Tuning Oracle Text for Rapid Document Retrieval
- A Case Study

Sandi Wilson
Senior Oracle DBA
Summit Information Systems, a Fiserv Company
Speaker Background

- Corvallis, Oregon, US
- BS in Computer Science, UCF
- Oracle DBA 10+ years (7.3.4+)
- Current Employment : Fiserv
  - Banking applications
  - Financial document archiving and retrieval
Agenda

- What is Oracle Text?
- Vocabulary
- Scope
- Structures
- Parameters
- Packages
- Impacting Performance
- Q&A
What is Oracle Text?

- Earlier versions
  - SQL*TextRetrieval
  - TextServer
  - ConText Option/Cartridge
  - InterMedia Text

- Robust text search and document classification

- Supports a wide variety of source document types
What is Oracle Text? (cont.)

- **Query methods include:**
  keywords, contexts, themes, word stems, pattern matching, fuzzy matching, HTML/XML sections

- **Output formats include:**
  original document format, unformatted text, HTML with keyword highlighting, information visualization
Vocabulary

- **OT**
- **Token/Search token**
  - a text string
- **Docid**
  - an identifier assigned by and used within the OT index to reference a source document
- **Preferences**
  - Parameter classes
Scope

- CONTEXT index type
- Keyword & XML section searches
- 10gR2 Standard edition features
- Non-static source: 50-500GB
Index Creation

- **Use:**

```sql
CREATE INDEX mytext_idx
ON Documents(text)
INDEXTYPE IS ctxsys.Context PARMETERS
(‘datastore mydatastore storage mystorage section group
mysectiongroup lexer mylexer memory 400M’);
```

- Creates a set of internal index tables using defined or default index preferences
Index Creation (cont.)

- Parses and tokenizes text according to lexer parameters
- Filters by removing tokens identified in stopword list
- Source documents are processed in batches and appended to the index
Internal Index Structure

- **DR$<\text{index\_name}>$I**
  - “Token” table
  - List of tokens and where they occur
  - BLOB: token_info
- **DR$<\text{index\_name}>$X**
  - B-tree index on $I$ table
- **DR$<\text{index\_name}>$N**
  - “Deleted” docids table
  - List of invalid docids
Internal Index Structure (cont.)

- **DR$<\text{index}_\text{name}>$K**
  - Index-organized “Lookup” table
  - Maps external (source data) rowids to internal index docids

- **DR$<\text{index}_\text{name}>$R**
  - Index-organized “Reverse Lookup” table
  - Maps internal index docids to external rowids
Index Parameters

- **Storage**
  - Define storage clauses for DR$ objects
- **Datastore**
  - Identifies where source text is located
- **Lexer**
  - Rules for converting text to tokens
- **Stop words**
  - High frequency words, excluded from indexing
- **Section Groups**
  - HTML, XML, AUTO
Index Packages

- CTX_ADM
- CTX_DDL
- CTX_OUTPUT
- CTX_REPORT
CTX_ADM

- **Set_Parameter**
  
  - Use:
    
    ```
    CTX_ADM.SET_PARAMETER ('max_index_memory','400M');
    ```

  - default_index_memory
  
  - log_directory

  - Default_<preference> (i.e. datastore, lexer, stoplist, etc)
**CTX_DDL**

- **SYNC_INDEX**
  - Use:
    
    ```
    CTX_DDL.SYNC_INDEX('mytext_idx','100M');
    ```
  - Processes newly added or modified documents
    - **CTX_USER_PENDING** view
  - Source documents are processed in batches and appended to the index
**CTX_DDL** (cont.)

- **Optimize_Index**
  - Use:
    ```java
    CTX_DDL.OPTIMIZE_INDEX('mytext_idx', 'fast');
    ```
  - Defragments token info
  - Removes references to deleted documents
  - Modes: Fast (no GC), Full, Token, Token Type, Rebuild
CTX_DDL (cont.)

- Create_Preference
- Add_Stopword
- Add_Stop_Section
- Add_MDATA_Section
CTX_OUTPUT

- Start_Log/Stop_Log
- Add_Event/Remove_Event
  - EVENT_INDEX_PRINT_ROWID
  - EVENT_OPT_PRINT_TOKEN
  - EVENT_INDEX_PRINT_TOKEN
- Start_Query_Log/Stop_Query_Log
- Add_Trace/Remove_Trace
**CTX_REPORT**

- Use to generate reports on index information and query activity

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIBE_INDEX</td>
<td>Creates a report describing the index.</td>
</tr>
<tr>
<td>DESCRIBE_POLICY</td>
<td>Creates a report describing a policy.</td>
</tr>
<tr>
<td>CREATE_INDEX_SCRIPT</td>
<td>Creates a SQL*Plus script to duplicate the named index.</td>
</tr>
<tr>
<td>CREATE_POLICY_SCRIPT</td>
<td>Creates a SQL*Plus script to duplicate the named policy.</td>
</tr>
</tbody>
</table>
##CTX_REPORT (cont.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX_SIZE</td>
<td>Creates a report to show the internal objects of an index, their tablespaces and used sizes.</td>
</tr>
<tr>
<td>INDEX_STATS</td>
<td>Creates a report to show the various statistics of an index.</td>
</tr>
<tr>
<td>QUERY_LOG_SUMMARY</td>
<td>Creates a report showing query statistics</td>
</tr>
<tr>
<td>TOKEN_INFO</td>
<td>Creates a report showing the information for a token, decoded.</td>
</tr>
<tr>
<td>TOKEN_TYPE</td>
<td>Translates a name and returns a numeric token type</td>
</tr>
</tbody>
</table>
Impacting Performance

- Parameters
- Creation
- Synchronization
- Optimization
- Query tuning
- Other
Index Parameters and Performance

- Define large `max_index_memory` and `default_index_memory`
- Choose appropriate lexer preferences
  - Avoid tokenizing useful composite strings
    Ex:  `john.smith@example.net`
- Define comprehensive stopword lists
  - Avoid indexing unhelpful tokens
    Ex:  “account”, “credit”, “authorize”
  - Use `CTX_REPORT.INDEX_STATS` to identify new stopword candidates
Index Creation and Performance

- Maximize memory allocation for index creation
  - If possible, reduce SGA in order to increase max_index_memory
- Define MDATA sections to avoid costly “mixed” queries
- Use NOLOGGING storage parameter during creation, alter to LOGGING after index is created
Synchronization and Performance

- Sync infrequently
- Maximize memory allocation for sync
Optimization and Performance

- Optimize after index creation
- Optimize frequently
- Rebuild mode achieves best $I$ defragmentation and space consolidation
- Use lightweight ‘token’ and ‘token type’ modes for targeted improvements
- Implement an aggressive optimization schedule
Query Tuning

- Use a single CONTAINS clause
- Do not nest CONTAINS clauses in inner loop
- When possible, use MDATA sections instead of mixed queries
Other Performance Considerations

- Gather statistics on the index, but not the internal tables
- Use KEEP pool for internal tables
Summary

- Determine appropriate parameters for your OT application
- Allocate maximum memory for OT processing
- Synchronize infrequently
- Optimize frequently
- Tune the text queries
References

- Faisal, Mohammed, “Real World Performance Part III - Oracle Text”, OOW 2006
- Kyte, Thomas, *Expert One-On-One*
Q&A