFERENCE,28
3, Atlas of Irish History, MACMILLAN REFERENCE, 35
4, Bed and Breakfast Guide: Arizona, New Mexico and Texas, MACMILLAN REFERENCE, 37
5, Before You Say "I Quit"; A Guide to Making Successful Job Transitions, MACMILLAN
REFERENCE,25
6,Born to Shop Hong Kong; The Ultimate Travel Guide for Discriminating
Shoppers,MACMILLAN REFERENCE, 28
7,Complete Book of Sauces, MACMILLAN REFERENCE,16
8,Complete Idiot's Guide to American History,MACMILLAN REFERENCE, 28
9,Advanced Java Programming, with CD-ROM, MCGRAW HILL BOOK CO, 10
10, Java Master Reference With CDROM,IDG BOOKS WORLDWIDE,10
11, Oracle Performance Tuning Tips & Techniques, OSBORNE, 10
12, Core Java 1.1; Fundamentals, with CDROM, PRENTICE HALL, 11
13, Lady Oracle, DOUBLEDAY & CO 11
14, Core Java 1.1; Advanced Features, with CDROM, PRENTICE HALL, 12
15, Discover Java With Cd, IDG BOOKS WORLDWIDE, 12
16, CORBA & Java; Where Distributed Objects Meet the Web With CDROM, MCGRAW HILL
BOOK CO,13
17, Java 1.1 Developer's Handbook; With CDROM With CDROM, SYBEX INC, 13
18, Java with Borland C++,AP PROFESSIONAL, 13
19, Just Java 1.1, PRENTICE HALL, 17
20, Internet Programming; An Introduction to Object Oriented Programming with Java,
ADDISON WESLEY PUB CO INC, 14
```


21, Oracle Certified Professional DBA Certification Exam Guide With CDROM, OSBORNE, 14
22, Eye of Horus; An Oracle of Ancient Egypt, THOMAS DUNNE BOOKS, 15
23, Java 1.1 Certification Study Guide With CDROM, SYBEX INC, 15

catalogSearch.jsp

```
<%@ page import="java.sql.* , oracle.jsp.dbutil.*" %>
<jsp:useBean id="name" class="oracle.jsp.jml.JmlString" scope="request" >
<jsp:setProperty name="name" property="value" param="v_query" />
</jsp:useBean>

<%
    String connStr="jdbc:oracle:thin:@machine-domain-name:1521:betadev";

    java.util.Properties info = new java.util.Properties();

    Connection conn = null;
    ResultSet rset = null;
    Statement stmt = null;

    if (name.isEmpty() ) {

%>
        <html>
        <title>Catalog Search</title>
        <body>
        <center>
            <form method=post>
            Search for book title:
            <input type=text name="v_query" size=10>
            where publisher is
            <select name="v_publisher">
                <option value="ADDISON WESLEY">ADDISON WESLEY
                <option value="AP PROFESSIONAL">AP PROFESSIONAL
                <option value="DOUBLEDAY & CO">DOUBLEDAY & CO
                <option value="IDG BOOKS WORLDWIDE">IDG BOOKS WORLDWIDE
                <option value="MACMILLAN REFERENCE">MACMILLAN REFERENCE
                <option value="MCGRAW HILL BOOK CO">MCGRAW HILL BOOK CO
                <option value="OSBORNE">OSBORNE
                <option value="PRENTICE HALL">PRENTICE HALL
                <option value="SYBEX INC">SYBEX INC
                <option value="THOMAS DUNNE BOOKS">THOMAS DUNNE BOOKS
            </select>
```



```

        and price is
        <select name="v_op">
            <option value="">=
            <option value="&lt;">&lt;
            <option value="&gt;">&gt;
        </select>
        <input type="text" name="v_price" size=2>
        <input type="submit" value="Search">
    </form>
</center>
<hr>
</body>
</html>

<%
    }
    else {

        String v_query = request.getParameter("v_query");
        String v_publisher = request.getParameter("v_publisher");
        String v_price = request.getParameter("v_price");
        String v_op = request.getParameter("v_op");

    %>

    <html>
        <title>Catalog Search</title>
        <body>
            <center>
                <form method="post" action="catalogSearch.jsp">
                    Search for book title:
                    <input type="text" name="v_query" value=
                    <%= v_query %>
                    size=10>
                    where publisher is
                    <select name="v_publisher">
                        <option value="ADDISON WESLEY">ADDISON WESLEY
                        <option value="AP PROFESSIONAL">AP PROFESSIONAL
                        <option value="DOUBLEDAY & CO">DOUBLEDAY & CO
                        <option value="IDG BOOKS WORLDWIDE">IDG BOOKS WORLDWIDE
                        <option value="MACMILLAN REFERENCE">MACMILLAN REFERENCE
                        <option value="MCGRAW HILL BOOK CO">MCGRAW HILL BOOK CO
                        <option value="OSBORNE">OSBORNE
                        <option value="PRENTICE HALL">PRENTICE HALL
                        <option value="SYBEX INC">SYBEX INC
                        <option value="THOMAS DUNNE BOOKS">THOMAS DUNNE BOOKS
                    </select>
                    and price is

```



```
        <select name="v_op">
            <option value="">=<br>
            <option value="&lt;">&lt;<br>
            <option value="&gt;">&gt;<br>
        </select>
        <input type="text" name="v_price" value=
        <%= v_price %> size=2>
        <input type="submit" value="Search">
    </form>
</center>

<%
    try {

        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver() );
        info.put ("user", "ctxdemo");
        info.put ("password","ctxdemo");
        conn = DriverManager.getConnection(connStr,info);

        stmt = conn.createStatement();
        String theQuery = request.getParameter("v_query");
        String thePrice = request.getParameter("v_price");

        // select id,title
        // from book_catalog
        // where catsearch (title,'Java','price >10 order by price') > 0

        // select title
        // from book_catalog
        // where catsearch(title,'Java','publisher = 'PRENTICE HALL' and price < 40 order
        // by price' )>0

        String myQuery = "select title, publisher, price from book_catalog where
        catsearch(title, '"+theQuery+"', 'publisher = '"+v_publisher+"' and price "+v_
        op+thePrice+" order by price' ) > 0";
        rset = stmt.executeQuery(myQuery);

        String color = "ffffff";

        String myTitle = null;
        String myPublisher = null;
        int myPrice = 0;
        int items = 0;

        while (rset.next()) {
            myTitle      = (String)rset.getString(1);
            myPublisher = (String)rset.getString(2);
```



```

        myPrice      = (int)rset.getInt(3);
        items++;

        if (items == 1) {
%>
            <center>
                <table border="0">
                    <tr bgcolor="#6699CC">
                        <th>Title</th>
                        <th>Publisher</th>
                        <th>Price</th>
                    </tr>
%<
                }
%>
                <tr bgcolor="#<%= color %>">
                    <td> <%= myTitle %></td>
                    <td> <%= myPublisher %></td>
                    <td> $<%= myPrice %></td>
                </tr>
%<
                if (color.compareTo("ffffff") == 0)
                    color = "eeeeee";
                else
                    color = "ffffff";

            }

        } catch (SQLException e) {

%>

            <b>Error: </b> <%= e %><p>

%<

        } finally {
            if (conn != null) conn.close();
            if (stmt != null) stmt.close();
            if (rset != null) rset.close();
        }

%>
    </table>
    </center>
</body>
</html>

```



```
<%  
}  
%>
```


A

ABOUT 問合せ, 3-12
 大 / 小文字区別, 3-10
 言語への追加, 7-10
 定義, 3-8
 例, 1-16
ACCUM 演算子, 3-13
ADD_STOPCLASS プロシージャ, 2-27
ADD_STOPTHEME プロシージャ, 2-27
ADD_STOPWORD プロシージャ, 2-26, 2-27
ADD_SUB_LEXER プロシージャ
 例, 2-24
ALTER INDEX コマンド
 索引の再構築, 2-35
 失敗した索引の再開, 2-35
AND 演算子, 3-12
AUTO_SECTION_GROUP オブジェクト, 6-3

B

BASIC_LEXER, 2-13
BASIC_SECTION_GROUP オブジェクト, 6-2
BFILE 列, 1-8
 索引付け, 1-11, 2-28
BINARY
 形式列の値, 2-12
BLOB 列, 1-8
 索引付け, 1-11, 2-28

C

CATSEARCH, 3-4
 SQL 例, 3-4
 演算子, 3-16

 構造化問合せ, 3-4
 索引の作成, 2-31
CHARSET_FILTER, 2-5, 2-13
CHAR 列, 1-8
CHINESE_VGRAM_LEXER, 2-17
CLOB 列, 1-8
 索引付け, 1-11, 2-28
CONTAINS
 PL/SQL 例, 3-3
 SQL 例, 3-2
 演算子, 3-11
 構造化問合せ, 3-3
 問合せ, 3-2
CONTEXT 索引, 1-2
 HTML 例, 2-28, A-3
 概要, 1-10, 2-9
 カスタマイズ, 1-12
 作成, 1-11, 2-20, 2-27
CONTEXT 文法, 3-11
CREATE INDEX コマンド, 2-27
CREATE INDEX の paramstring, 2-27
CREATE_STOPLIST プロシージャ, 2-26, 2-27
CTX_CLS.TRAIN プロシージャ, 1-2
CTX_DDL.SYNC_INDEX プロシージャ, 2-37
CTX_DOC パッケージ, 4-2
CTX_INDEX_ERRORS ビュー, 2-34, 8-3
CTX_PENDING ビュー, 8-3
CTX_REPORT, 2-39
CTX_THES パッケージ
 概要, 7-2
CTX_USER_INDEX_ERRORS ビュー, 2-34, 8-3
CTX_USER_PENDING ビュー, 8-3
CTXAPP ロール, 8-2

CTXCAT 索引, 1-2, 1-12
 概要, 1-11, 2-9
 パフォーマンスについて, 5-18
 例, 2-29
CTXCAT 文法, 3-16
ctxkbtc
 例, 7-9
ctxload
 シソーラスのロード例, 7-2, 7-7, 7-9
CTXRULE 索引, 1-2, 1-13
 概要, 1-11, 2-10
 作成, 2-33
CTXSYS ユーザー, 8-2
CTXXPATh 索引
 概要, 2-10

D

DATE 列, 1-11, 2-28
DBMS_JOB.SUBMIT プロシージャ, 2-37
DEFAULT_INDEX_MEMORY, 5-19
DETAIL_DATASTORE, 1-7
 概要, 2-11
DIRECT_DATASTORE, 1-7
 概要, 2-11
 例, 2-22
DML
 保留中の表示, 2-36
DML キュー, 8-3
DML 処理, 1-13
 バックグラウンド, 8-3
DOMAIN_INDEX_NO_SORT ヒント
 スループットの向上例, 5-9
drjobdml.sql スクリプト, 2-37
DROP INDEX コマンド, 2-35
DROP_STOPLIST プロシージャ, 2-27

E

EQUIV 演算子, 3-13

F

FILE_DATASTORE, 2-4
 概要, 1-7, 2-11
 例, 2-22
FILTER プロシージャ, 4-3

FIRST_ROWS ヒント, 3-17
 応答時間の短縮例, 5-6
 スループットの向上例, 5-9
 例, 1-16
FUZZY 演算子, 3-14

G

GIST プロシージャ, 4-5

H

HASPATH 演算子, 6-14
 例, 6-16
HFEEDBACK プロシージャ, 3-10
HIGHLIGHT プロシージャ, 4-2
HTML
 META タグの検索, 6-11
 索引付け, 2-23, 6-2
 索引付けの例, A-3
 ゾーン・セクションの例, 2-25, 6-10
 フィルタ処理, 1-22, 4-3
HTML_SECTION_GROUP オブジェクト, 2-25, 6-2,
 6-10
 NULL_FILTER, 2-23, A-3

I

IGNORE
 形式列の値, 2-13
INDEX_STATS プロシージャ, 2-39
INPATH 演算子, 6-14
 例, 6-14
INSERT 文
 テキストのロード例, 1-8
INSO_FILTER, 2-4, 2-12, 2-13
INSO フィルタ, 5-20

J

JAPANESE_LEXER, 2-17
JSP Web アプリケーション, B-2

K

KOREAN_MORP_LEXER, 2-17

L

LOB 列

索引付け, 1-11, 2-28

問合せパフォーマンスの向上, 5-15

M

MARKUP プロシージャ, 4-3

MATCHES

PL/SQL 例, 2-34, 3-5

SQL 例, 3-5

概要, 3-5

MAX_INDEX_MEMORY, 5-19

META タグ

ゾーン・セクションの作成, 6-11

MULTI_COLUMN_DATASTORE, 1-7

概要, 2-11

例, 2-22

MULTI_LEXER, 2-14

例, 2-24

N

NCLOB 列, 1-11, 2-28

NEAR 演算子, 3-13

NESTED_DATASTORE, 1-7

概要, 2-11

NEWS_SECTION_GROUP オブジェクト, 6-3

NOT 演算子, 3-12

NULL_FILTER, 2-4

例, 2-23, A-3

NULL_SECTION_GROUP オブジェクト, 6-2

NUMBER 列, 1-11, 2-28

O

Oracle Enterprise Manager, 8-3

Oracle9i Text Manager, 8-3

OR 演算子, 3-12

P

PATH_SECTION_GROUP

例, 6-14

PL/SQL ファンクション

CONTAINS でのコール, 3-15

printjoin 文字, 2-14

PROCEDURE_FILTER, 2-12

PSP アプリケーション, A-2

R

REMOVE_SQE プロシージャ, 3-14

REMOVE_STOPCLASS プロシージャ, 2-27

REMOVE_STOPTHEME プロシージャ, 2-27

REMOVE_STOPWORD プロシージャ, 2-26, 2-27

S

SGA メモリー割当て, 5-19

skipjoin 文字, 2-14

SORT_AREA_SIZE, 5-11, 5-14, 5-19

SQE 演算子, 3-14

STEM 演算子, 2-18, 3-14

STORE_SQE プロシージャ, 3-14

SYNC_INDEX プロシージャ, 2-37

SYN 演算子, 7-5

T

TEXT

形式列の値, 2-12

Text Manager ツール, 8-3

THEMES プロシージャ, 4-4

TRAIN プロシージャ, 1-2

U

URL

格納, 1-7

URL_DATASTORE

概要, 2-11

例, 2-22

USER_DATASTORE, 2-8

概要, 2-11

USER_FILTER, 2-12

V

VARCHAR2 列, 1-8

W

WITHIN 演算子, 2-25

X

XML_SECTION_GROUP オブジェクト, 6-3

XML ドキュメント

索引付け, 6-3

セクション検索, 6-11

属性検索, 6-12

ドキュメント・タイプ別のセクション, 6-13

あ

アクセント

文字の索引付け, 2-16

アプリケーション

サンプル, A-1, B-1

う

ウムラウト

文字の索引付け, 2-16

え

エラー

DML, 8-3

表示, 2-34

演算子

CATSEARCH, 3-16

CONTAINS, 3-11

シソーラス, 7-2

論理, 3-12

お

応答時間

最適化, 1-16, 3-17

短縮, 5-4

大 / 小文字区別

ABOUT 問合せ, 3-10

索引付け, 2-15

シソーラス, 7-3

問合せ, 3-10

オフセット情報

ハイライト, 4-2

か

概念問合せ, 「ABOUT」を参照

各国語別の機能, 2-16

拡張可能問合せオブティマイザ, 5-2

ガベージ・コレクション, 2-39

韓国語の索引付け, 2-17

管理ツール, 8-3

き

記憶域

概要, 2-21

機能的検索, 5-13

基本文字変換, 2-16

キャラクタ・セット

索引付け, 2-13

複合の索引付け, 2-13

キャラクタ・セット列, 1-7, 2-13

キュー

DML, 8-3

行のバイパス, 2-13

く

句問合せ, 3-7

け

形式

サポートされている, 1-8

フィルタ処理, 2-12

形式列, 1-7, 2-12, 2-13

結果バッファ・サイズ

増加, 5-11

言語

索引付け, 2-13

索引付けのためのデフォルト設定, 1-12, 2-28

言語固有のナレッジ・ベース, 7-10

こ

構造化問合せ

例, 2-29

構造化フィールド

アプリケーションに表示, 1-21

構造化フィールドの検索

概要, 1-17

さ

索引

概要, 2-2

構造, 2-3, 2-38

再構築, 2-35

最適化, 2-38, 2-39

削除, 2-35

作成, 2-20, 2-27

同期化, 2-37, 8-3

複数, 2-8

索引エラー

表示, 2-34

索引エンジン

概要, 2-5

索引更新のパフォーマンス

FAQ, 5-22

索引タイプ

概要, 1-10

選択, 2-8

索引付き検索, 5-13

索引付け

概要, 1-9

行のバイパス, 2-13

考慮事項, 2-8

失敗した場合の再開, 2-35

処理の概要, 2-3

制限事項, 2-8

特殊文字, 2-14

パラレル, 2-7, 5-21

索引付けの時間, 5-19

索引付けのパフォーマンス

FAQ, 5-19

パラレル, 5-21

索引デフォルト

汎用, 1-11, 2-28

索引の構造, 2-38

索引の再構築, 2-35

索引の最適化, 2-38

単一のトークン, 2-39

例, 2-39

索引の削除, 2-35

索引の断片化, 2-38, 5-23

表示, 2-39

索引の同期化, 1-13, 2-37, 8-3

パフォーマンスの向上, 5-22

索引のメンテナンス, 1-13, 2-34

索引メモリー, 5-19

し

シソーラス

DEFAULT, 7-4

アプリケーションでの使用, 7-6

演算子, 7-2

大 / 小文字区別, 7-3

階層関係, 7-6

概要, 7-2

カスタムのロード, 7-7

提供される, 7-4

デフォルト, 7-4

ナレッジ・ベースに追加, 7-7

用語の定義, 7-5

シソーラス演算子, 3-14

シソーラスを使用した問合せ

概要, 1-17

実行計画, 3-11

失敗した索引の再開, 2-35

自動セクション, 6-12

シノニム

定義, 7-5

す

スコア

表示, 1-21

ステミング

デフォルト, 1-12, 2-28

パフォーマンスの向上, 5-16

ストアド・クエリー式, 3-14

ストップクラス, 2-26

ストップテーマ, 2-26

概要, 2-19

定義, 3-8

- ストップリスト, 2-26
 - PL/SQL プロシージャ, 2-27
 - 概要, 2-21
 - 管理ツールによる作成, 8-3
 - デフォルト, 1-12, 2-28
 - マルチ言語, 2-19, 2-26
- ストップワード, 2-26
 - 大 / 小文字区別, 3-10
 - 概要, 2-19, 3-7
- スペル
 - 代替, 2-16
- スループット
 - 問合せの向上, 5-9

せ

- セクション
 - 概要, 2-5
- セクション
 - HTML 例, 2-25
 - オーバーラップ, 6-6
 - 繰返しゾーン, 6-6
 - 自動, 6-12
 - ゾーン, 6-5
 - 属性, 6-8
 - 特殊, 6-9
 - ネストされた, 6-6
 - パス, 6-14
 - フィールド, 6-7
- セクション・グループ
 - 概要, 2-21
 - 管理ツールによる作成, 8-3
- セクション検索
 - HTML, 6-10
 - 概要, 1-17, 6-2
 - 使用可能化, 6-2
- 全テーマ
 - 取得, 4-4

そ

- ゾーン・セクション
 - オーバーラップ, 6-6
 - 繰返し, 6-6
 - 定義, 6-5
 - ネストされた, 6-6

- 属性
 - XML の検索, 6-12
- 属性セクション, 6-8

た

- 代替スペル, 2-16
- 単一テーマ
 - 取得, 4-4

ち

- 中国語の索引付け, 2-17

て

- ティルデ
 - 文字の索引付け, 2-16
- データ記憶域
 - 索引デフォルト, 1-11, 2-28
 - プリファレンス例, 2-22
- データストア
 - 概要, 2-4, 2-21
- テーマ
 - 索引付け, 2-16
- テーマ機能
 - 概要, 1-3
 - 追加, 7-10
- テーマ・サマリー
 - 定義, 4-3
- テーマ問合せ, 「ABOUT」を参照
- テーマのハイライト表示, 4-2
- テーマのリスト
 - 取得, 4-4
 - 定義, 4-3
- テキスト記憶域, 2-11
- テキスト問合せアプリケーション
 - 概要, 1-2
- テキストの格納, 2-11
 - 概要, 1-7
- テキストのハイライト表示, 4-2
- テキストの場所, 2-11
- テキストのロード
 - SQL の INSERT 例, 1-8
 - 概要, 1-5
- テキスト列
 - サポートされている型, 1-8

デフォルト
索引, 1-11, 2-28
デフォルトのシソーラス, 7-4
テンプレートによる問合せ, 3-14, 3-16

と

問合せ
ABOUT, 3-12
CATSEARCH, 3-4
CONTAINS, 3-2
MATCHES, 3-5
大 / 小文字区別, 3-10
概要, 1-15, 3-2
スループットの向上のための最適化, 5-9
パラレル, 5-10
ヒット数のカウント, 3-18
ブロック操作, 5-11
問合せアプリケーション
サンプル, 1-14
前提条件, 1-4
問合せ機能, 1-18
問合せ式, 3-8
問合せテンプレート, 3-14, 3-16
問合せの最適化, 3-17, 5-2
FAQ, 5-12
応答時間, 1-16, 5-4
スループット, 5-9
統計, 5-2
ブロック操作, 5-11
問合せの実行計画, 3-11
問合せのチューニング
応答時間, 1-16, 5-4
結果バッファ・サイズの増加, 5-11
スループットの向上, 5-9
統計の利用, 5-2
問合せのパフォーマンス
FAQ, 5-12
問合せのフィードバック, 3-10
問合せ例, 1-15
統計
最適化, 5-2
ドキュメント形式
索引パフォーマンスへの影響, 5-20
サポートされている, 1-3, 1-8
パフォーマンスへの影響, 5-13
フィルタ処理, 2-12

ドキュメント・サービス
概要, 1-22
ドキュメント・セクション, 2-25
ドキュメントのヒット件数
表示, 1-21
ドキュメントの表示方法
概要, 1-22
ドキュメントのフィルタ処理, 2-12
HTML およびプレーン・テキスト, 4-3
ドキュメントの無効性, 2-39
ドキュメントのロード
方法, 1-8
ドキュメント分類, 2-33
概要, 1-2
特殊セクション, 6-9
特殊文字
索引付け, 2-14

な

ナレッジ・ベース
概要, 7-9
サポートされているキャラクタ・セット, 7-10
新規語句のリンク, 7-8
補強, 7-7
ユーザー定義, 7-10

に

日本語の索引付け, 2-17

ね

ネストされたゾーン・セクション, 6-6

は

パーティション索引, 5-17
応答時間の短縮, 5-7
ハイライト表示
概要, 1-22, 4-2
パス・セクション検索, 6-14
発音区別記号
文字, 2-16
バックグラウンド DML, 8-3

パフォーマンス・チューニング

索引付け, 5-19

索引の更新, 5-22

問合せ, 5-12

パラレル索引付け, 2-7, 5-21

パーティション表, 5-21

パラレル問合せ, 5-10, 5-17

ひ

ヒット数, 3-18

ヒット数のカウント, 3-18

ヒットリスト

表示, 1-19

ヒットリストの表示, 1-19

ビュー

索引付け, 2-8

ビューの索引付け, 2-8

表外の LOB への格納

パフォーマンスの向上, 5-15

ふ

ファイル・パス

格納, 1-7

ファジー・マッチング, 2-18

デフォルト, 1-12, 2-28

フィードバック

問合せ, 3-10

フィールド・セクション

繰返し, 6-8

参照可能および参照不能, 6-7

定義, 6-7

ネストされた, 6-8

フィルタ

概要, 2-4, 2-21

フィルタ処理

カスタム, 2-12

索引デフォルト, 1-11, 2-28

ブレーン・テキストおよび HTML, 1-22

複合語

索引付け, 2-17

複合書式

フィルタ処理, 2-12

複数の CONTAINS

パフォーマンスの向上, 5-15

複数の索引, 2-8

プリファレンス

管理ツールによる作成, 8-3

削除, 2-36

作成 (例), 2-21

ブレーン・テキスト

NULL_FILTER による索引付け, 2-23

フィルタ処理, 4-3

ブレーン・テキストのフィルタ処理, 1-22

ブロック操作

問合せのチューニング, 5-11

文法

CONTEXT, 3-11

CTXCAT, 3-16

分類

概要, 1-2

ほ

保留中の DML

表示, 2-36

保留中の更新, 8-3

ま

マークアップされたドキュメント

取得, 4-3

マルチ言語のストップリスト

概要, 2-26

マルチ言語列

索引付け, 2-14

め

メモリー割当て

索引付け, 5-19

索引の同期化, 5-23

問合せ, 5-14

ゆ

ユーザー

システム定義, 8-2

よ

要旨

定義, 4-3

例, 4-5

れ

レクサー

概要, 2-5, 2-21

列型

索引付けにサポートされている, 1-8

ろ

ローカル・パーティション索引, 5-17

応答時間の短縮, 5-7

ロール

システム定義, 8-2

付与, 8-2

ロールの付与, 8-2

論理演算子, 3-12

わ

ワード問合せ, 3-7

大 / 小文字区別, 3-10

例, 1-15

ワードリスト

概要, 2-21

ワイルド・カード演算子, 3-14

パフォーマンスの向上, 5-16

