New Features in Oracle Text with Oracle Database 11*g* Release 1

An Oracle White Paper
May 2007

ORACLE

New Features in Oracle Text with Oracle Database 11*g* Release 1

INTRODUCTION

Oracle Text is the leading text searching, retrieval and management system to be integrated into a database environment. With Oracle Database 11g Release 1, Oracle Text introduces new features which aim to keep it in the leading position. These new features may be grouped into four target areas

- Performance
- Minimization of application downtime
- Internationalization
- Ease of Maintenance

This paper briefly discusses the new features. A companion paper will delve into more detail for each feature, with examples of them in use.

Summary of New Features

- Performance
    - Improved query performance and scalability
    - New SDATA sections and Composite Domain Indexes
    - Increased number of partitions
    - New parallel Optimize Index "rebuild" mode

    - User-defined scoring mechanism
- Minimization of application downtime
    - Incremental indexing
    - Online index recreation
- Internationalization
    - New AUTO_LEXER for sophisticated handling of many languages
    - INDEX_STEMS feature now available for many more languages
- Ease of Maintenance
    - Enterprise Manager is extended with Oracle Text Manager, which allows you to
        - Monitor health of text indexes
        - Modify index settings
        - Generate index-level statistics about disk space, fragmentation, garbage, frequency of words, and more
        - Diagnose problems, and resume failed operations
        - Manage logs
    - Usage tracking (allows you to quickly establish which features are in use for a specific installation)

PEFORMANCE IMPROVEMENTS

SDATA Sections and Composite Domain Indexes

These features are very closely related so they need to be considered together. Indeed they can be considered to be two facets of the same feature. Together, they are designed to improve the performance of "mixed queries" – queries which have a text search part and a structured part. For example, the query:

```
SELECT item_id FROM items WHERE
CONTAINS (description, 'madonna') > 0
AND itmtype = 'BOOK'
AND price < 10
ORDER BY price DESC
```

The issue with this query is that multiple indexes are used. With necessary btree/bitmap indexes created and proper execution plan, this query typically can perform very efficiently if either the contains() predicate is very selective or the structure predicate is very selective (although we still need to do base table access

on the query hits for sorting). However, if none of the predicates is selective (e.g. the *contains()* predicate has 5K hits, itmtype predicate has 10K hits, and price predicate has 30K hits) and query result is selective, then usually the best execution plan is to either drive from the Text index or do bitmap AND operation with the Text index. Even with such optimal execution plan, the query can still perform very poorly due to extra overhead of internal text index docid values to external rowid resolution and base table accesses. These queries are never likely to be as fast as a simple text-only query.

In Oracle Database 10*g* Release 1 we introduce MDATA (for *M*eta*DATA*) sections. These were designed for short character fields which would be indexed "as a whole" inside the text index. This would allow us to rewrite the query above as something like:

MDATA Sections were introduced in Oracle 10g. They allow you do equality searches against short text fields.

```
SELECT item_id FROM items WHERE
CONTAINS (description, 'madonna and MDATA(itmtype, BOOK') > 0
AND price < 10
ORDER BY price DESC
```

That's great – as far as it goes. We have potentially removed many unnecessary docid to rowid resolutions, and the base table access to evaluate "itmtype='BOOK'" predicate, since we can get itmtype=BOOK from the text index. However, it doesn't solve the problem completely:

- We can only do equality searches, we can't do "price < 10" with MDATA
- We can't use it for sorting.

SDATA Sections

SDATA sections are defined, and used, in a manner similar to MDATA sections. Unlike MDATA, they can be used for range searching on numeric or date fields, as well as equality searching.

In Oracle Database 11*g* Release 1 we can instead do

```
SELECT item_id FROM items WHERE
CONTAINS (description, 'madonna and
SDATA(itmtype="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 our original query again:

```
SELECT item_id FROM items WHERE
CONTAINS (description, 'madonna') > 0
AND itmtype = '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 (itmtype)

CREATE INDEX priceind ON items (price)

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

In Oracle 11*g* Release 1 we can do all we need with a single call:

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

Oracle will now store price and itmtype 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.

Note: A Composite Domain Index created as such is not meant to replace btree indexes for structured-only queries. For example,

select * from items where itmtype='BOOK';

may give you a different result than

select * from items where contains(description, 'SDATA(itmtype="BOOK")')>0;

because SDATA sections are part of Text index, only synchronized documents are indexed. Therefore, the first query above will return qualified rows that have not been synchronized, but the second query won't.

Whenever, a Text index is created with SDATA sections (whether they are implicitly using FILTER BY or ORDER BY clauses in CREATE INDEX or explicitly created using ctx_ddl.add_sdata_section()), a new $S index table will be

created for looking up SDATA section values. The $S is an Index-Organized Table (IOT), and must not be created with an overflow segment. So an error will be raised during CREATE INDEX is if you specify a $S storage preference with PCTTHRESHOLD clause or OVERFLOW clause. Also, the $S table must be created on a tablespace with db block size >=4K.

Increased Number of Partitions

Partitioning database tables and creating locally partitioned Text indexes on them is a common way of maximizing the performance of text indexes where queries may be optimized to hit a limited number of partitions.

In Oracle Database 10*g* the maximum number of partitions that could be used was 9999. In Oracle Database 11*g* this limit is relaxed, and the limit for text index partitions is now the same as the limit for table partitions – 2^{20} - 1, or 1,048,575.

New parallel Optimize Index "rebuild" mode

Optimizing your index in "rebuild" mode means you can build the best possible index, if you have the space available since the optimize takes a complete copy of the index and packs it tightly into new blocks. This option now supports parallel optimize.

MINIMIZATION OF APPLICATION DOWNTIME

Incremental Indexing

The **CTX_DDL.SYNC_INDEX** procedure supports a new parameter **MAXTIME** This allows you to limit the duration of any SYNC call. In conjunction with **CTX_DDL.POPULATE_PENDING,** this allows you to create an index gradually at quiet times for your system. This feature is particularly useful in conjunction with the next feature, online index recreation.

Online Index Recreation

Sometimes it may be desirable to drop and recreate a text index from scratch.- for example if you want to change indexing options such as break characters or case sensitivity.

In the past, this would mean that your text index would be unavailable for the full time required to build the new index – which for a very large index might be measured in days.

In Oracle Database 11*g* you can create a "shadow" text index. This can be built while the original index is still in use, and when the index build is complete, the original index can be exchanged for the newly built shadow index. As soon as this is done, queries will automatically transition to the new index.

This can all be done automatically using the new procedure **CTX_DDL.RECREATE_INDEX_ONLINE** or for more control, it can be run as a number of distinct steps. This would allow you to create the new index incrementally, so the work can take place gradually during the quiet periods for your system, the indexing load will not impact on the system performance during busy periods.

INTERNATIONALIZATION

New AUTO_LEXER

In previous versions, the WORLD_LEXER provided basic indexing capabilities for many languages. It could not, however, perform stemming or segmentation. Stemming is extremely important in languages which are highly inflected – that is, words take on many forms depending on case, gender, tense, etc. A single word may have tens or hundreds of forms, all of which need to be identified in text during indexing so that the base form of the word may be stored in the index for later retrieval. Segmentation is necessary to identify the components of compound words.

AUTO_LEXER can perform automatic identification of different languages, or the language of each document can be specified by means of a LANGUAGE column in the usual way. Automatic recognition works best on longer documents (several paragraphs or more), for very short passages it is better to manually provide the language setting if possible.

AUTO_LEXER supports segmentation and stemming for 32 languages. Of these, 23 support context-sensitive stemming, which enables the lexer to process words appropriately by knowing, for example, whether a word is a verb or a noun by analyzing the text around it.

The table below shows languages supported by the AUTO_LEXER. Those in bold support context-sensitive stemming.

> **Arabic**, **Catalan**, Simplified Chinese, Traditional Chinese, **Croatian**, **Czech**, **Danish**, **Dutch**, **English**, **Finnish**, **French**, **German**, **Greek**, **Hebrew**, **Hungarian**, **Italian**, Japanese, Korean, Bokmal, **Nynorsk**, Persian, **Polish**, **Portuguese**, **Romanian**, **Russian**, **Serbian**, **Slovak**, **Slovenian**, **Spanish**, **Swedish**, Thai, Turkish

INDEX_STEMS feature now available for many more languages

Want to have the full control of the BASIC_LEXER but still have stemming for a variety of languages? No problem. The INDEX_STEMS feature can now be used with all the languages in bold in the list above.

EASE OF MAINTENANCE

Enterprise Manager Integration

The "Text Manager" section of Enterprise manager now has a comprehensive range of facilities, enabling the DBA to monitor the state and status of Oracle Text indexes, and run a variety of maintenance tasks on those indexes.

Usage Tracking

This feature is provided largely for use in conjunction with Oracle Support. By a simple table lookup, a support analyst is able to quickly and easily identify exactly which features are in use in a particular Oracle Text installation.

It will doubtless also prove useful to new DBAs or consultants who wish to quickly familiarize themselves with an existing installation.

New Features in Oracle Text with Oracle Database 11g Release 1
August 2007
Author: Roger Ford
Contributing Authors: Wesley Lin, MohammadFaisal

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com