[Oracle Text 超入門]

目次

確認環境：Solaris 8、Oracle9.2.0 です。

1. Oracle Text とは

全文検索ツールです。
全文検索とは、マニュアル、報告書、Mail、WEB リソース等の膨大なテキスト情報の、本文自体を検索対象とする検索方法です。
インターネット/イントラネット上に点在する膨大な情報の中から、利用者が必要な情報を効率的に探し出すためのサービスとして多く利用されています。
yahoo や google 、日本オラクルの技術サイトである OISC で使用されている検索方法と言えば、イメージしやすいかと思います。

Oracle をはじめとする RDBMS では、もともと定型データを管理する目的で設計されました。
例えば社員情報であれば、社員番号、社員名、入社年月日、住所 等の項目を設け、その項目にデータを入力することで、管理・検索を行うことが可能です。
しかし、通常業務の中で有効な情報は、このような定型データだけではありません。
むしろ、マニュアルや報告書等の情報の方が多い状況です。

そこで、これらの非定形文書を検索対象とするために、Oracle が提供している全文検索機能が、Oracle Text です。
Oracle Text では、プレーン・テキスト、HTML、XML、Microsoft Word、Excel 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。(PDF、一太郎 等も含まれます。)
これらのデータは、必ずしもデータベース内に格納する必要はありません。

Oracle Text では、これらのデータを元に、テキスト索引を作成します。
テキスト索引とは、どの単語(語句)がどの文書に出現するかをまとめたものです。
マニュアル等の巻末にある索引のようなものをイメージしていただければ分かりやすいかと思います。 例えば、以下のようなイメージです。

実際の文　　索引

実際の文
The dog runs. The cat runs.

索引	
THE	(1-1)(2-1)
DOG	(1-2)
RUNS	(1-3)(2-3)
CAT	(2-2)

例えば「cat」という単語が条件に指定された場合は、まず索引を確認し、「cat」が、文書番号 2 番の 2 単語目にあることが分かりますので、「The cat runs.」が結果として返されます。

例では短い文章が 2 レコードしかないため、感動は薄いかと思いますが、1000 ページあるマニュアルが 1000 ドキュメントあると想像してみて下さい。
1000 ドキュメントを 1 つ 1 つ開き、該当する単語があるかどうか調べるよりも、索引を使用してピンポイントでドキュメントを取得する方がはるかに早いことは容易に予測できます。

本資料では、Oracle Text の概要、使用方法を簡単にご紹介いたします。
参考にしていただければ幸いでございます。

2. Oracle Text の概要

1. 使用可能なデータ型

Oracle Text で使用できる(テキスト索引を作成可能な)データ型は以下のものになります。

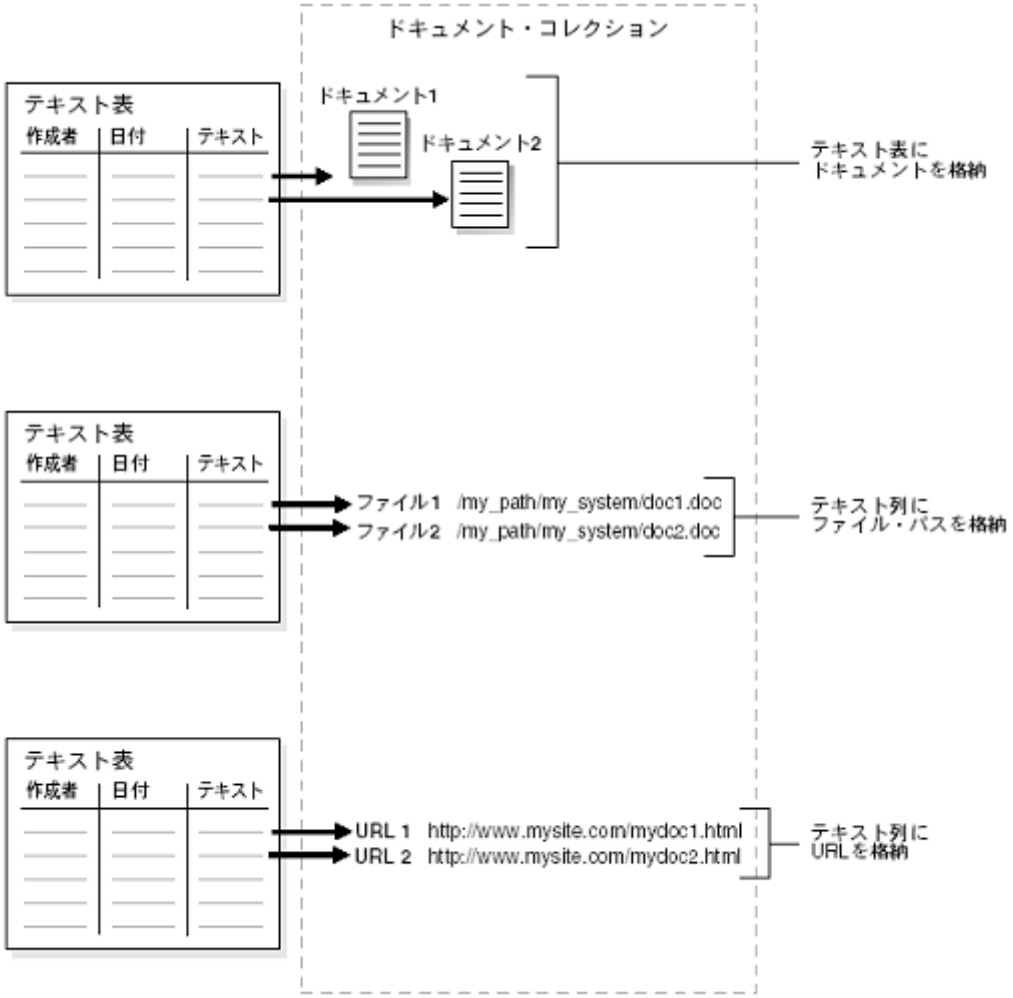
char / varchar2 / clob / blob / bfile / XMLType / URIType

通常の index は lob 型に対して作成することはできませんが、Oracle Text では、clob や blob 等の lob 型のデータを扱うことができます。

2. テキストデータの格納方法

データの格納方法には以下のように、3 通りあります。

- 直接テキストデータをデータベースに格納する方法
- データベースには、テキストまでのフルパスを格納しておき、実際のテキストは OS 上に置く方法
- データベースに URL を格納しておき、WEB リソースはそのまま WEB 上に置く方法



どの格納方法でも、テキスト索引を作成し、検索時に使用することが可能です。
また、どの格納方法でも、Oracle データベースに格納された既存の定型データとの連携が可能です。
実際のテキストデータをデータベースに格納する必要がありませんので、既存のリソースをフルに活用できます。

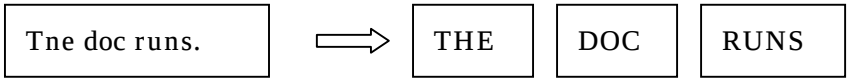
3. アルゴリズム

テキスト索引を作成する際の、単語(語句)を切り出す単位を トークン といいます。
テキスト索引を作成するアルゴリズムは、英語やフランス語等の単語が空白で区切られる言語と、日本語とでは異なります。
ここでは簡単に、英語と日本語における、トークン作成のアルゴリズムをご紹介いたします。

A. 英語の場合

英語は空白で単語が区切られていますので、トークンも空白区切りで作成されます。

例えば、 The doc runs. では、 The 、doc 、 runs に分けられます。



※すべてのアルファベットは基本的にシングルバイトの大文字のアルファベットに変換されてトークン表に格納されます。

英語等、空白区切りでトークンが作成される場合には、検索時に注意が必要です。

トークン表には「RUNS」しかありませんので、検索時に「RUN」を条件指定しても、「The doc runs.」のレコードはヒットしません。

「RUNS」をヒットさせるためには、「RUNS」もしくは「RUN%」のようにワイルドカード(%)を使用した検索語句を指定する必要があります。

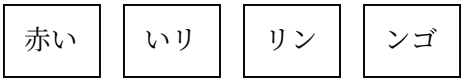
B.日本語の場合

日本語は単語が空白で区切られておりませんので、特殊なアルゴリズムを用いています。

日本語テキストを全文検索するアルゴリズムの 1 つに N-GRAM インデックスがあります。

N-GRAM インデックスとは、文書に現れる文字を固定の長さ N ごとに切り出す方式です。

文字列 「赤いリンゴ」 を N=2 として切り出すと、



トークン	出現位置
赤い	1-1
いリ	1-2
リン	1-3
ンゴ	1-4

となります。

検索時には、検索条件に指定した語句もトークンに切り分けられ、それぞれのトークンがどこに出現するかを調べます。そして、トークンが連続している文書がヒットすることになります。

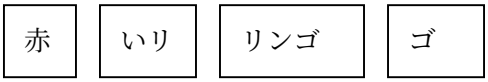
Oracle Text では N-GRAM を改良した V-GRAM アルゴリズムを使用しています。

2-GRAM で切り出すとトークンの先頭に「ゃ」や「ん」などがあらわれる場合があります。

日本語の先頭文字が「ゃ」や「ん」などから始まる単語はありません。

そこでこのようなトークンを作成しないようにトークン長を可変にしています。

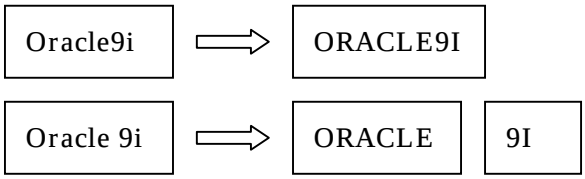
V-GRAM アルゴリズムでは、「赤いリンゴ」は以下のようにトークンに切り出されます。



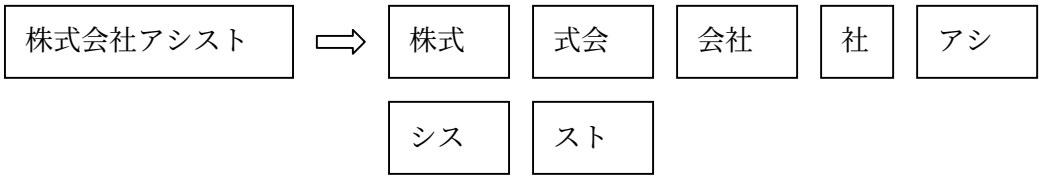
漢字・カタカナ/ひらがな で分割され、さらにそれぞれの文字列をトークンに分割します。

トークン作成時の動作をまとめますと、以下になります。

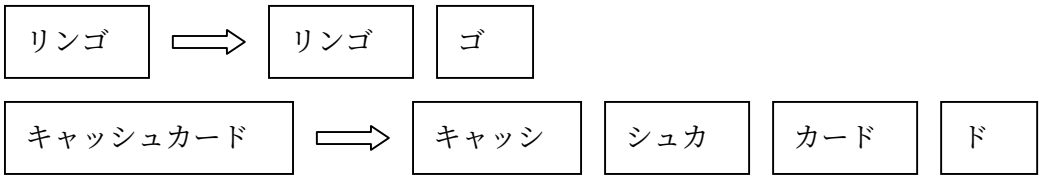
英語



日本語



を・ん・長音・ゃ・ゅ・ょ 等を含む場合は以下のようになります。



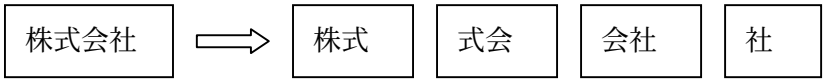
日本語では、指定した文字とトークン表との完全一致ではなく、上記のように、部分一致で検索を行います。
そのため、英語の場合と違い、ワイルドカード(%)を使用しましても意味がありません。

※ ワイルドカードを使用するように設計されておりませんので、意図しない動作になる可能性がございます。日本語に対しては、ワイルドカードは使用されないようにご注意下さい。

尚、検索語句が日本語の場合、もう 1 つ注意すべき点があります。
検索語句が日本語の場合は、内部的に Like 検索が行われる場合があります。

・検索語句が 1 文字の場合

「株式会社」は、以下のように 4 つのトークンに分割されます。

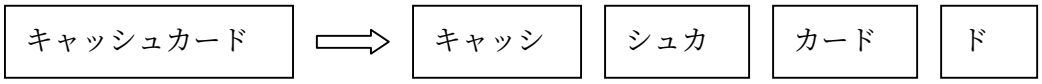


検索文字に「株」を指定した場合、トークン表に「株」がありませんので、上記レコードはヒットしません。

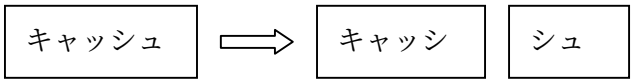
このような状況が考えらますので、Oracle Text では、「株式会社」をヒットさせるために、内部的に「株%」として検索されます。

・語尾が長音や「ゃ」「ゅ」「ょ」で終わる場合

「キャッシュカード」は、以下のように 4 つのトークンに分割されます。



また、「キャッシュ」は以下のように 2 つのトークンに分割されます。



検索語句に「キャッシュ」と指定した場合、「キャッシ」はトークン表にありますが、「シュ」はありません。そのため、「キャッシュカード」が検索されません。

このような状況が考えらますので、Oracle Text では、「キャッシュカード」をヒットさせるために、内部的に「シュ%」として検索されます。

テキスト索引に対して Like 検索を行いますと、トークン表でヒットする件数が多くなり、検索パフォーマンスが劇的に劣化する恐れがありますので、ご注意下さい。

3. 基本的な使用方法

ユーザ作成から検索ができるようになるまでの一連の流れを簡単にご紹介いたします。
この章を参照していただくことで、以下のことが分かります。

1. Oracle Text を使う上で必要な権限
2. Preference の作成方法
3. Index の作成方法
4. テキスト索引を使用した検索方法
5. Like 検索との比較
6. テキスト索引の確認方法
7. データを追加・更新・削除した際のテキスト索引の動作
8. 9. キスト索引のメンテナンス方法

0. 事前準備

0-1. Oracle Text が使用可能な状態か確認します。

System ユーザでデータベースにログインし、「ctxsys」というユーザが存在するかどうかご確認下さい。

```
sqlplus system/manager
SQL> select username from all_users
  2   where username='CTXSYS';

USERNAME
------------------------------
CTXSYS
```

Ctxsys ユーザが存在していない場合、Oracle Text を使用できる環境ではありません。
「付録-1 Oracle Text を手動で設定する方法」を参照し、まずは Oracle Text の動作環境をインストールして下さい。

0-2. Ctxsys ユーザのロックを解除

Oracle9i では、デフォルトで ctxsys ユーザがロックされており、使用できない状態になっています。
以下の手順でロックを解除して下さい。

```
sqlplus system/manager
SQL> alter user ctxsys identified by ctxsys account unlock;

ユーザーが変更されました。
```

0-3. テスト用ユーザ作成・Table 作成

※テストユーザとして、test、テストテーブルとして doc を作成します。

```
sqlplus system/manager
SQL>   create user test identified by test
    2    default tablespace USERS
    3    temporary tablespace TEMP;

ユーザーが作成されました。

SQL>   grant connect,resource to test;

権限付与が成功しました。
```

```
SQL>   connect test/test
接続されました。
SQL>   create table doc (no number,text varchar2(30));

表が作成されました。

SQL>   insert into doc values (1,'ｳｻｷﾞとｶﾒ');
SQL>   insert into doc values (2,'兎とカ　メ');
SQL>   commit;

SQL>   select * from doc;

    NO TEXT
---------- --------------------
       1 ｳｻｷﾞとｶﾒ
       2 兎とカ　メ
```

また、現在、test ユーザが所有するテーブルは DOC のみであることを確認します。

```
SQL> select * from tab;

TNAME                TABTYPE     CLUSTERID
------------------------------ ----------------- --------------------
DOC                   TABLE
```

1. CTXAPP ロールの付与

Oracle Text のテキスト索引を作成する為には、CTXAPP ロールが必要です。

```
SQL>   connect ctxsys/ctxsys
接続されました。
SQL>   grant ctxapp to test;

権限付与が成功しました。
```

CTXAPP ロールには以下の権限が含まれます。

・テキスト・データ・ディクショナリの管理（プリファレンスの作成および削除を含む）

・Oracle Text の PL/SQL パッケージの使用

2. preference 作成

今回は、JAPANESE_VGRAM_LEXER を使用します。

Preference の作成・削除・更新は、ctx_ddl パッケージを使用します。

```
SQL> begin
  2        ctx_ddl.create_preference('my_lexer','JAPANESE_VGRAM_LEXER');
  3    end;
  4    /

PL/SQL プロシージャが正常に完了しました。
```

※日本語を扱う lexer には、JAPANESE_VGRAM_LEXER の他に、JAPANESE_LEXER もあります。

3. Index 作成

Oracle Text のテキスト索引は、通常のインデックス作成と同様に、create index 文で作成します。

その際、indextype として ctxsys.context を、parameter 句で priference を指定します。

```
SQL> create index doc_idx on doc(text)
  2  indextype is ctxsys.context
  3  parameters('lexer my_lexer');

索引が作成されました。
```

※ 指定する preference が複数ある場合、例えば lexer を my_lexer に、storage を my_storage に設定する場合、parameters('lexer my_lexer storage my_storage') というように、prefernce 全体を「' (シングルコーテーション)」で囲み、各 preference はスペースで区切ります。

4. テキスト検索

index が作成できましたので、3 で作成した index を使用して、実際に検索してみましょう。

テキスト索引を使用した検索結果

```
SQL> select * from doc
  2  where contains (text,'カメ') > 0;

   NO TEXT
---------- --------------------
         1 ｳｻｷﾞとｶﾒ
         2 兎とカ メ
```

DOC テーブルの全データ

```
SQL>  select * from doc;

   NO TEXT
---------- --------------------
         1 ｳｻｷﾞとｶﾒ
         2 兎とカ メ
```

WHERE 句で CONTAINS(text, 'カメ') > 0 と指定することで、text 列に「カメ」という文字が含まれている文書が検索されます。

※ 「> 0 」 は、「スコアが 0 より大きいもの」、つまり、ヒットしたものすべてを表示するという意味になります。

※ JAPANESE_VGRAM_LEXER では、全角・半角の区別はなく、空白は考慮されませんので、「カメ」というキーワードを指定して検索しますと、NO = 1 の「ｶﾒ」、NO = 2 の「カ メ」、も検索されます。

5. Like 検索との比較

一部の単語(語句)を条件に、テキスト文書の中から曖昧検索を行うという意味では、Like 検索と Oracle Text のテキスト検索は似ていますが、全く別の機能です。

4 で、テキスト検索で行った検索を、Like 検索で実行した例をご覧下さい。

Like 検索の結果

```
SQL> select * from doc
  2   where text like '%カメ%';

レコードが選択されませんでした。
```

DOC テーブルの全データ

```
SQL>   select * from doc;

    NO TEXT
---------- --------------------
         1 ｳｻｷﾞとｶﾒ
         2 兎とカ メ
```

Like では、全角・半角、大文字・小文字の区別、空白等も考慮しますので、「カメ」を条件に指定して検索しますと、1 件もヒットしません。

6. トークン確認

Oracle Text の索引は、検索可能なものがいくつかあります。

その主なものに、トークン表があります。

Index を作成しますと、4 つの実表が作成されます。

```
SQL> select * from tab;

TNAME                          TABTYPE     CLUSTERID
------------------------------ ----------------- --------------------
DOC                             TABLE
DR$DOC_IDX$I                    TABLE
DR$DOC_IDX$K                    TABLE
DR$DOC_IDX$N                    TABLE
DR$DOC_IDX$R                    TABLE
```

このうち DRDOC_IDXI 表がテキスト索引表です。

DR$<index_name>$I 表の構成は以下のようになっています。

```
SQL> desc DR$DOC_IDX$I
 名前                                      NULL?    型
 ----------------------------------------- -------- ----------------------------
 TOKEN_TEXT                                NOT NULL VARCHAR2(64)
 TOKEN_TYPE                                NOT NULL NUMBER(3)
 TOKEN_FIRST                               NOT NULL NUMBER(10)
 TOKEN_LAST                                NOT NULL NUMBER(10)
 TOKEN_COUNT                               NOT NULL NUMBER(10)
 TOKEN_INFO                                         BLOB
```

TOKEN_INFO 列は BLOB 型のため、確認することはできませんが、その他の情報は通常の SQL 文で確認可能です。

トークンの確認は、以下のように行うことができます。

```
SQL> select TOKEN_TEXT from DR$DOC_IDX$I;

TOKEN_TEXT
----------------------------------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

7. データの追加・更新・削除

データが追加・更新・削除された場合、テキスト索引はどうなるでしょうか。

データが追加・更新・削除された際のトークン表の動きと、テキスト索引のメンテナンスについてご紹介します。

A. データの削除

NO = 2 のデータを削除します。 (NO=2 は、TEXT='兎とカ メ' です。)

```
SQL> delete from doc
  2   where no = 2;

1 行が削除されました。

SQL> commit;

コミットが完了しました。

SQL> select * from doc;

    NO TEXT
---------- ------------------------------
       1 ｳｻｷﾞとｶﾒ
```

トークン表がどうなったか、確認してみましょう。

データ削除後

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

データ削除前

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

NO= 2 のデータを delete したにも関わらず、トークン表には反映されておらず、「兎」のエントリが残っています。

この状態で、条件に「兎」を指定して検索してみましょう。

```
SQL> select * from doc
  2    where contains (text,'兎') > 0;

レコードが選択されませんでした。
```

トークン表には delete の情報は反映されていませんが、検索結果には反映されます。
Oracle Text では、更新情報が直ぐにトークン表に反映されない代わりに、行が削除された情報は、DRDOC_IDXN 表に一時的に格納されます。

```
SQL> select * from DR$DOC_IDX$N;

 NLT_DOCID N
---------- -----
         2 U
```

$N 表には、削除された行の DOCID が格納されます。delete が行われますと、NLT_MARK 列に'U'が格納されます。

B. データの追加
では insert 処理はどうでしょうか。

```
SQL> insert into doc values(3,'桃太郎');

1 行が作成されました。

SQL> commit;

コミットが完了しました。

SQL> select * from doc;

    NO TEXT
---------- ------------------------------
         1 ｳｻｷﾞとｶﾒ
         3 桃太郎
```

トークン表を確認します。

データ挿入後

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

データ挿入前

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

Insert の場合も、トークン表には反映されません。
そのため、検索にもヒットしません。

```
SQL> select * from doc
  2   where contains (text,'太郎') > 0;

レコードが選択されませんでした。
```

C. データの更新

Update を行った場合も同様にトークン表には反映されません。

```
SQL> update doc set text='ウサギ'
  2   where no=1;
SQL> commit;
SQL> select * from doc;

     NO TEXT
---------- ------------------------------
         1 ウサギ
         3 桃太郎
```

データ更新後

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

データ更新前

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

トークン表に反映されておりませんので、もちろん、検索にもヒットしません。

```
SQL> select * from doc
  2   where contains (text,'ウサギ') > 0;

レコードが選択されませんでした。
```

このように、テキスト索引では、データの更新が直ぐにトークン表に反映しませんので、insert や update の情報が検索結果に反映しません。
そのため、Oracle Text では、リアルタイムの情報を取得するためには、定期的に元表とテキスト索引との同期を取る必要があります。これを同期化(sync) と言います。

8. メンテナンス(同期化)

元表への更新をテキスト索引に反映させる(同期化する)には、ctx_ddl パッケージの sync プロシージャを使用します。

```
SQL> begin
  2       ctx_ddl.sync_index('DOC_IDX');   -- Index 名を指定します。
  3   end;
  4   /

PL/SQL プロシージャが正常に完了しました。
```

トークン表を確認してみましょう。

同期化処理後

```
TOKEN_TEXT
--------------------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎
ウサ
ギ
サギ
太郎
桃太
郎

13 行が選択されました。
```

同期化前

```
TOKEN_TEXT
----------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎

7 行が選択されました。
```

処理後の $N 表

```
NLT_DOCID N
---------- -----
         2 U
```

追加テキストから得られたトークンはテキスト索引の最後に追加されます（既にそのトークンのエントリが存在しても、そこへは追加されません）。
また、delete や update 前のデータである無効なエントリ(例えば、兎)が削除されず、そのまま残ってしまいます。
この結果、同じトークンが複数エントリされるという状況や、無効なエントリが残ったままという状況になります。※$N 表の情報もそのままです。

データが更新され、同期化(sync)を行うたびにトークン表のエントリが増えますので、このままでは表領域を圧迫してしまいます。
また、検索時に無駄にスキャンしなければならない件数が増えますので、パフォーマンスにも影響します。
そのため、Oracle Text では、同期化の他に、定期的なメンテナンスを行う必要があります。
これを最適化(optimize)といいます。

9. メンテナンス(最適化)

最適化(optimize)は索引内に存在する同一トークンを一つにまとめ、無効なエントリを削除する作業です。

最適化(optimize)には、3 つのモードがあります。

・高速
・完全
・トークン

A. 高速最適化（optimize fast）

既存のテキスト索引を元にテキスト索引の整理を行うだけで、delete されて既にテキスト表に存在していないトークンでもテキスト索引から削除することはしません。

```
SQL> begin
  2       ctx_ddl.optimize_index('DOC_IDX','fast');
  3   end;
  4   /

PL/SQL プロシージャが正常に完了しました。
```

最適化処理後

```
TOKEN_TEXT
----------------------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎
ギ
太郎
桃太
郎

11 行が選択されました。
```

最適化処理前

```
TOKEN_TEXT
----------------------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎
ウサ
ギ
サギ
太郎
桃太
郎

13 行が選択されました。
```

重複していたエントリは纏められましたが、「兎」などの無効なエントリは削除されていないことがわかります。　※削除行の情報は、$N 表にあります。

```
NLT_DOCID N
---------- -----
         2 U
```

B. 完全最適化（optimize full）

既存のテキスト索引に存在するトークンを整理すると共に、delete されて既にテキスト表に存在しておらず、$N 表の情報を元に、削除されたテキストデータのトークンをテキスト索引から削除します。

```
SQL> begin
  2       ctx_ddl.optimize_index('DOC_IDX','full');
  3   end;
  4   /

PL/SQL プロシージャが正常に完了しました。

SQL> select TOKEN_TEXT from DR$DOC_IDX$I;

TOKEN_TEXT
----------------------------------------------------
ウサ
ギ
サギ
太郎
桃太
郎

6 行が選択されました。

SQL> select * from DR$DOC_IDX$N;

レコードが選択されませんでした。
```

C. トークン最適化（token）

ある特定のトークンのみの完全最適化を行います。

ネックとなっているトークンが判明している場合などは、こちらをご使用下さい。

※8 のトークン表を見ますと、「ウサ」が重複していますので、「ウサ」に対して最適化を行ってみましょう。

```
SQL> begin
  2  ctx_ddl.optimize_index('DOC_IDX','token',token=>'ウサ');
  3  end;
  4  /

PL/SQL プロシージャが正常に完了しました。
```

トークン表を確認しますと、「ウサ」が最適化され、重複が解消されていることが分かります。

最適化後

```
TOKEN_TEXT
----------------------------------------------------
とカ
カメ
ギと
サギ
メ
兎
ギ
サギ
太郎
桃太
郎
ウサ

12 行が選択されました。
```

最適化前

```
TOKEN_TEXT
----------------------------------------------------
とカ
ウサ
カメ
ギと
サギ
メ
兎
ウサ
ギ
サギ
太郎
桃太
郎

13 行が選択されました。
```

以上、Oracle Text の概要から簡単な使用方法までをご紹介しました。

付録-1　Oracle Text の手動設定手順

0.　JVM がインストールされているかどうかの確認

Oracle Text を設定するには、JVM が設定されている必要があります。
以下の SQL 文を SYSTEM ユーザ等で実行し、値を確認して下さい。

```
sqlplus system/manager
SQL> select count(*) from dba_objects
    2    where object_type like 'JAVA%';

      COUNT(*)
----------
         10211
```

R9.2.0 では、10000 程度の値になります。結果が 0 の場合、JVM がインストールされていません。まずは JVM をインストールして下さい。

尚、JVM のインストール手順は以下になります。

０) init.ora の修正
SHARED_POOL_SIZE、JAVA_POOL_SIZE を 150M 以上に設定し、Oracle を再起動します。
また、ディスクに空き容量があるかどうかも事前にご確認下さい。

1)インストール
以下のスクリプトを順に実行します。
※必ず SYS ユーザで(SYSDBA 接続で)実行して下さい。

```
$ORACLE_HOME/javavm/install/initjvm.sql
$ORACLE_HOME/xdk/admin/initxml.sql
$ORACLE_HOME/xdk/admin/xmlja.sql
$ORACLE_HOME/rdbms/admin/catjava.sql
```

※実行前に spool コマンドでインストールログを取っておくことを強くお勧めいたします。
※システムの状態やスペックによって差がでますが、大体 1 時間程度でイインストールが完了します。

2) 確認作業

正常にインストールが完了した場合は、システム表領域の Java オブジェクトの数は 10000 前後で、且つ、ステータスは全て "VALID" になっています。

以下の方法でご確認ください。

```
sqlplus system/manager
SQL> select count(*) from dba_objects
    3    where object_type like ‘JAVA%’;

      COUNT(*)
    ----------
         10211

SQL> select object_name, status, object_type
    2    from dba_objects
    3    where upper(object_type) like '%JAVA%' and status = 'INVALID';

レコードが選択されませんでした。
```

JVM のインストールは以上です。

尚、インストール時、ログに以下のエラーが出力されます。

・ORA-01432　削除するパブリック・シノニムが存在しません。
・ORA-02289　順序が存在しません。
・ORA-00942　表またはビューが存在しません。
・ORA-01418　指定した索引は存在しません。
・ORA-04043　オブジェクトは存在しません。
・ORA-29515: ステータス 0 で Java コードから終了がコールされました。

JVM インストール時に実行していただいたスクリプトでは、まず最初に既存のものを DROP し、その後、CREATE するという作業を行っております。
そのため、該当のオブジェクトが存在しない旨のエラーに関しましては、出力されておりましても問題ございません。また、戻り値 0 は、処理が問題なく終了したという意味ですので、このエラーも問題ございません。

JVM のインストールが終了しましたので、次に、Oracle Text の実行環境をインストールして下さい。

1. SQL*Plus を起動し、SYS ユーザーで接続します。

```
sqlplus /nolog
SQL> connect sys/ change_on_install as sysdba
```

2. CTXSYS ユーザを作成します。

（注意）予め CTXSYS ユーザに設定するデフォルト表領域、一時表領域が
作成済みであることをご確認ください。
表領域が作成されていない場合には作成後、次の手順へ進んでください。

この時点では CTXSYS ユーザ が所有するオブジェクトは作成されません。

```
SQL> @$ORACLE_HOME/ctx/admin/dr0csys ctxsys DRSYS TEMP;
```

ctxsys -- CTXSYS ユーザーのパスワード
DRSYS -- CTXSYS ユーザーのデフォルト表領域
TEMP -- CTXSYS ユーザーの一時表領域
※全て任意です。

3) CTXSYS ユーザーで接続して必要なオブジェクトを作成します。
ご使用のプラットフォームに応じて以下の通りに実行してください。

```
SQL> connect ctxsys/ctxsys

<Solaris, AIX の場合>
SQL> @$ORACLE_HOME/ctx/admin/dr0inst  <$ORACLE_HOME のパス>/ctx/lib/libctxx9.so;

<HP-UX の場合>
SQL> @$ORACLE_HOME/ctx/admin/dr0inst <$ORACLE_HOME のパス>/ctx/lib/libctxx9.sl;

<WindosNT の場合>
SQL> @%ORACLE_HOME%¥ctx¥admin¥dr0inst <$ORACLE_HOME のパス>¥bin¥oractxx8.dll;
```

4. 次のスクリプトで defaults preference, default lexer, wordlist, stoplist をインストールします。

※スクリプト名は drdef<country code>.sql となります。
※日本の場合 drdefjp.sql です。

```
SQL> @$ORACLE_HOME/ctx/admin/defaults/drdefja.sql;
```

5. 正常にインストールされたかどうか確認します。

A. CTXSYS ユーザーで以下の SQL 文を実行して作成されたオブジェクトの数を確認します。

```
sqlplus ctxsys/ctxsys
SQL> select object_type, count(*) from user_objects
  2  group by object_type order by 1;

OBJECT_TYPE             COUNT(*)
----------------------------------- --------------------
FUNCTION                         3
INDEX                           46
INDEXTYPE                        4
LIBRARY                          2
LOB                              2
OPERATOR                         5
PACKAGE                         53
PACKAGE BODY                    44
PROCEDURE                        1
SEQUENCE                         3
TABLE                           36
TYPE                            10
TYPE B                           7
VIE                             47
```

B. オブジェクトが VALID であるかどうかを確認します。

```
SQL>    select object_type, count(*) from user_objects
     2    where status = 'INVALID' group by object_type order by 1;

レコードが選択されませんでした。
```

C. 使用されているライブラリをライブラリを確認します。

```
SQL> column library_name format a8
SQL> column file_spec format a60
SQL> select library_name,file_spec,dynamic,status from user_libraries;

  LIBRARY_NAME    FILE_SPEC                                                    D STATUS
  --------------- ------------------------------------------------------------ --- ----------
  DR$LIB                                                                        N VALID
  DR$LIBX    /app/oracle/product/9.2.0/ctx/lib/libctxx9.so                      Y VALID
```

以上でインストールは終了です。