ORACLE

Oracle DatabaseのOracle Textの新しいユーザー・ガイド

テキスト・データの活用

2023年9月、バージョン1.0

ORACLE

目次

ORACLE

はじめに

私たちは皆、Googleなどの検索エンジンを使用してインターネットで単語ベースの検索を行うことに慣れています。しかし、多くの人は、Oracle Databaseにも同様の検索テクノロジーが組み込まれていることに気付いていません。

オラクルの統合型全文検索テクノロジーであるOracle Textは、Oracle Databaseのすべてのエディションに搭載されています。Oracle Textでは、Oracleデータベース内に保存されたテキストやドキュメントの索引作成、検索、分析に標準のSQLを使用します。

プレーン・テキストは通常、VARCHAR2列およびCLOB列に保存されていますが、BLOB列に保存されているWordファイルやPDFファイルなどのバイナリ・ドキュメントに索引を付けることもできます。また、データベースに保存されている外部ファイルへのポインタを使用するだけで、ファイル・システム、Web、またはクラウド・ストレージ上に保持されている外部ファイルに索引を付けることさえできます。

Oracle Textは、スペースで区切られた言語や、中国語、日本語、韓国語などのピクトグラム言語のすべてを含む、複数の言語とキャラクタ・セットをサポートしています。

テキスト索引に対する問合せを使用すると、カスタマイズ可能である高度な関連性ランキング・アルゴリズムを使用した、あいまい検索が可能になり、もっとも適切な結果を最初に返すことができます。

いくつかのユースケース

"典型的な" Oracle Textアプリケーションというようなものは存在しません。しかし、Oracle Textアプリケーションが実際のお客様によってどのように使用されているかを示す例を、ほんのいくつかですがご紹介します。

会社名の検索

購買システムには、サプライヤ会社のリストがあります。会社名は、短いVARCHAR2(80)フィールド内に保持されます。完全な社名は知らないものの、その社名に含まれる1、2単語なら知っているということがよくあります。社名フィールドのテキスト索引を作成すると、次のような検索を実行できるようになります。

- **部分的名前検索** - 社名に"Chemical"などの特定の単語が入っているすべての会社を見つけます。
- **ファジー一致** - "Philips"と"Phillips"のどちらなのかわからない場合、ファジー検索を実行して両方を見つけることができます。
- **語幹検索** - "chemical"、"chemicals"、"chemistry"など、語の先頭が"chem"であるすべての会社を見つけます。

ドキュメント管理

ドキュメント管理システムには、チェックイン/チェックアウト/ダウンロード機能があります。ドキュメントは、BLOB列または外部ファイル内に元のバイナリ形式で保存されます。表内のその他の列には、作成者、作成日、チェックアウト・ステータスなどの情報が含まれます。バイナリ・データに対して索引を作成すると、ユーザーは多くの場合、表内の、より構造化された、その他の列に対する検索と組み合わせてコンテンツベースの検索を実行できます。このアプリケーションでは、次のような検索を実行できます。

- **AND検索** - "Microsoft"と"employment"が記載されているドキュメントをすべて見つけます。
- **トピックベースの検索** - 金融または政治に関するドキュメントを（これらの単語が明確に記載されていない場合でも）すべて見つけます。
- **近接性検索** - 単語"windows"の近くに"Microsoft"が記載されているドキュメントをすべて見つけます。ドキュメントは、用語の一致数と近さに応じてランク付けされます。

コンテンツ分析

大手のオンライン小売業者とオンライン・マーケットプレイスは、製品レビューと売上データを収集しています。顧客はテキスト検索を使用してカタログ内の製品を見つけることができますが、小売業者は、製品レビュー・データに対して機械学習ベースの分析を実行して、顧客ベースと製品に関する理解を深めることもできます。ここで使用される、Oracle Textの高度な機能には、以下が含まれます。

- **センチメント分析** - レビュー・テキストからポジティブなレビューとネガティブなレビューを認識する機能です。
- **クラスタリング** - 類似したレビューをまとめて偽のレビューを特定します。特定の販売者がさまざまな"顧客"から多数の類似したレビューを得ている場合、問題が存在する可能性があります。
- **固有表現認識（NER）** - レビュー・テキスト内の特定の"もの/こと"を認識する機能です。レビュー・ポリシーでは、（変動する可能性がある）価格や、他の製品または販売者についてレビュー内で言及することは禁止されています。NERは、レビュー・テキスト内のこのようなもの/ことを特定し、さらに確認するためにそのレビューにフラグを付けることができます。

Oracle Textの入門ガイド

次の例は、サポートされているすべてのOracleデータベース・リリースで機能します。該当する場合、リリース間の構文の違いについて明示的に言及します。

次のように作成された"customers"と呼ばれる表があるとしましょう。

```
CREATE TABLE customers (
        cust_id NUMBER,
        cust_name VARCHAR2(80),
        create_date DATE );
```

次のように、この表にデータを追加しました。

```
INSERT INTO customers VALUES (1, 'The Acme Manufacturing Company, Inc.', SYSDATE); INSERT INTO customers VALUES (2, 'Coyote Trap Construction GMBH', SYSDATE);
```

ここで、"create search index"構文を使用してテキスト索引を作成できます。Oracle Database 21c以降では、これは次のように簡単になります。

```
CREATE SEARCH INDEX cust_text_index ON customers (cust_name);
```

21cより前のデータベース・バージョンを使用している場合、別の構文を使用してこの索引を作成する必要があります。次の例では、コミット時に自動的に同期される索引を作成します。

```
-- この構文は21cより前で使用されています。
CREATE INDEX cust_text_index ON customers (cust_name) INDEXTYPE IS ctxsys.context
PARAMETERS ('SYNC (ON COMMIT)');
```

ここで、CONTAINS問合せ演算子を使用してこの表内の行を見つけることができます。CONTAINSは、一致するものがない場合に0を返し、一致するものがある場合に0より大きい数値を返す関数です。この関数は、検索するための列名とテキスト問合せを使用します。

```
SELECT cust_id, cust_name FROM customers
WHERE CONTAINS ( cust_name, 'acme' ) > 0;
```

これはSQLであるため、CONTAINSをその他の演算子と簡単に組み合わせることができます。

```
SELECT cust_id, cust_name FROM customers
WHERE CONTAINS ( cust_name, 'construction' ) > 0
AND create_date > '01-JAN-23';
```

関連性を確認するには、別のテキストを追加する必要があります。

```
INSERT INTO customers VALUES (3, 'Trapezium and Trappist Developments', SYSDATE);
```

これをコミットしてから、索引がバックグラウンドの索引付けタスクによって更新されるまで数秒待つ必要があります（通常のBTREE索引とは異なり、テキスト索引はデフォルトでは同期していません）。

```
COMMIT;
```

関連性は複雑なトピックですが、基本的なレベルでは、探している検索用語が1回の結果でより頻繁に得られる場合、その結果の関連性はより高いです。それでは、'trap'から始まる単語を探してみましょう。

```
SELECT cust_id, cust_name FROM customers
WHERE CONTAINS ( cust_name, 'trap%' ) > 0;
```

ワイルドカードの'%'に注目してください。SQLのLIKE句の場合のように、%は複数文字のワイルドカードを表しますが、SQL演算子LIKEと比較してみてください。Oracle TextのCONTAINS検索では、本質的に大文字と小文字が区別されません。

この場合、行2および3が返されますが、その順序は決まっていません。もっとも関連性が高い結果を最初に表示したい場合、SCORE()演算子を、CONTAINSの別のパラメータと一緒に使用する必要があります。この3番目のパラメータは数値（その内容は関係ありません）です。これは、SCORE演算子内でも使用されます。この数値は単にSCOREを特定のCONTAINSに関連付けます。問合せに複数のCONTAINSが含まれる場合、各CONTAINSには独自のスコアがあります。

```
SELECT SCORE(1), cust_id, cust_name FROM customers WHERE
CONTAINS ( cust_name, 'trap%' ,1 ) > 0
ORDER BY SCORE(1) DESC;
```

ここでは"1"を使用しましたが、99や768などを使用することもできました。各位置の数値が同じである限り、それでも問題ありません。出力は次のようになります。

```
  SCORE(1)        CUST_ID       CUST_NAME
---------- ---------- ----------------------------------------------------------

         7          3 Trapezium and Trappist Developments
         4          2 Coyote Trap Construction GMBH
```

このSCORE()値に絶対的な意味はありません。最大値は100ですが、"最初の結果の関連性は7%である"と言うことはできません。これは、関連性がより高い結果を関連性がより低い結果より前に返すことができるように設計されているにすぎません。一般的に、アプリケーションは、結果をスコア順にソートし、表示する上位N個を決定しますが、このスコアをユーザーに見える値として返すわけではありません。

アーキテクチャ

これまで見てきた例は、デフォルトの標準テキスト索引が対象です。データベース列内の英語のプレーン・テキストに索引を付けたい場合、必要なのはデフォルトの索引のみです。しかし、テキスト索引はほぼ無限にカスタマイズ可能です。何がカスタマイズ可能であるか（およびなぜカスタマイズした方がよいのか）を理解するには、ドキュメントにどのように索引が付けられるのかをもう少し詳しく理解する必要があります。

このセクションでは、Oracle Textによるテキスト処理のメカニズムについて説明します。このプロセスをパイプラインとみなすことができます（図1）。パイプラインの各ステージはカスタマイズ可能です。このセクションでは、各ステージについて説明し、それぞれのステージで使用可能なオプションの一部を取り上げていきます。

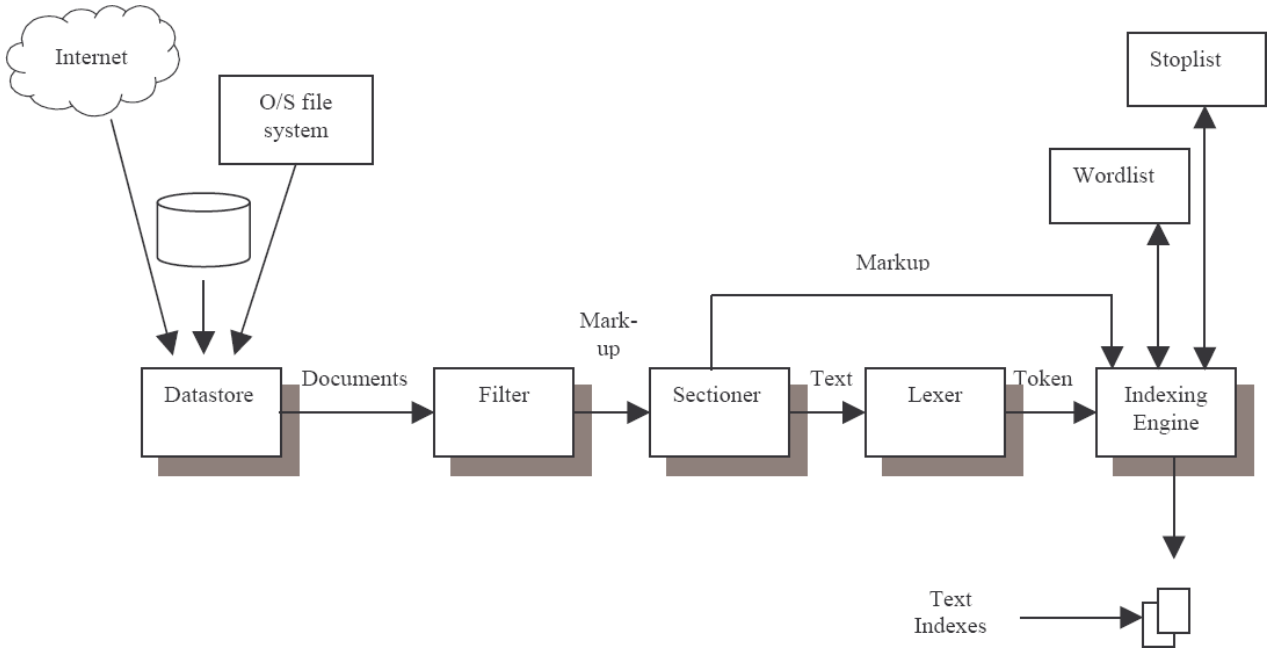

図1：索引付けのアーキテクチャ

ORACLE

データ・ストア

データ・ストアでは、索引付けするテキストの保存元を定義します。テキストが通常のデータベース列に保存されている場合、データ・ストアを指定する必要はありません（デフォルトのDIRECT_DATASTOREが使用されます）。提供されたデータ・ストアは、データベースやファイル・システムに保存されているテキスト（DIRECTORY_DATASTORE）、またはHTTPまたはHTTPSプロトコルによってリモート・アクセスするテキスト（NETWORK_DATASTORE）を対象とすることができます。ユーザーが選んだ場所、プロトコル、またはアプリケーションからデータをフェッチする、カスタム・データ・ストアを定義できます。

デフォルト・データ・ストア

デフォルト・データ・ストアはデータベース内にあります。テキストはVARCHAR2列またはCLOB（Character Large OBject）列に保存できます。フォーマット済みテキスト（WordやPDFドキュメントなど）はBLOB（Binary Large Object）列に保存できます。

ディレクトリ・データ・ストア

索引付けするテキストは、データベース・サーバーにアクセス可能なファイル・システムに保存されます。データ・ストアが定義されると、データベースDIRECTORYオブジェクトが提供されます。ファイル名は（必要に応じてサブディレクトリ・パスとともに）表の索引付き列内に保存されます。

ネットワーク・データ・ストア

データベースにHTTPプロトコルURLが保存されて、索引付けされるテキストは索引付けの時点でURLから直接フェッチされます。これは、標準のデータベース・アクセス制御リスト（ACL）メカニズムを使用して、外部ソースへのアクセスが制御されるようにします。

ユーザー定義データ・ストア

索引付けの対象となる表の各行に対して呼び出されるPL/SQLプロシージャが指定されます。次にPL/SQLプロシージャが、Java（データベースで実行中の場合は直接）やC/C++プログラムなど、他の言語で作られたプログラムをEXTPROC外部プロシージャ・メカニズムを介して呼び出すことができます。その結果、ユーザーは索引付けの対象を完全に管理できます。

フィルタ

フィルタ・ステージでは、Microsoft OfficeファイルやPDFドキュメントなど、"フォーマット済み"のドキュメントの処理が実行されます。組込みのAUTO_FILTERは一般的なすべてのドキュメント・フォーマットを認識でき、索引付け可能なHTMLテキストに変換できます。

アプリケーション開発者は、このフィルタ・ステージを自分たちが開発したカスタム構築のフィルタや、サード・パーティから購入したフィルタに置き換えることが可能です。

カスタム・フィルタは、単に2つの引数を取る実行可能プログラムまたはスクリプトであり、最初の引数はフォーマット済み入力テキストを含むファイルで、2番目の引数はフィルタリングされた出力を書き込むファイルの名前です。必要に応じて、カスタム・フィルタによって標準（自動）フィルタを呼び出すことができます。この方法により、企業独自のファイル・フォーマットを処理できますが、標準のファイル・フォーマットは標準フィルタに渡されます。

セクショナ

セクショナ・オブジェクトは、各テキスト・ユニットを含むセクションを識別します。たとえば、次のテキストに索引を付けたいとします。

<title>A Tale of Two Cities</title>

<author>Charles Dickens</author>

It was the best of times, it was the worst of times...

AUTO_SECTIONERを使用すると、このテキストに索引を付けることができます。これにより、書名と著者の前後にあるXMLに似たセクション・タグが識別され、次のような検索を行うことができます。

```
WHERE CONTAINS (text, 'Dickens WITHIN author') > 0
```

あるいは、BASIC_SECTIONERを使用し、書名と著者のセクションを、これらを識別するタグを使用して事前に定義することもできました。

また、セクショナを使用して、構造化された値（文字列、数値、日付）をテキスト検索に含めることもできます。

ORACLE

レクサー

レクサーは、セクショナの出力をワードまたはトークンに分離するためのものです。西ヨーロッパ系の言語におけるもっとも簡単な例をあげると、レクサーによってテキストはアルファベットの連続した文字列に分割されます。次の文字列があるとします。

Aha!It's the 5:15 train, coming here now!

この文字列は、記号と特殊文字を取り除いた単語に分割されます。

aha it s the 5 15 train coming here now

レクサーは通常、ストップワードを削除します。ストップワードとは、アプリケーション開発者により定義されるか、またはデフォルト・リストから取得される共通ワードです。その結果、上記の文字列は以下のようになります。

aha * * * 5 15 train coming * now

アスタリスクは、削除されたストップワードを表します。実際に索引付けはされませんが、ストップワードがその位置に存在することが索引に記述されます。検索では、ストップワードはフレーズの一部として使用されている単語と一致します。たとえば、"*coming in now*"を検索すると、上記のテキストがマッチします（"*in*"はストップワードであり、テキスト内の任意のストップワードにマッチするためです）。しかし、"*coming rapidly now*"はマッチしません。なぜなら、"rapidly"はストップワードではなく、元のフレーズ内で見つからないからです。

ストップワードはアプリケーション開発者が指定できます。また、すべての数字をストップワードとして定義できます。

レクサーの基本設定

レクサーを微調整するためのオプションが多数用意されています。たとえば、開発者は索引での大/小文字の区別の有無を選択でき、*"PL/SQL"*を2つの単語*"PL"*と*"SQL"*として索引付けしたり単一の文字列*"PL/SQL"*として索引付けするなど、特定の文字をトークンに分割したり、トークンの一部として索引付けしたりすることもできます。

索引付けエンジン

索引付けエンジンは、トークンを含むドキュメントにトークンをマッピングする逆向きの索引を作成します。この段階では、あらかじめ指定されていれば、Oracle Textによってストップリストが使用されます。ストップリストには、テキスト索引から排除する必要がある単語やテーマをユーザーが指定できます。

パイプラインの最終出力は逆向きの索引になります。これは、ドキュメントから選び出した単語のリストであり、各単語にはそれが使用されているドキュメント（および場合によってはドキュメント・セクション）のリストが添付されます。これはテキストの通常の見方とは逆であることから逆向きの索引と呼ばれており、各ドキュメントに単語リストが付いた状態でのドキュメント・リストになっています。

索引のカスタマイズ

索引は、CREATE INDEX文の*PARAMETERS*句を使用してカスタマイズされます。これは、21c以降の"*create search index*"構文を見ているか、それより前のバージョンの"*CREATE INDEX … INDEXTYPE IS ctxsys.context*"を見ているかとは関係なく当てはまります。簡単な例を見てみましょう。一連の歌に索引を付けます。

create table songs (songtitle varchar2(80));

insert into songs values ('let it be');
create search index songtitleindex on songs(songtitle); （Oracle Database 21c以降）

CREATE INDEX songtitleindex ON songs (songtitle) INDEXTYPE IS ctxsys.context PARAMETERS (‘SYNC (ON COMMIT)’);
（以前のバージョン）

ここで、'let'を検索すると、次のエントリが見つかります。
SQL> select * from songs where contains (songtitle, 'let') > 0; SONGTITLE
----

let it be

しかし、'it'を検索しても、見つかりません。

SQL> select * from songs where contains (songtitle, 'it') > 0;
no rows selected

どうなっているのでしょうか。単語'it'はデフォルトのストップワードであり（データベースがデフォルトの言語を英語としてインストールされている場合）、ストップワードには索引が付けられていません。すべての単語に索引を付けたい場合、CTXSYSスキーマでempty_stoplistという事前定義済のSTOPLISTを使用できます。

create search index songtitleindex on songs(songtitle) parameters ('stoplist ctxsys.empty_stoplist'); （Oracle Database 21c以降）

CREATE INDEX songtitleindex ON songs (songtitle) INDEXTYPE IS ctxsys.context PARAMETERS ('SYNC (ON COMMIT) stoplist ctxsys.empty_stoplist'); （以前のバージョン）

これで、'it'などの一般的な単語を検索できるようになります。

```
SQL> select * from songs where contains (songtitle, 'it') > 0;
SONGTITLE
--------------------------------------------------------------------------------
let it be
```

ここでは、カスタマイズされた基本設定とストップリストを作成するシステムについては説明しません。詳細については、ドキュメントを参照してください。

統合テキスト検索機能の利点

外部テキスト・エンジンは広く利用可能です。その大半は、ファイル内のテキストを利用するか、REST APIを介してエンジンに渡されたテキストのみを利用します。しかし、最初にテキストがデータベース内にある場合は、Oracle Database内の統合テキスト・エンジンを使用する方が賢明です。

これは、開発者またはアプリケーション所有者が以下を利用していることを意味します。

- すべてのデータ（テキスト・データと構造化データ）に対しての2つのリポジトリではなく単一のリポジトリ。これにより、保守やバックアップなどが容易になります。
- 同一リポジトリでの索引。これにより、テキスト問合せおよび複合問合せを効率的に処理できます。
- 単一のAPIによるアプリケーション開発。オラクルのコンバージド・データベースを活用し、テキスト索引をデータベース内のその他の処理（空間、JSON処理、グラフなど）と組み合わせます。
- Oracle SQL実行エンジンおよび問合せプラン・オプティマイザとの統合。文全体を対象としたもっとも効率的な実行計画をオラクルに決定させます。

次のように要約すると、統合の利点は明白です。

- **低コスト**

  Oracle TextはOracle Databaseのすべてのエディションに含まれています。別の製品を購入、統合、または保持する必要はありません。

- **高パフォーマンス**

  データベースによって、テキストと構造コンテンツの両方を含む問合せを実行する最速のプランが選択されます。

- **高度な整合性**

  テキストはデータベースに保存されるため、テキストが整合性の利点をすべて継承します。たとえば、データベースに対するすべての更新をテキスト検索機能に反映できるため、ユーザーはすべてのデータについて統合された総合的なビューを得られます。

- **複雑さの軽減**

  テキストは構造化データと同じように処理されます。このため、既存システムでのテキスト検索アプリケーションの開発および統合が容易です。

- **優れた管理性**

  Oracle Textは、管理者が一般的に持つスキルを活用して、標準的なエンタープライズ管理ツールによって管理することが可能です。

- **セキュリティ**

  Oracle Textではデータベースのセキュリティ機能が使用されます。

ORACLE

Connect with us

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

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