

# Oracle Text In Oracle Database 12c

Adding value to your textual assets.

ORACLE WHITE PAPER | MAY 2017



## **Table of Contents**

Introduction	1
Text is Everywhere	1
Oracle Text	1
Integrated security	1
Architecture	1
Datastore	2
Default Datastore	2
File Datastore	2
URL Datastore	2
User Defined Datastore	2
Filter	2
Sectioner	3
Lexer	3
Lexer Preferences	3
Language Specific Functionality	3
Western Languages	3
Multi-Byte Languages	3
Indexing Engine	4
Benefits of Integrated Text Search Capability	4
Oracle Text Features	4
Index Types	5

Substring and Prefix indexes	5	
Maintaining Indexes and Synchronization		
Parallel Indexing	5	
Locally Partitioned Indexes	6	
Query Operators	6	
Internationalization	8	
Document Services	8	
Highlighting	9	
Markup	9	
Snippet	9	
Theme Extraction	9	
Gist Generation	9	
Classification and Clustering	9	
Knowledge Base	10	
Using Oracle Text	10	
Creating Indexes with Oracle Text	10	
Optimizer Hints	12	
SDATA Sections	13	
Composite Domain Indexes	13	
XML Support	14	
Searching for content and structure in XML documents	14	

## Introduction

Oracle Text, Oracle's integrated full-text retrieval technology, is part of Oracle Database 12c (all editions). Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the Web. Oracle Text can perform linguistic analysis on documents; search text using a variety of strategies including keyword searching, contextual queries, Boolean operations, pattern matching, mixed thematic queries, HTML/XML section searching, etc. Oracle Text excels at mixed queries, i.e. those that involve structured relational attributes as well as text.

Oracle Text can render search results in various formats including unformatted text, HTML with term highlighting, and original document format. Oracle Text supports multiple languages and uses advanced relevance-ranking technology to improve search quality.

## Text is Everywhere

Over the last few decades, organizations have invested heavily in systems that enable rapid access to structured data stored in database systems. However, this data represents a fraction of all corporate information. A far larger volume exists as text - in documents, web pages, manuals, reports, emails, and presentations. These valuable sources of business information are often inaccessible and not managed in a cost-effective manner. Users accessing organization information - whether they are employees visiting an intranet portal or buyers browsing a catalog - need sophisticated support from text search infrastructure to find what they want.

## **Oracle Text**

Oracle Text offers a complete text search solution. Oracle Text is included with all editions of Oracle Database 12c. For users of an Oracle database, Oracle Text eliminates the need to buy and integrate a different Text searching product.

Oracle Text provides specialized text indexes for traditional full text retrieval applications such as - website searching, e-business catalogs, document classification and routing applications, text warehousing, document libraries and archives.

Oracle Text can filter and extract content from different document formats. It supports a large number of document formats including popular ones like the Microsoft Office file formats, the Adobe PDF family of formats, HTML and XML.

Oracle Text offers the best multilingual set of features in the market - supporting search across documents in western languages (English, French, Spanish, German, etc.), Japanese, Korean, Traditional and Simplified Chinese.

As part of Oracle 12c, Oracle Text transparently integrates with and benefits from a number of key enterprise features such as

- Data partitioning (for higher throughput and availability)
- Real application clustering or parallel server (for the highest server scalability)
- Automatic query optimization
- Tools and development environments
- · Administration and manageability

## Integrated security

These aspects of integration are also greatly beneficial to system administrators, who do not have to undergo a paradigm shift to learn to manage and organization's text assets. Oracle Text is a core piece of other Oracle products like Oracle Application Server Portal, Oracle E-Business Suite, Oracle eXchange, Oracle Secure Enterprise Search, and Oracle Content Database.

All of the search capabilities of Oracle's own internal and external web sites are powered by Secure Enterprise Search, which uses Oracle Text as its core search technology.

This section looks at the mechanism for processing text with Oracle Text. This process can be considered as a pipeline (Figure 1). This section discusses each stage, and considers some of the options available at that stage

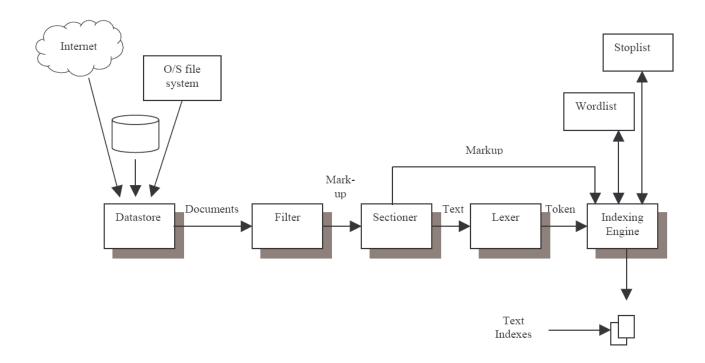


Figure 1: Indexing Architecture

#### Datastore

The datastore defines from where the text to be indexed should be fetched. Provided datastores allow for text which is stored within a database, on a file system, or accessed remotely via the HTTP protocol (the URL datastore). Custom datastores may be defined which fetch the data from a location, protocol or application of the customer's choice.

#### **Default Datastore**

The default datastore is in the database itself. Text may be stored in a VARCHAR2 column (up to 4000 characters), or in a CLOB (Character Large Object) column. Formatted text (such as Word or PDF documents) can be stored in BLOB (Binary Large Object) columns.

## **File Datastore**

Text to be indexed is stored on any file system which is accessible to the database server. The name or path to the file is stored in the database, typically in a VARCHAR2 column.

## **URL Datastore**

The database contains an HTTP protocol URL, and the text to be indexed is fetched directly from the URL at indexing time.

## **User Defined Datastore**

A PL/SQL procedure is specified, which will be called for each row in the table being indexed. The PL/SQL procedure may, in turn, call other language programs such as Java (directly, if running in the database) or C/C++ programs via the EXTPROC external procedures mechanism. This gives the customer complete control over what gets indexed.

#### Filter

The filter stage is responsible for processing "formatted" documents such as Microsoft Office files or PDF documents. The built-in AUTO\_FILTER recognizes all common document formats and can translate them into indexable HTML text.

Application developers may replace the filter stage with their own custom-built filter, or a filter purchased from a third-party.

A custom filter is simply an executable program or script that takes two arguments, the first being the file containing the formatted input text, and the second being the name of the file where the filtered output should be written. If required, a custom filter can call the standard —autorecognize" filter. This allows it to process any file formats unique to the business, but pass on any standard file formats to the standard filter.

#### Sectioner

The sectioner object is responsible for identifying the containing section(s) for each text unit. Typically, these sections will be predefined HyperText Markup Language (HTML) or eXtensible Markup Language (XML) sections. Optionally, the sectioner can process all tags as sections delimiters. For example: <TITLE>XML Handbook</TITLE>. This allows search between tags using the WITHIN operator. Use of the WITHIN is illustrated in the section on XML searching.

#### Lexer

The lexer's job is to separate the sectioner's output into words or tokens. In the simplest case for a Western European language, the lexer just splits text into uninterrupted strings of alphanumeric characters. So the string:

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

would be split into the words, minus any punctuation or special symbols:

aha it s the 5 15 train coming here now

The lexer typically removes stopwords, which are common words defined by the application developer; or taken from a default list. That would likely reduce the list above to:

aha \* \* \* 5 15 train coming \* now

Note the asterisks representing removed stopwords. Although they are not actually indexed, the presence of a stopword at the position is noted in the index. In a search, any stopword will match that word when used as part of a phrase. For example, "kicking the ball" will match "kicking a ball" but will not match "kicking ball".

The set of stopwords may be specified by the application developer, who can also choose to explicitly define all numbers as stopwords.

#### **Lexer Preferences**

There are many options available for fine-tuning the lexer. For example, the developer can choose that an index should be case sensitive or case insensitive, and can choose whether particular characters should split tokens or be indexed as part of them – for example, should "PL/SQL" be indexed as two terms "PL" and "SQL" or the single string "PL/SQL".

Language Specific Functionality

#### Western Languages

- Base Letter Conversion For accented characters, it is possible to "normalize" them to their non-accented form. Thus, a search for "acción" would match "accion" and "acción".
- Alternate Spelling Some languages, such as German, have alternate ways of spelling words with accented characters. For
  example, the words "Muenchen" and "München" are considered identical. If the alternate spelling index option is chosen, then
  both of these words will be indexed as "Muenchen". The same transformation is applied at query time, so a search for either
  term will match "Muenchen" in the index.
- Compound Word Processing Oracle Text contains technology for processing compound words in German and Dutch languages. Such words are broken down into their component forms for the index.

#### Multi-Byte Languages

Symbolic languages do not have space delimited —words" in the same way as western languages. Different rules are required to decide how to index groups of characters. Oracle Text provides special lexers for Chinese, Japanese, and Korean texts.

It is also possible to build multi-lingual search applications. If the language of the documents are known in advance, a particular database column can be designated as the LANGUAGE column at indexing time. If the language of the documents is not known, the AUTO\_LEXER may be used, which provides automatic language recognition, and extensive segmentation and stemming capabilities for multiple languages.

## Indexing Engine

The indexing engine creates the inverted index that maps tokens to the documents that contain them. In this phase, Oracle Text uses - if specified - a *stoplist* where users can specify words or themes which should be exluded from the text index.

The final output of the pipeline is an *inverted index*. This is a list of the words from the document, with each word having a list of documents in which it appears. It is called inverted because it is the inverse of the normal way of looking at text, which is a list of documents where each document contains a list of words.

## Benefits of Integrated Text Search Capability

Oracle Database 12c provides an extensibility framework that enables developers to extend the data types understood by the database kernel. Oracle Text uses this framework to fully integrate the text indexes with the standard Oracle query engine. This means the user has:

- A single repository for all data (text and structured) instead of two. This is easy to maintain, backup, etc.
- Indexes in the same repository. This makes for efficient processing of text and mixed queies.
- A single API for developing applications.
- Integrationb with the Oracle SQL execution engine and query plan optimizer.

The Cost Based Optimizer must be able to choose the fastest execution plan based on the run-time properties of the query. Thus, Oracle Text offers two distinct methods to evaluate a text predicate against a column:

- 1. The extensibility framework can set up the Text index as a row source and pipeline ROWIDs satisfying the predicate to the kernel.
- 2. The extensibility framework can answer the question "does the row with this ROWID satisfy the predicate?" (A *functional invocation* of the index)

To summarize, the advantages of integration are apparent:

- Low Cost
  - Oracle Text is part of all editions of Oracle Database 12c. There are no separate products to buy or integrate.
- High Performance
  - The database will choose the fastest plan to execute queries that involve both text and structure content.
- High Integrity
  - Since text is stored in the database it inherits all the integrity benefits  $\infty$  for example, any update to the database can be reflected to the text search functionality, which means users can get an integrated, holistic view of all their data.
- Low Complexity
  - Text is treated just like structured data. It is easy to develop and integrate text search applications with existing systems.
- Superior Manageability
  - Oracle Text can be managed from standard enterprise management tools, leveraging commonly available administrators' skills.
- Security
  - Oracle Text leverages the security features of the database.

In this section we describe in detail the main features of Oracle Text.

#### **Index Types**

Oracle Text provides three types of indexes that cover all text search needs: standard, catalog, and classification. Table 1 shows an overview of the three index types.

- 1. Standard index type for traditional full-text retrieval over documents and web pages. The context index type provides a rich set of text search capabilities for finding the content you need, without returning pages of spurious results.
- Catalog index type the first text index designed specifically for eBusiness catalogs. The ctxcat catalog index type provides flexible searching and sorting at web-speed.
- 3. Classification index type for building classification or routing applications. The ctxrule index type is created on a table of queries, where the queries define the classification or routing criteria.

Index Type	Application Type	Query Operator
CONTEXT	Use this index to build a text retrieval application when your text consists of large coherent documents You can index documents of different formats, such as MS Word, HTML, XML or plain text. With a CONTEXT index you can customize your index in a variety of ways.	CONTAINS
CTXCAT	Use this index type to index small text fragments such as item names, prices and descriptions that are stored across columns. Particularly suited to mixed queries.	CATSEARCH
CTXRULE	Use a CTXRULE index to build a document classification application. The CTXRULE index is an index created on a table of queries, where each query has a classification.  Single documents (plain text, HTML or XML) can be classified using the MATCHES operator.	MATCHES

**Table 1: Index Type Overview** 

## Substring and Prefix indexes

Oracle Text also provides substring and prefix indexes with the CONTEXT index type. Substring indexing improves performance for left-truncated or double-truncated wildcard queries. Prefix indexing improves performance for right truncated wildcard queries.

#### Maintaining Indexes and Synchronization

In 12c users can specify at index creation the index update preference: manually, on commit, or at regular intervals. Users can also specify a transactional text index, where documents are searchable immediately after being inserted or updated. Note that the catalog index type - designed specifically for the short pieces of text typically found in eBusiness catalogs – is always transactional and needs no synchronization.

## Parallel Indexing

Parallel indexing can take advantage multiple CPU cores.

Parallel index creation is useful for

- · Performance improvement
- Data Staging

- Rapid initial deployment of applications based on large data collections
- Application testing, when users need to test different index parameters and schemas while developing an application

The following example creates a text index with degree 3:

```
CREATE INDEX myindex ON docs(tk)
INDEXTYPE IS ctxsys.context PARALLEL 3;
```

Figure 2 shows how the text index creation works in parallel. The Oracle Parallel facility splits up the table into pieces (portions) according to the degree of parallelism. Each parallel slave works on one portion of the table.

## Creating a Text Index in Parallel

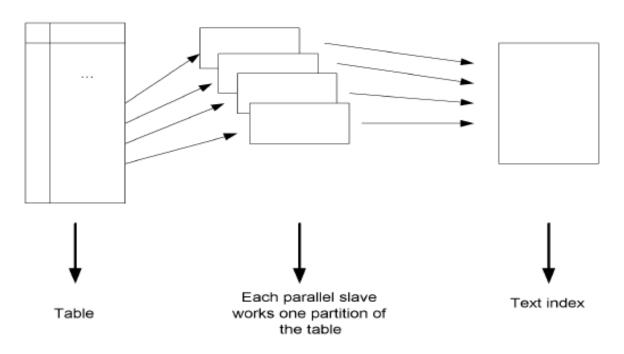


Figure 2: Creating a text index in parallel

## Locally Partitioned Indexes

It is possible to create a text index on a local partition basis – effectively each partition of the base table has its own index, and queries which span two or more partitions will access all the necessary local indexes. The major benefits of this approach are:

- Managability. An administrator can decide how to partition the index, which partitions are online/offline, which partitions to backup, etc.
- Performance. There is a tremendous improvement in scalability under certain circumstances with locally partitioned indexes.

## **Query Operators**

Oracle Text can intelligently process search queries using several strategies:

- Keyword searching. Searching for keywords in a document. User enters one or more keywords that best describe the query.
- Context queries. Searching for words in a given context. User search for text that contains words near to each other.

- Boolean operations. Combining keywords with Boolean operations. User can express a query connecting Boolean operations to the keywords.
- Linguistics features. Using fuzzy and other natural language processing techniques. User searches for text that is about something.
- Substring matching. Retrieval of text that contains a certain property. User searches for text that contains words that contain a string.

Table 2 shows an incomplete list of query operators:

Operator	Description	
ABOUT	Performs a theme search where available, and increases the number of relevant documents returned from the query	
ACCUMULATE (,)	Searches for documents that contain at least one occurrence of any of the query terms. Increases relevance as more terms are found.	
AND (&)	Searches for documents which contains all the query terms	
Broader Tem (BT, BTG, GTP, BTI)	Expands a query to include the term that has been defined in a thesaurus as a broader or higher level term.	
EQUIValence (=)	Specifies alternate substitution terms in a query	
FUZZY	Expands queries to include words which are spelled similarly, or sound similar to the specified term.	
HASPATH	Finds all XML documents which contain a specified section path	
INPATH	Searches within a particular path in an XML document	
MDATA	Queries MDATA (MetaDATA) sections	
MINUS (-)	Lower the relevance of documents that contain a particular term, but do not necessarily exclude them	
MNOT	Mild NOT – finds words which are not part of a specified phrase. For example "York MNOT New York" will find occurrences of the single word "York" even if the phrase "New York" appears elsewhere in the document.	
Narrow Term (NT, NTG, NTP, NTI)	Expands a query to include all the terms which have been defined in a thesaurus as the narrower or lower level terms for a specified term	
NDATA	Performs a "name search" on an NDATA section when the exact name is not known	
NEAR (;)	Returns a score based on the proximity of two or more query terms	
NOT (~)	Exclude documents which contain a particular term (must be used in the form "term1 NOT term2" – you cannot just use "NOT term1")	
OR ( )	Find documents which contain at least one occurrence of any of the query terms.	
Preferred Term (PT)	Replaces a term in a query with the preferred term that has been defined in a thesaurus for the term	
Related Term (RT)	Replaces a term in a query with the related term that has been defined in	

	a thesaurus for the term	
SDATA	Searches within a Structured DATA section	
Soundex (!)	Expands queries to include words which have similar sounds	
Stem (\$)	Searches for terms which have the same linguistic root as the query term.	
Stored Query Expression (SQE)	Calls a stored query expression created with the CTX_QUERY.STORE_SQE procedure	
SYNonym	Expands a query to include all the terms that have been defined in a thesaurus as synonyms for the specified term	
Threshold (>)	Eliminates documents in the result set that score below a threshold number. This operator at the query term level selects a document based on how a term scores in the document.	
Translation Term (TR)	Expands a query to include all foreign language terms defined in a thesaurus	
Translation Term Synonym (TRSYM)	Expands a query to include all the defined foreign equivalents of the query terms, thesynonyms of query term, and the foreign equivalents of the synonyms.	
Top Term	Replaces a term in a query with the top term that has been defined for the term in the standard hierarchy in a thesaurus.	
Weight (*)	Multiplies the score by the given factor (topping out at 100).	
WITHIN	Narrows a query to a document section	

**Table 2: CONTAINS Query Operators Summary** 

#### Internationalization

As organizations operate globally, multilingual features become important for worldwide distributed operations. Enterprises portals, libraries or content management systems need to search across content that might be authored in different languages or encoded in different character sets. With the rise of XML, multilingual metadata and content search capabilities have come into sharper focus.

Oracle Text supports all Oracle NLS character-sets. For example, ASCII, UTF- 8, JA165JIS, GBK, BIG5, etc. Oracle Text supports search across documents in western languages (English, French, Spanish, German, etc.), Japanese, Korean, Traditional, and Simplified Chinese. With these multilingual features, users can develop cross-language search applications and:

- Mix languages within a document collection (e.g. Chinese and English documents).
- Use English to query e.g. Chinese terms or vice versa. The following query finds products whose description contains "monitor" or its Chinese equivalents.

```
select score(1), product_id, product_name
from product_information
where contains (product_description,
    'TRSYN(monitor, Chinese)',1)>0
order by score(1) desc
```

(Note that TRSYN requires the loading of a translation thesaurus, Oracle Text does not ship with any such thesaurus).

Oracle Text provides highlighting, markup, snippet, themes, and gists as the main document services. This type of services can be very useful for browsing strategies and for document presentation. They also provide informative feedback to the user.

#### Highlighting

The highlight service takes a query string, fetches the document contents, and shows you which words in the document cause it to match the query.

## Markup

Markup takes the highlight service one step further, and produces a text version of the document with the matching words marked up.

#### Snippet

This document service is useful for producing a short piece of text with keywords highlighted. This is a very popular technique that gives the user an idea of what the document is about before opening it. Figure 4 shows an example of you can use this service for presenting search results.

## Oracle Developer Suite Features and Components

Oracle development tools - Oracle Developer Suite, Oracle JDeveloper, Forms, Designer, SCM, Reports, Discoverer, and more. www.oracle.com/tools/tools\_editions.html - 37 Kb - Cached Links

#### Oracle Discoverer Software

COMMUNITY Oracle Discoverer Standalone downloads Oracle Discoverer Desktop Edition 9.0.4 (Part of Oracle Developer www.oracle.com/technology/software/products/discoverer/index.html - 21 Kb - Cached Links

#### Oracle Application Server

Oracle Application Server 10g is an application platform suite that offers full support for J2EE, high-speed caching, rapid application enterprise portals, identity management, business www.oracle.com/appserver/index.html - 44 Kb - Cached Links

Figure 4: Document Snippet in Search Results Presentation

To supplement traditional text searching capabilities, Oracle Text provides advanced linguistic features. The linguistic features in the document services enable you to generate document themes or theme summaries, on-demand and per-document.

## Theme Extraction

A *theme* provides a snapshot that describes what the document is about. Rather than searching for documents that contain specific words or phrases, users can search for documents that are about a certain subject, even if that subject is not mentioned explicitly in the document. Theme queries return a hit list of those documents that are about the requested subject, along with a score that indicates how strongly each document reflects to the subject in question.

## Gist Generation

A Generic Gist is a summary consisting of the sentences or paragraphs, which best represent the overall subject matter of the document. You can use the Generic Gist to skim the main content of the text, or assess your interest in the text's subject matter. You can generate paragraph-level or sentence-level gists. You can also generate "Point of View" gists, which shows the section of the document most relevant to one of the extracted themes of the document.

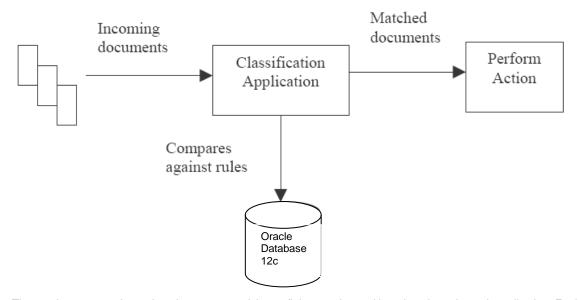
#### AdvanCed Features

## **Classification and Clustering**

A document classification application is one that classifies an incoming stream of documents based on their content. These applications are also known as document routing or filtering applications. For example, an online news agency might need to classify its incoming stream of articles as they arrive into categories such as politics, economy, or sports.

Oracle Text offers a number of techniques for helping users chose the best strategy for classifying content.

The rule-based approach consists of users defining categories (rules) that explain why documents belong to them. Then with the CTXRULE index type, the application indexes the rules (queries) that define classifications or routing criteria. When documents arrive, the MATCHES operator can be used to categorize each document. Figure 8 shows the main structure of a classification



The previous approach requires that user spend time refining queries and learning about the entire collection. For large data sets this approach doesn't scale up.

The classification training approach consists of users providing a set of sample documents from certain subjects. The CTX\_CLS package then takes the training set and generates automatically rules that would identify documents in that subject area. There are two methods available: decision trees and support vector machines (SVM).

Converseley to classification, clustering is the *unsupervised* classification of patterns into groups. Oracle Text offers the CTX\_CLS.CLUSTERING package for building clusters. The package automatically clusters a set of documents according to their semantic meanings. Each cluster contains a subset of documents of the collection. The document in a cluster is believed to be more similar with each other inside the cluster than with outside documents. There are two methods available: k-means for flat partitioning, and hierarchical clustering.

## Knowledge Base

Oracle Text's knowledge base, contains over 400,000 concepts from very broad domains classified into 2000 major categories. These categories are organized hierarchically under six top terms: business and economics, science and technology, geography, government and military, social environment, and abstract ideas and concepts. Concept classification, choice of categories, and the hierarchical organization are all carefully designed for their usefulness in information retrieval rather than ontological purity, with a special focus on avoiding problems of semantic ambiguity. Users can extend and customize this knowledge base by adding new terms or redefining existing ones. For example, users can import a medical thesaurus and later extend the knowledge base.

## Using Oracle Text

In this section, we present some syntax and usage samples for the major Oracle Text features described above.

#### **Creating Indexes with Oracle Text**

describe product information

Let's assume the following table contains some typical product information:

_		
Name	Null?	Туре
PRODUCT_ID	NOT NULL	NUMBER(6)

```
PRODUCT_NAME VARCHAR2 (50)
PRODUCT_DESCRIPTION VARCHAR2 (2000)
CATEGORY NUMBER (2)
PRODUCT_STATUS VARCHAR2 (20)
LIST PRICE NUMBER (8,2)
```

We would like to create a text index on the PRODUCT\_DESCRIPTION column to make it searchable. The index creation is a SQL statement:

```
CREATE INDEX description_idx ON
product_information(product_description)
INDEXTYPE IS CTXSYS.CONTEXT
```

## Searching is also a SQL statement:

```
SELECT score(1), product_id, product_name
FROM product_information
WHERE CONTAINS
(product_description, 'monitor NEAR "high resolution"', 1)>0
ORDER BY score(1) DESC;
```

As discussed earlier, the text index structures are stored in the database. The Oracle Text index consists of four basic tables, referred to as the \$I, \$K, \$N and \$R tables respectively. Other tables are used when particular indexing options are turned on. The tables exist within the schema of the text index owner, and have names concatenated from DR\$, the name of the index, and the suffix (e.g. \$I).

The \$I table consists of all the tokens that have been indexed, together with a binary representation of the documents they occur in, and their positions within those documents. Each document is represented by an internal DOCID value.

The \$K table is an index-organized table (IOT) which maps internal DOCID values to external ROWID values. Each row in the table consists of a single DOCID/ROWID pair. The IOT allows for rapid retrieval of DOCID given the corresponding ROWID value.

The \$R table is designed for the opposite lookup from the \$K table - fetching a ROWID when you know the DOCID value.

The \$N table contains a list of deleted DOCID values, which is used (and cleaned up) by the index optimization process.

These tables are created for all CONTEXT indexes.

All sub-tables are created in the index-owners schema. They may be viewed using normal SQL commands, for example:

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME

DR$DESCRIPTION_IDX$I

DR$DESCRIPTION_IDX$K

DR$DESCRIPTION_IDX$N

DR$DESCRIPTION_IDX$R

PRODUCT INFORMATION
```

We can also look at the index name in the usual views:

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME

DR$DESCRIPTION_IDX$I

DR$DESCRIPTION_IDX$K

DR$DESCRIPTION_IDX$N

DR$DESCRIPTION_IDX$R

PRODUCT_INFORMATION
```

## **Optimizer Hints**

We can also "hint" the database optimizer to improve query performance if we know ahead of time what plan is best:

```
SELECT /*+ index product_information description_idx */
score(1), product_id
FROM product_information
WHERE CONTAINS (
product_description, 'monitor NEAR "high resolution"', 1) > 0
AND list price < 500;</pre>
```

The last example uses standard SQL to mix a content-based predicate with a classical relational predicate.

We can see the explain plan for any type of query. For example:

```
SELECT score(0) scr, id, author, title
FROM docs
WHERE CONTAINS(text, 'money', 0) > 0 and id > 16
ORDER BY scr DESC;
Rows
      Execution Plan
     SELECT STATEMENT GOAL: CHOOSE
0
      SORT (ORDER BY)
0
      TABLE ACCESS (BY INDEX ROWID) OF 'DOCS'
0
     BITMAP CONVERSION (TO ROWIDS)
0
      BITMAP AND
      BITMAP CONVERSION (FROM ROWIDS)
Ω
0
     SORT (ORDER BY)
      DOMAIN INDEX OF 'DOCS TEXT'
0
0
      BITMAP CONVERSION (FROM ROWIDS)
0
       SORT (ORDER BY)
       INDEX (RANGE SCAN) OF 'SYS_C001220' (UNIQUE)
```

We mentioned earlier that Oracle Text supports theme or concept-based retrieval using the ABOUT operator which extracts themes from free text queries to match against themes in the inverted index. For example a user can retrieve news articles about trains even if none of the documents contains the word "train".

```
SELECT id title
FROM news_table
WHERE CONTAINS(article'about(train) ' ) > 0;
```

```
Id Title
334 Rail Transportation in Europe
```

All theme based features in Oracle Text - themes, ABOUT queries, gists, ABOUT query highlighting, and hierarchical query feedback - depend on the internal knowledge base.

The power of the Oracle database makes possible the construction of multi-domain queries. For example, find the number of patients older than 50, that live within 35 km of Toronto, have had a family medical history of cancer and who smoke, and get their chest x-ray.

```
SELECT count(p) p.age p.xray
FROM patients p cities c
WHERE p.age > 50
AND c.name = 'Toronto'
AND SDO_WITHIN_DISTANCE(p.loc, o.loc '<= 35 km')
AND Contains(p.medical history 'smoke AND cancer')>0
```

#### **SDATA Sections**

SDATA (**S**tructured **DATA**) sections were introduced in Oracle Text 11*g*. These sections are embedded in the text of a document – like field or zone sections – but unlike previous sections they may contain character, numeric or date information and may be searched using operators such as "greater than", "less than" and "between" as well as equality searches.

Here's an example of a query which makes use of SDATA query operators:

```
SELECT item_id FROM items WHERE

CONTAINS (description, 'racing and

SDATA(itemtype=''BOOK'') and SDATA(price<10)') > 0

ORDER BY price DESC
```

Note that we're now doing a range search as part of the text query. This is an entirely new feature, and one that will aid in many situations.

#### **Composite Domain Indexes**

Composite Domain Indexes use the same underlying technology as SDATA sections, but in an easier-to-use and more standard fashion.

First, a word on the terminology. A 'domain index' is a type of index for use with a particular type of data (in our case, textual data). A composite index in normal Oracle terms is an index that covers more than one column. So a Composite Domain Index (CDI, for short) is a extension of the usual domain index to cover multiple columns.

Let's look at a typical "mixed" query which searches a text index and two structured colums:

```
SELECT item_id FROM items WHERE

CONTAINS (description, 'racing') > 0

AND itemtype = 'BOOK'

AND price < 10

ORDER BY price DESC
```

To create appropriate indexes for this query in previous versions we may have run the following SQL commands:

```
CREATE INDEX typeind ON items (itemtype)

CREATE INDEX priceind ON items (price)

CREATE INDEX descind ON items (description) INDEXTYPE IS ctxsys.context
```

In Oracle Database 12c we can do all we need with a single call:

```
CREATE INDEX compind ON items (description)
INDEXTYPE IS ctxsys.context
FILTER BY itemtype, price
SORT BY price
```

Oracle will now store price and itemtype information inside the text index. There is no need to modify our query (as we had to with SDATA) – the optimizer will realise that the query can be satisfied by the text index alone and will "push down" the filtering of rows into the text index processor, to get the right itemtype and price to satisfy the query. It will also request the text index to return rows correctly sorted, which is considerably more efficient than fetching all the rows from the database and sorting them afterwards.

#### XML Support

XML features include the operator WITHIN, nested section search, search within attribute values, mapping multiple tags to the same name, path searching using INPATH and HASPATH operators.

Let's use the following XML example to demonstrate Oracle Text's features.

```
<?xml version="1.0"?>
<FAQ OWNER="Billy Text">
<TITLE>Oracle Text FAQ</TITLE>
<DESCRIPTION>
Everything you always wanted to know about Text</DESCRIPTION>
<QUESTION>What is Oracle Text?
</QUESTION>
<ANSWER>
Oracle Text uses standard SQL to index search and analyze text and documents stored in the database files or websites.
</ANSWER>
</FAQ>
```

## This allows the search:

```
SELECT title description

FROM FAQTable

WHERE CONTAINS(text'Oracle WITHIN QUESTION')> 0;
```

## You can also search by attribute values:

```
SELECT title description

FROM FAQTable

WHERE CONTAINS(text'Billy WITHIN FAQOOWNER')> 0;
```

#### Path searching can be done as:

```
SELECT title description
FROM FAQTable
WHERE CONTAINS(text'Oracle INPATH(FAQ/TITLE)')> 0;
```

## Path testing, which determines if a path exists, looks like:

```
SELECT title description
FROM FAQTable
WHERE CONTAINS(text'HASPATH(FAQ/TITLE/DESCRIPTION)')> 0;
```

#### Searching for content and structure in XML documents

Traditionally databases have allowed to search their content or their structure, but not both at the same time. Oracle provides unique features that enable querying for content and structure at the same time.

Oracle Text has two SQL functions  ${\tt existsNode}$  and  ${\tt extract}$  which operate on XMLType values:

- existsNode(): given an XPath expression, checks if the XPath applied over the document can return any valid nodes.
- extract(): given an XPath expression, applies the XPath to the document and returns the fragment as a XMLType.

We can combine the above functions with all the power of the Text query language for the content search. For example, we can search for those FAQs that contain "standard or SQL" in the answer tag and display the question.

```
select f.faq.extract('/FAQ/QUESTION/text()').getStringVal()
from faq f
where contains(faq, 'standard or SQL INPATH(FAQ/ANSWER)')>0
```



#### CONNECT WITH US



blogs.oracle.com/oracle



facebook.com/oracle



twitter.com/oracle



oracle.com

## Integrated Cloud Applications & Platform Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0517

White Paper Title May 2017 Author: Roger Ford Contributing Author: Asha Makur

