Oracle Text 12c
Roger Ford, Principal Product Manager
Program Agenda

- Introduction to Oracle Text
- Oracle Text 12c
  - New Features
The Challenge: Text is Everywhere

- Unstructured data exceeds structured data X10
- Information warehouses
- Web Sites
- Help-desks
- Research databases
Text Alongside the Database

- Expensive to develop
- Different vendors
- Separate APIs
- Separate Repositories
- Integration poses scalability and performance issues
So why not use the built-in text engine?

- One vendor, one support organization, one API
- Full-featured text search in a SQL environment
- All indexes and metadata stored in Oracle tables
  - Security
  - Reliability
  - High availability
- Low-level kernel integration provides best performance for “mixed” queries
- Scalability via RAC and Exadata
Oracle Text Is ...

- A SQL level toolkit for developing text-centric applications
  - Providing indexes for free-text (keyword) searching
    - Much improved version of “like” operator
  - And text summarization: snippets, themes, entity extraction
- Easy to use in any application which understands SQL
- Based on the extensibility framework within the Oracle kernel
- Free with all versions of Oracle, from Xe to Enterprise Edition
- Embedded in many Oracle applications
- Multilingual, and capable of managing many types of document
## Handling Text Anywhere

Any type of text, in any place

<table>
<thead>
<tr>
<th>In Database</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>varchar2</td>
<td>File System</td>
</tr>
<tr>
<td>LOB</td>
<td>URL</td>
</tr>
<tr>
<td>Plain Text</td>
<td>Word Doc</td>
</tr>
<tr>
<td>HTML</td>
<td>PDF</td>
</tr>
<tr>
<td>XML</td>
<td>CJK</td>
</tr>
<tr>
<td>English</td>
<td>European, Cyrillic</td>
</tr>
<tr>
<td></td>
<td>150 others</td>
</tr>
</tbody>
</table>

Plain Text, Word Doc, HTML, XML, PDF, CJK, others
Features Overview

- All classical full-text search features...
  - Exact word matching
  - Boolean word search: and, or, not
  - Phrases, word proximity and “within field” searches
  - Inexact search:
    - Wild-cards, “fuzzy” / soundex, name search
    - Stemming in multiple languages with auto detection
    - ISO Thesaurus
Features Overview

continued

- **Plus Advanced Capabilities...**
  - Name Search
  - Theme identification, indexing, and searching using million-word “knowledge base” and linguistic rules
  - Entity Extraction: find people, names, places, dates etc
  - Advanced XML search
  - Text Analytics: classification and clustering
## Usage Scenarios

<table>
<thead>
<tr>
<th>eDiscovery</th>
<th>Search Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Mining Applications</td>
<td>Intranet Search</td>
</tr>
<tr>
<td>Litigation Support</td>
<td>Intranet and Extranet</td>
</tr>
<tr>
<td>Forensic Investigation</td>
<td>Application Search</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content Management</th>
<th>Text Warehousing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed document types</td>
<td>Very high volume of data</td>
</tr>
<tr>
<td>Documents + metadata</td>
<td>Essentially read-only</td>
</tr>
<tr>
<td>Workflow and checkin/checkout</td>
<td>Often partitioned by customer</td>
</tr>
</tbody>
</table>

**Text-enabled transactional systems**
- Adding free-text to complex SQL
Creating an Oracle Text Index

CREATE INDEX prod_name_idx ON product_information(product_name) INDEXTYPE IS ctxsys.context;

SELECT score(99), product_id, product_name
FROM product_information
WHERE contains (product_name, 'monitor NEAR full hd', 99)>0
ORDER BY score(99) DESC;

<table>
<thead>
<tr>
<th>SCORE(99)</th>
<th>PRODUCT_ID</th>
<th>PRODUCT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>3331</td>
<td>Full HD Monitor 22 inch</td>
</tr>
<tr>
<td>56</td>
<td>3060</td>
<td>Monitor and TV combo, full HD</td>
</tr>
</tbody>
</table>
Index Types in Oracle Text

CONTEXT is primary indextype but other are available

- CONTEXT
- CTXCAT
- CTXRULE
- XQuery Full Text
Product Architecture

- Index data from tables, either directly or indirectly
  - Directly: varchar2, CLOB, BLOB
  - Indirectly: URL or filename stored in column

- All indexes use the EXTENSIBILITY FRAMEWORK which allows for “Domain Indexes”

- Oracle Text indexes reside in Oracle Database tables

- Features such as RAC, partitioning, parallel query are all “text aware”
Oracle Text Index Tables

Each index has a set of “DR$” tables

```
SQL> create table mytab (text varchar2(2000));
Table created.
SQL> create index myindex on mytab (text) indextype is ctxsys.context;
Index created.
SQL> select table_name from user_tables;
TABLE_NAME
MYTAB
DR$MYINDEX$I
DR$MYINDEX$R
DR$MYINDEX$K
DR$MYINDEX$N
```
A Conceptual View of the Text Index

<table>
<thead>
<tr>
<th>rowid</th>
<th>Text Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Night and day, day and night.</td>
</tr>
<tr>
<td>r2</td>
<td>It was a wild and stormy night.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>token_text</th>
<th>token_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>night</td>
<td>1 &lt;1,6&gt; 2 &lt;7&gt;</td>
</tr>
<tr>
<td>day</td>
<td>1 &lt;3,4&gt;</td>
</tr>
<tr>
<td>wild</td>
<td>2 &lt;4&gt;</td>
</tr>
<tr>
<td>stormy</td>
<td>2 &lt;6&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rowid</th>
<th>docid</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>1</td>
</tr>
<tr>
<td>r2</td>
<td>2</td>
</tr>
</tbody>
</table>

*Stopwords:* it, was, a
The Indexing Pipeline

- All pipeline stages are configurable by a system of “preferences” and “attributes”
- Most can be replaced by user-written plugin modules in PL/SQL, C or Java
Index Maintenance: Sync and Optimize

- Oracle Text indexes are *asynchronous* by default
- You must arrange for your index to be synchronized
  - `ctx_ddl.sync_index`, or “sync every ...” in create index command
- Trade-off between availability of changes and optimality of index
- Optimize index to remove garbage and compact lists
  - `ctx_ddl.optimize_index`
- Or use two-level “near real time” index
  - New feature, see later
Oracle Text Queries

- Can use
  - standard SQL “SELECT” query syntax or
  - XML-based Result Set Interface

- Return a “score” to indicate the relevance of each hit

- Use “sections” to mark structured data in documents or in other columns of the table

- Can mix structured (numeric, date) with unstructured (full text) searches in a single query expression
Some Useful Operators

- STEM ($) - matches words with the same linguistic base form
- FUZZY (...) - finds mis-spellings
- NEAR (...) - proximity search for words close to each other
- WITHIN section – simple section search
- SDATA (...) - performs structured search within text index
- NDATA (...) - match names (or other similar inexact data)
- MVDATA(...) – multi-valued section data
- NT, BT, SYN – thesaurus operators
Beyond the text index

Document level services

▪ Search-related
  – Highlighting, markup, snippets

▪ Theme and gist extraction
  – What a doc is “about”, and a summary based on that theme

▪ Entity extraction
  – Find people, places, dates, times, zip codes, etc
  – Customize with own user dictionary and user rules
Text Analytics

Machine learning algorithms for document classification

- Document classification
  - Supervised classification of documents using training set
    - Allows for routing of documents to classification sets
    - Uses K-Means or State Vector Machine algorithms
  - Unsupervised classification
    - Document clustering – groups documents according to “nearness” in n-dimensional “feature space”.
Exadata

- Oracle Text works great on Exadata
- Partition index for cross-node parallelism
- Recent POC on X3: Index 700MB per second
  = 2.5 TB per hour
Program Agenda

- Introduction to Oracle Text
- Oracle Text 12c
- Some Marquee Customers
- Roadmap
• Architecture Improvements
Result Set Interface

- XML-based query interface
- Bypasses SQL layer for performance and result set flexibility
- Result set descriptor (a block of XML) defines required results
- Results returned in one block of XML
  - Access directly in application or process via XML DB functions
- Effectively allows multiple queries in one call to database
  - What is the total number of hits for my query
  - Give me a summary of the first 20 hits in score order (the "hitlist")
  - Which authors are represented in the entire result set, and how many results for each author?
Index architecture changes

- **BIG_IO**
  - Previously index was optimized for BLOB – 4000 byte chunks
  - Now uses secure files with much bigger data chunks
  - Huge reduction in random I/O (disk seeks) for large index entries

- **Separate Offsets**
  - Separates DOCID from token offsets in $I$ table
  - Greatly improved performance when offset information not needed
    - Improved: single word search, multi-word AND / OR searches, field section searches
    - Not improved: phrase, proximity, zone section searches
Near Real Time Indexes

New “STAGE_ITAB” feature

- Keeps recent updates to index in small table
  - Easy to cache in SGA so fragmentation not an issue
- Automatically optimizes in-memory data to main index
- Allows for very-frequent syncs without slowing queries
- Use “STAGE_ITAB” storage preference
- New
  - postings table $G$ (equivalent to $I$)
  - and postings index $H$ (equivalent to $X$)
Forward Index

Much faster document services

- Problem: Snippets and highlighting are slow since full document must be processed to find hits
  - Requires datastore, filter, lexer processing
- Solution: Save document during indexing
  - Tokenized
  - Compressed
  - Indexed by word
- Store offsets only, or plain text only, or full HTML
  - "save copy" option
XQuery Full Text

- Full-text search within XML database framework
- “Universal index” allows full mix of structured and unstructured data
- Use PATH section type with XML_ENABLE = ‘true’

```
SELECT po.id
FROM PURCHASEORDER po
WHERE XMLExists
  ('$src/purchaseOrder/billingInstruction/Address
   [. contains text {$PHRASE1}
   ftand {$PHRASE2} using stemming]
PASSING po.x, 'Science' as “PHRASE1”, ‘Magdalen’ as “PHRASE2”)
```
Languages and Lexer updates

- New CTX_DOC procedure:
  - policy_languages
    - Provides list of most likely languages for a document

- Document Level Lexer
  - Apply different lexer to sets of documents
  - Similar to multi_lexer, but not just for languages

- Bigram mode for Japanese Vgram Lexer
  - Eliminates need for expensive wildcard searches
Miscellaneous Improvements - cont

- Query Filtercache
  - Queries often filtered by identical expressions
    - “and PUBLICFLAG or (A or B or C) WITHIN securitygroup”
    - “and SDATA(instock, T)”
  - Only need to evaluate on first use

- Pattern Stopclass
  - Remove words based on regular expression
  - Great for clearing “garbage” tokens
  - Eg. Remove words longer than certain length, or words with numbers, or words beginning with a certain string
Document Lexer
Choose lexer options for different document types

- Based on MULTI_LEXER but untied from query language
- LANGUAGE column may now contain any value
- Internally uses special section DR$ML to identify lexer used for each row
- Query will be lexed by each possible lexer
- Where different query terms are found they will be applied only to the relevant rows using DR$ML section
Query Filter Cache
Cache frequent sub-queries in Text layer

- It’s common to have the same subquery in many searches
- Eg a security or location filter
- Subquery must be fetched from SGA and evaluated every time
- New ctxFilterCache operator declares frequently-used part
- Will be evaluated in full first time, fetched from cache subsequently
- Cache size tunable, subqueries aged out of cache will be reevaluated
- Options to save score, and just store “top N by score” rather than all results
SDATA Improvements
More flexible Structured Data within Text Index

- **Updatable SDATA Sections**
  - Synchronous updates to SDATA without reindexing
  - Ideal for rapidly changing info like stock levels

- **Add SDATA sections to existing index**
  - No need to recreate the index to add new structured data

- **Number of SDATA sections increased from 32 to 99**
  - Maximum FILTER BY / ORDER BY columns still 32
SDATA Improvements - Cont

Sorting

- Query template can now include sorting by SDATA

```xml
<query>
  <textquery> digital and sdata(stocklevel > 4) </textquery>
  <order>
    <orderkey> SDATA(stocklevel) desc </orderkey>
    <orderkey> score desc </orderkey>
  </order>
</query>
```

- Only affects order of rows returned from text query
  - Over-ridden by explicit ORDER BY in query
  - Ignored in functional invocation
Miscellaneous Improvements

- NEAR enhancements
  - Many more expression types may be used in a NEAR
  - NEAR may be nested inside NEAR
- MNOT (mild not) operator
  - York MNOT (New York)
    - Finds all documents where “York” is not part of the phrase “New York”
      “York is in England. New York is in the USA” => matches search
- Session duration Stored Query Expressions
  - Allow temporary SQEs without clogging system
- Removed limit on number of MDATA and FIELD sections
• Document Service Enhancements
Forward Index and Save Copy

Faster document services

- Document Services: ctxdoc.
  - Filter, Highlight, Markup, Snippet

- Must fetch original document
  - May not be available
  - Expensive if document must be fetched, filtered, lexed...

- Forward Index maps word offsets (from index) to character offsets
  - For markup, highlight, snippet

- Save Copy stored compact representation of document
  - With HTML for filter, markup in non-plaintext mode
  - Without HTML for snippet, and filter, markup in plaintext mode
Snippet Support in Result Set Interface

- Result Set Interface – XML query support introduced in 11g
- Fetch rowid, sdata values and now snippets

```xml
<ctx_result_set_descriptor>
  <hitlist start_hit_num="1" end_hit_num="10" order="SCORE DESC">
    <rowid />
    <score />
    <sdata name="title" />
    <snippet radius="20" max_length="160" starttag="&lt;b&gt;" endtag="&lt;/b&gt;" />
  </hitlist>
  <count />
</ctx_result_set_descriptor>
```
Language Identification

- New ctx_doc procedure POLICY_LANGUAGES
  - Lists document language (in order of likelihood)
  - Based on characters and dictionary
  - Uses same technology as AUTO_LEXER
  - Works best with at least a paragraph of text
• Other Enhancements
Pattern Stopclass
Avoid junk in index

- Define stopwords by regular expression
- Usage examples
  - no numbers >= 5 digits
    ```python
ctx_ddl.add_stopclass ( 'stop', 'fivedignums', '[:digit:],\.[{5,}]' )
```
  - avoid indexing tokens >= 20 characters
    ```python
ctx_ddl.add_stopclass ( 'stop', 'longwords', '[:alnum:]{20,}' )
```
  - avoid alphanumerics with >= 5 digits anywhere
    ```python
ctx_ddl.add_stopclass ( 'stop', 'wordswith5digitssomewhere', '[:alpha:]*[:digit:]){5,}[:alpha:]*' )
```
- Use for simple redaction – eg. don’t index SSNs

Avoid junk in index
Unlimited MDATA sections

- Restriction on number of MDATA sections lifted
- Effectively unlimited
XQuery Full Text

- XML DB now supports complete XML Full Text syntax
- “Universal Index”
  - Implemented via Oracle Text index with XML_ENABLE=true
- Constructs not supported by text index implemented by post-filter