Oracle8™ ConText® Cartridge

QuickStart

Release 2.4

July 1998

Part No. A63819-01

ORACLE®

Enabling the Information Age™

Oracle8 ConText Cartridge QuickStart, Release 2.4

Part No. A63819-01

Release 2.4

Author: D. Yitzik Brenman

Contents

3 Using Theme Queries and CTX_LING

4 Implementing ConText

Send Us Your Comments

Oracle8 ConText Cartridge QuickStart, Release 2.4

Part No. A63819-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- FAX - (650) 506-7200. Attn: Oracle8 ConText Cartridge Documentation
- Postal service:
  Oracle Corporation
  Oracle8 ConText Cartridge Documentation
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle World Wide Support Center.

1

Introduction

This chapter introduces Oracle8 ConText Cartridge and the QuickStart process, and describes the requirements for enabling ConText to store and retrieve text.

The following topics are covered in this chapter:

- What is Oracle8 ConText Cartridge?
- What is QuickStart?
- QuickStart Requirements

What is Oracle8 ConText Cartridge?

Oracle8 ConText Cartridge provides powerful search, retrieval, and viewing capabilities for text stored in an Oracle8 database. In addition, ConText provides advanced linguistic processing of English-language text.

These capabilities are implemented in Oracle8 using ConText server processes and ConText indexes for each database column in which text is stored.

With ConText, you can use SQL and PL/SQL to incorporate advanced text and linguistic functionality into new or existing applications.

Text Processing

ConText provides advanced text searching and viewing functionality, such as full text retrieval (queries), relevance ranking, and query term highlighting.

Text queries support a wide range of search options, including: logical operators (AND, OR, NOT, etc.), proximity searches, thesaural expansion, section searches, and stored queries.

Text viewing capabilities include WYSIWIG and plain text viewing of documents, as well as highlighting of query terms.

Note: Text viewing is not covered in this manual. For complete details about text viewing, see *Oracle8 ConText Cartridge Application Developer's Guide*.

For an explanation of the steps required to set up text queries, see "Startup and Hot Upgrade" in Chapter 2, "Using Text Queries".

For examples of some basic text queries, see "Text Queries" in Chapter 2, "Using Text Queries".

Linguistic Processing

For English-language text, ConText provides in-depth linguistic analysis and output. The linguistic output can be used to create theme indexes for performing theme queries and theme highlighting.

A theme query retrieves documents based on the main topics found in the documents in a column. Theme highlighting highlights the paragraphs that contribute to a theme for a selected document.

Note: Theme queries are performed in the same way as text queries and support many of the same search options.

Theme highlighting is similar to text highlighting and is not covered in this manual. For complete details about theme highlighting, see *Oracle8 ConText Cartridge Application Developer's Guide*.

In addition, ConText can be used to generate linguistic output for individual documents. The output can be used to provide advanced document viewing capabilities such as lists of themes, theme summaries, and document Gists.

For an explanation of the steps required to set up theme queries and generate linguistic output, see "Startup and Hot Upgrade" in Chapter 3, "Using Theme Queries and CTX_LING"

For examples of some basic theme queries and linguistic output usage, see "Theme Queries" and "Linguistics" in Chapter 3, "Using Theme Queries and CTX_LING".

What is QuickStart?

Before text and theme query functionality can be integrated into an application, the database columns used to store text must be defined as text columns and text/theme indexes must be created for the columns.

In addition, if you want to incorporate theme lists and Gists into your English-language text applications, you must use the procedures in the CTX_LING package to generate linguistic output.

QuickStart provides step-by-step instructions for quickly and easily performing these tasks. Once the QuickStart tasks have been performed, your system is text-enabled for text/theme queries and using linguistic output.

QuickStart also provides examples of text and theme queries, as well as examples of queries that use the output generated by the ConText Linguistics.

QuickStart Requirements

Before you can perform the QuickStart tasks, Oracle8 and ConText must be installed and the ConText implementation tasks must be completed. In addition, QuickStart is easiest to perform when text is already stored in the database.

If the required installation and implementation tasks have not been completed, see Chapter 4, "Implementing ConText".

Oracle Users and Tables

This QuickStart manual presents an example Oracle user named *ctxdev* and an example text table named *docs*. *Ctxdev* and *docs* are used in all QuickStart examples; however, they are used for illustration purposes only and are not provided by ConText.

To perform QuickStart, you can use an existing Oracle user and table or you can create a user and table (i.e. *ctxdev* and *docs*).

ConText Defaults

QuickStart assumes the following conditions for your text and uses the appropriate defaults in ConText:

- non-pictorial language (i.e. English or the other supported Western European languages)
- unformatted, plain (ASCII) text
- text stored directly in database columns
- each document stored as a single row

If these conditions are not valid for your setup, QuickStart can still be performed; however, you cannot use the defaults in ConText and must specify settings that match your setup.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

2

Using Text Queries

This chapter provides a quick description of the setup tasks that must be performed to enable text queries with ConText. It also provides examples of the three methods for performing queries.

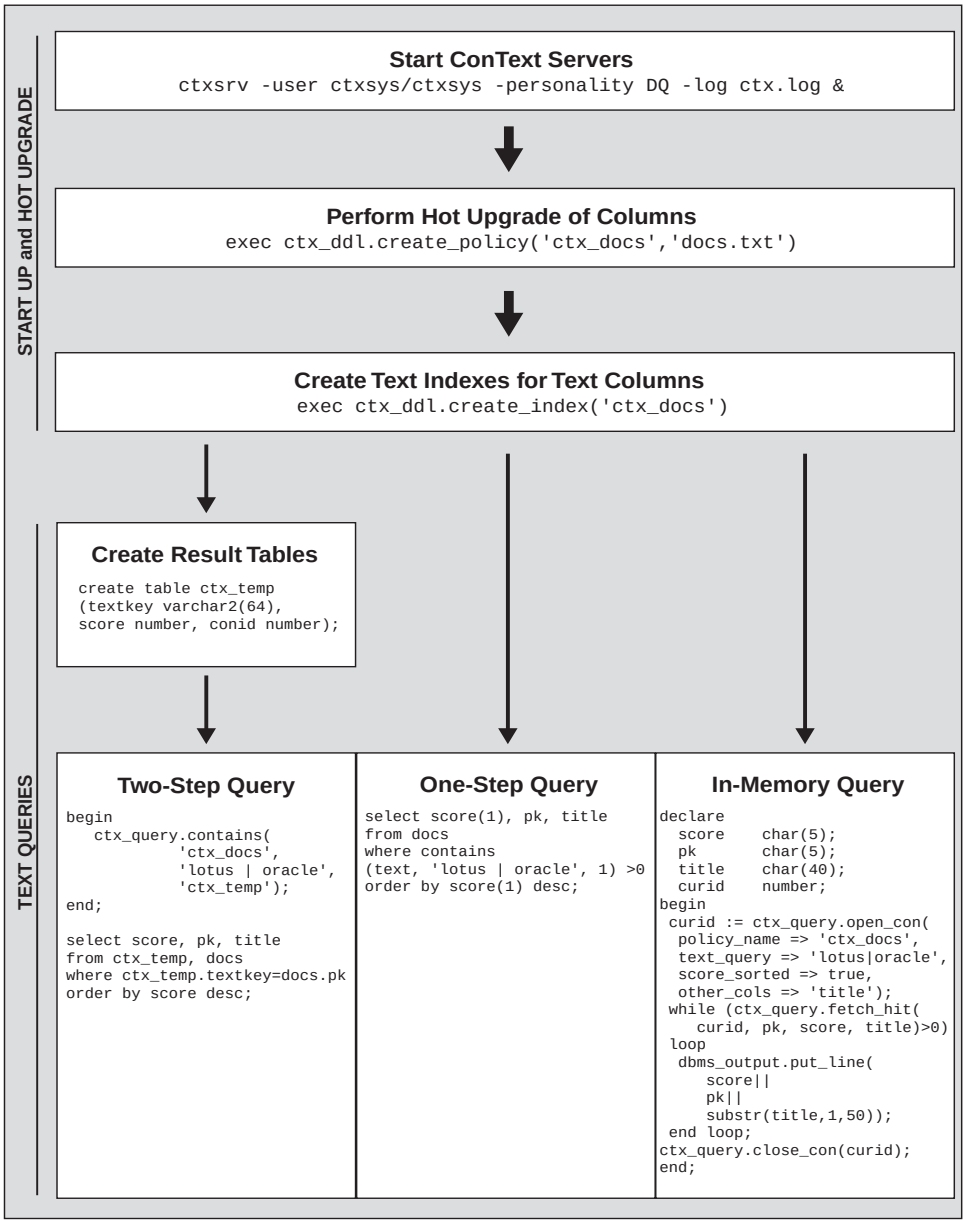
The following topics are covered in this chapter:

- Text Query Task Map
- Startup and Hot Upgrade
- Text Queries

Note: Before you can perform the QuickStart tasks described in this chapter, ConText must be installed and certain implementation tasks must be completed.

If the required installation and installation tasks have not been completed, see Chapter 4, "Implementing ConText".

Text Query Task Map



QuickStart Tasks

Perform the following tasks to set up a text column in a table, index the column, and perform text queries on the column:

- Startup and Hot Upgrade
  - Start ConText Servers
  - Perform Hot Upgrade of Columns
- Text Queries
  - Create Text Indexes for Text Columns
  - Create Result Tables (Two-Step Queries Only)
  - Two-Step Query Example
  - One-Step Query Example
  - In-Memory Query Example

Startup and Hot Upgrade

The first two setup tasks for text queries are:

- Start ConText Servers
- Perform Hot Upgrade of Columns

Note: These tasks can be performed in any order.

Start ConText Servers

To create text indexes for a column, one or more ConText servers must be running with the DDL (D) personality. In addition, to perform text queries, one or more ConText servers must be running with the Query (Q) personality.

Note: ConText servers can only be started by the CTXSYS Oracle user provided by ConText.

You can start a ConText server by calling the ctxsrv executable from the command-line. You can also use the ctxctl command-line utility to start ConText servers.

For example, to start a ConText server with the required personalities from the command-line of your server machine, execute the following command:

```
$ ctxsrv -user ctxsys/ctxsys -personality DQ -log ctx.log &
```

Note: This example is specific to UNIX-based operating systems. The executable names and command-line syntax for ConText servers may be different in Windows NT and other operating systems.

For information about the executable names / syntax for ConText servers in your operating system, see the Oracle8 installation documentation for your operating system.

In this example, the password for the CTXSYS user is 'ctxsys'. The server is started as a background process on the server machine and all ConText information for the session is written to a file named *ctx.log*.

Note: This example results in the password for the CTXSYS user being visible to all users on the server machine.

If, for security reasons, you require the CTXSYS password to be masked out, you can call the ctxsrv executable without specifying the *-user* argument. The system then prompts you to enter this user information as *username*/*password*.

An additional personality, DML (M), can be assigned to ConText servers. ConText servers with the DML personality automatically update the ConText index for a column when changes which affect the index are made to rows in the table for the column.

Because the DML personality is not required for QuickStart, it is not discussed in this manual.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Perform Hot Upgrade of Columns

Hot upgrade is the process of defining database columns as text columns. A text column is any table or view column for which a policy has been created.

A policy identifies the column used to store text, the text storage method, and the options that ConText uses to create a ConText index for the column. ConText supports creating column policies for text indexing and theme indexing.

To create a text indexing policy for a column, call the CTX_DDL.CREATE_POLICY procedure and specify the following parameters:

- policy name
- fully qualified column name for the policy
- indexing options

Each indexing option, also known as a **preference**, answers one of the questions that ConText requires to create an index:

Indexing Option	Question Answered	Default Values
Data Store	How is the text stored in the column?	Text stored directly in the column, each row in the table represents a separate, complete document
Filter	What format is the text in?	Plain (ASCII) text
Lexer	What language is the text in and how are tokens identified in the text?	English or other Western European languages; tokens identified by blank space(s); standard punctuation and non-alphanumeric characters; case-sensitivity disabled
Engine	How is the ConText index created and where is it stored	ConText index created using approximately 12.5 Mb of memory; index stored in user's tablespace
Stoplist	Which tokens are not included as entries in the index?	Supplied list of English stopwords
Wordlist	Which special query options are enabled for the index	Soundex query expansion disabled; fuzzy matching and stemming set to English

The following example creates a text indexing policy named *ctx_docs* for the *text* column in the *docs* table owned by *ctxdev*. Because this example uses all the default ConText indexing options, no preferences have to be explicitly set for *ctx_docs*:

```
exec ctx_ddl.create_policy('ctx_docs','docs.text')
```

If you want to specify different indexing options for a policy, you can specify corresponding preferences when you call CREATE_POLICY.

For example, if you want to enable case-sensitive lexing, which, in turn, enables case-sensitive text queries, you can create a Lexer preference with case-sensitive lexing enabled, then use the preference when creating a policy with CREATE_POLICY.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Text Queries

A text query searches the text column(s) in the queried table(s) for specified terms (words and phrases) and returns all rows (i.e. documents) which contain occurrences of the terms.

In addition, a score is returned for each selected document. The score is based on the number of occurrences of the query terms in the document and represents the relevance of the document to the query.

ConText supports a wide range of boolean and expansion operators which can be applied to the terms in a text query to produce different results. In addition, a text query can include searches for structured data.

Before you can perform a text query, you must perform the following tasks:

- Create Text Indexes for Text Columns
- Create Result Tables (Two-Step Queries Only)

You can then perform text queries using any of the supported query methods.

For examples of the query methods, see "Two-Step Query Example", "One-Step Query Example", or "In-Memory Query Example" in this chapter.

Note: All of the text query examples are case-insensitive because case-sensitive lexing is disabled for the *ctx_docs* policy.

Create Text Indexes for Text Columns

To create a text index for a column, call the CREATE_INDEX stored procedure in the CTX_DDL PL/SQL package and specify the text indexing policy for the column.

For example:

```
exec ctx_ddl.create_index('ctx_docs')
```

In this example, CREATE_INDEX is called in SQL*Plus to create a text index for the text column (*ctxdev.docs.text*) in the *ctx_docs* policy.

After a text index is created for a column, ConText servers with the Query personality can process text queries for the column.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Create Result Tables (Two-Step Queries Only)

If you want to perform two-step queries, you must create a result table which stores a list of the primary keys (textkeys) and scores for the documents that satisfy the search criteria you specify in the first step of the two-step query.

The result table can have any name; however, it *must* have the structure (column names and datatypes) specified in the following example:

```
create table ctx_temp (textkey varchar2(64), score number, conid number);
```

In this example, a result table named *ctx_temp* is created in SQL*Plus. The *textkey* column stores the primary key for the documents and the *score* column stores the scores generated by the query.

The third column, *conid*, stores a number which identifies the results for each query. The *conid* column is used only when the result table is used to store the results for multiple queries from either a single user or multiple users.

> **See Also:** *Oracle8 SQL Reference*, *Oracle8 ConText Cartridge Application Developer's Guide*

Two-Step Query Example

In the first step of a two-step query, you call the CONTAINS stored procedure in the CTX_QUERY PL/SQL package to populate an existing result table.

In the second step, you query the result table to return a hitlist of the documents.

> **Note:** Because the result table does not store document details or the text of the document, if you want to create a hitlist that includes document details and/or the text of a document, you must perform a query that joins the original text table and the results table.

The following example illustrates a basic two-step query:

1.
```
begin
  ctx_query.contains('ctx_docs','lotus|oracle','ctx_temp');
end;
```

2.
```
select score,title from ctx_temp, docs
where ctx_temp.textkey=docs.pk
order by score desc;
```

In this example, a search is performed on the text column (*ctxdev.docs.text*) in the *ctx_docs* policy to find all documents in which the term *oracle* or *lotus* occurs. The results of the search are stored in the *ctx_temp* results table.

Then, the *ctx_temp* and *docs* tables are joined in a query to create a hitlist which lists *score* and *title* for each document returned in step one.

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

One-Step Query Example

One-step queries use the ConText SQL function, CONTAINS, which is called directly in the WHERE clause of a SELECT statement.

In a one-step query, the CONTAINS stored procedure and result tables required for two-step queries, are *not* used.

Note: Because SELECT statements operate on column and table names, the name of the text column is used in a one-step query, rather than the policy for the column.

The following example illustrates a one-step query that returns the same results as in "Two-Step Query Example":

```
select score(1), pk, title from docs
where contains(text, 'lotus | oracle', 1) > 0
order by score(1) desc;
```

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

In-Memory Query Example

In-memory queries can be performed using OPEN_CON, FETCH_HITS, and CLOSE_CON in the CTX_QUERY PL/SQL package.

OPEN_CON opens a CONTAINS cursor to a query buffer and executes a query. The results of the query are stored in the query buffer. FETCH_HIT retrieves the results, one hit at a time, and CLOSE_CON releases the CONTAINS cursor.

Note: In-memory queries are generally faster than one-step and two-step queries for queries that return large hitlists. In addition, in-memory queries do not require the allocation of database tables for the results.

The following example illustrates an in-memory query that returns the same results as in "Two-Step Query Example":

```
declare
   score  char(5);
   pk     char(5);
   title  char(40);
   curid  number;
begin
   curid := ctx_query.open_con(policy_name => 'ctx_docs',
                               text_query => 'lotus|oracle',
                               score_sorted => true,
                               other_cols => 'title');
   while (ctx_query.fetch_hit(curid, pk, score, title)>0)
      loop
        dbms_output.put_line(score||pk||substr(title,1,50));
      end loop;
   ctx_query.close_con(curid);
end;
```

In this example, *score*, *pk*, *title*, and *curid* are declared as variables.

The *score_sorted* argument for OPEN_CON specifies that the results of the query are stored in the buffer in descending order by score. The *other_cols* argument species that the *title* column from the queried table is returned along with the *score* and *pk* columns in the query results.

FETCH_HITS retrieves *score*, *pk*, and *title* for each hit until the buffer is empty.

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

3

Using Theme Queries and CTX_LING

This chapter provides a quick description of the tasks that must be performed to enable theme queries for ConText, as well as to generate linguistic output for use in an application. It also provides examples of theme queries and queries using linguistic output.

Note: Theme queries and linguistic output are only available for English-language text.

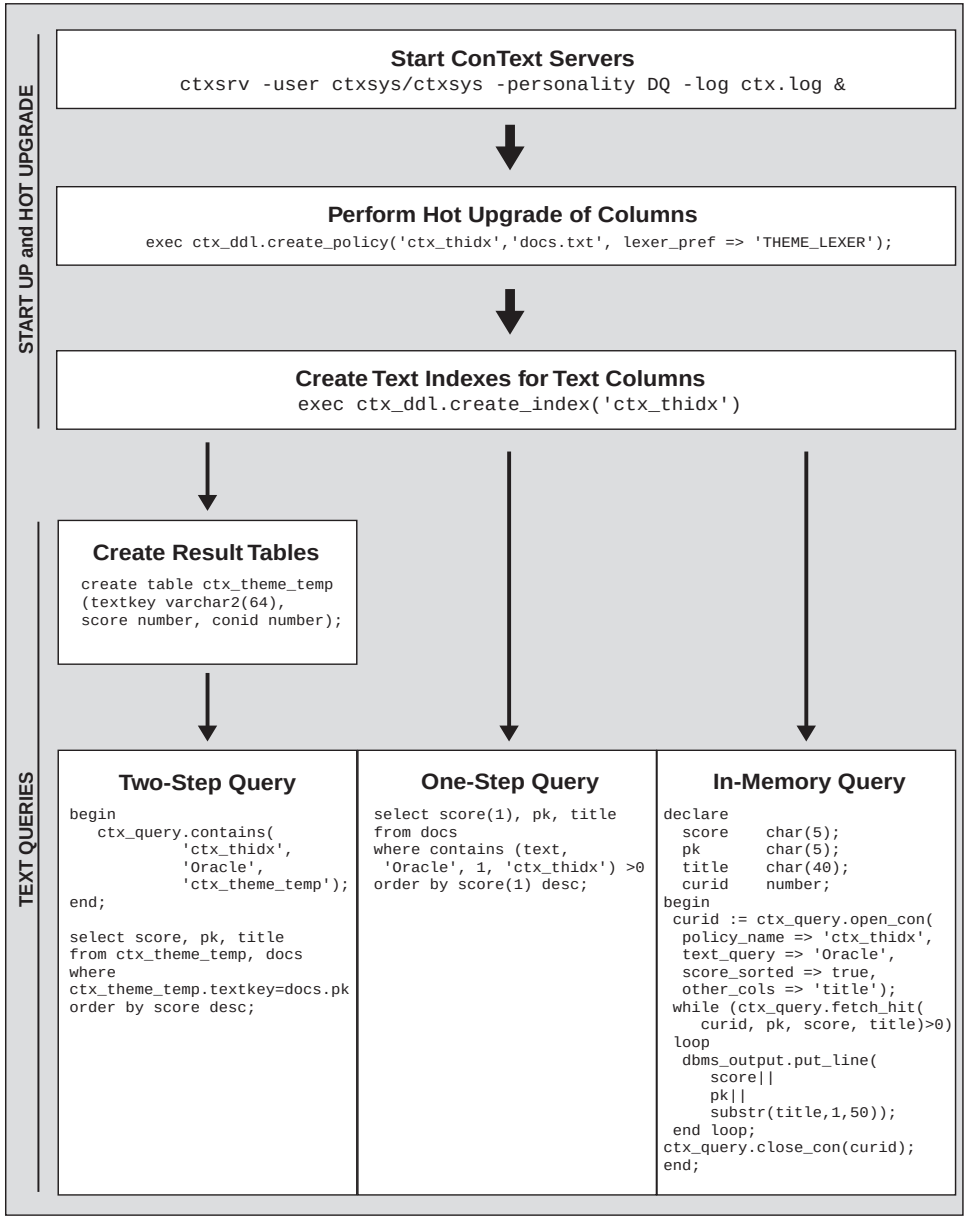
The following topics are covered in this chapter:

- Theme Query Task Map
- Linguistics Task Map
- Startup and Hot Upgrade
- Theme Queries
- Linguistics

Note: Before you can perform the QuickStart tasks described in this chapter, ConText must be installed and certain implementation tasks must be completed.

If the required installation and installation tasks have not been completed, see Chapter 4, "Implementing ConText".

Theme Query Task Map

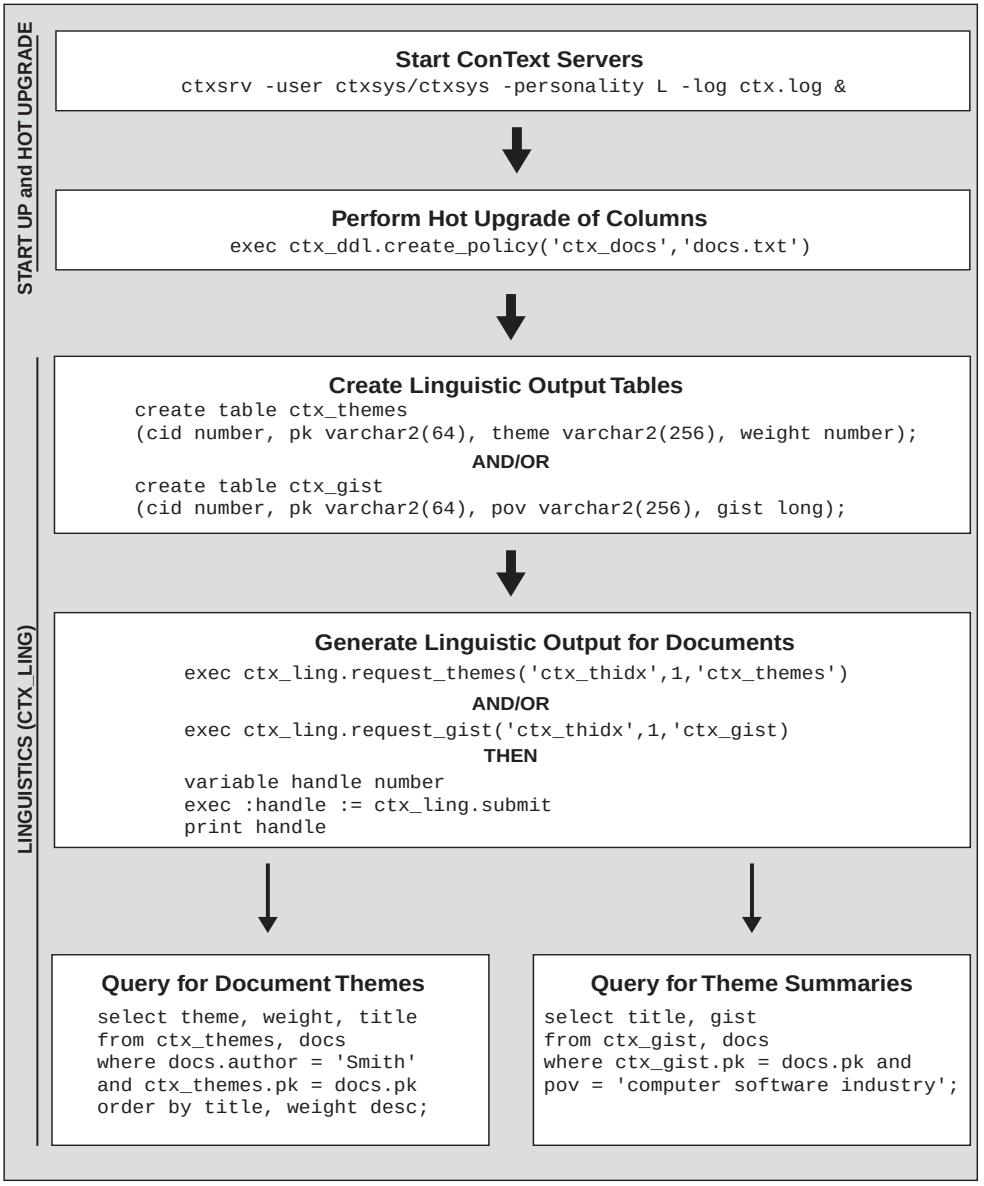


QuickStart Tasks for Theme Queries

Perform the following tasks to set up a text column in a table, index the column, and perform theme queries on the column:

- Startup and Hot Upgrade
  - Start ConText Servers
  - Perform Hot Upgrade of Columns
- Theme Queries
  - Create Theme Indexes for Text Columns
  - Create Result Tables (Two-Step Queries Only)
  - Theme Query Examples

Linguistics Task Map



QuickStart Tasks for Linguistics

Perform the following tasks to generate linguistic output for the documents in the column and to query the generated output:

- Startup and Hot Upgrade
  - Start ConText Servers
  - Perform Hot Upgrade of Columns
- Linguistics
  - Create Linguistic Output Tables
  - Generate Linguistic Output for Documents
  - Example Queries for Linguistic Output

Startup and Hot Upgrade

The first two setup tasks for theme queries and the Linguistics are:

- Start ConText Servers
- Perform Hot Upgrade of Columns

Note: These tasks can be performed in any order.

Start ConText Servers

Similar to text indexing and queries, theme indexing is performed by ConText servers with the DDL (D) personality and theme queries are processed by ConText servers with the Query (Q) personality.

However, to enable a ConText server to generate linguistic output through the Linguistics, the Linguistic (L) personality must be specified for the server.

The following command starts a ConText server with the appropriate personalities for creating a theme index and performing theme queries:

```
$ ctxsrv -user ctxsys/ctxsys -personality DQ -log ctx.log &
```

The following command starts a ConText server with the appropriate personality for generating linguistic output:

```
$ ctxsrv -user ctxsys/ctxsys -personality L -log ctx.log &
```

Suggestion: Once linguistic output has been generated for all your documents, if the documents are not going to change, the Linguistic personality is not needed and the server can be shut down or reassigned for other text operations.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Perform Hot Upgrade of Columns

The procedure for upgrading a column for theme indexing is the same as the procedure for text indexing, except that the lexer used in the policy for theme indexing is not the same as the lexer used in a text indexing policy.

To create a theme indexing policy, the Theme Lexer is specified for the policy, using the predefined Lexer preference, THEME_LEXER.

The following example illustrates using a PL/SQL block to create a column policy named *ctx_thidx* for theme indexing:

```
begin
ctx_ddl.create_policy('ctx_thidx',
                      'docs.text'
                      lexer_pref => 'THEME_LEXER');
end;
```

The only differences between this theme indexing policy and the text indexing policy created in"Perform Hot Upgrade of Columns" in Chapter 2 are the policy names and the specification of THEME_LEXER as the Lexer preference for the theme indexing policy.

Note: This example and the text indexing policy example together result in the *ctxdev.docs.text* column having two policies and, subsequently, two indexes: one for text queries and one for theme queries.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Theme Queries

Theme queries search the text column(s) of the queried table(s) for specified themes and returns all rows (i.e. documents) that have the specified themes. A theme represents a major topic or developed subject in a document.

Before you can perform theme queries, you must perform the following tasks:

- Create Theme Indexes for Text Columns
- Create Result Tables (Two-Step Queries Only)

Once these tasks have been performed, you can perform theme queries for your documents. Some of the issues related to theme queries are discussed in "Theme Query Examples" in this chapter.

See Also: *Oracle8 ConText Cartridge Administrator's Guide*

Create Theme Indexes for Text Columns

To create a theme index for a column, call the CTX_DDL.CREATE_INDEX stored procedure and specify the theme indexing policy for the column.

The following example creates a theme index for the text column (*ctxdev.docs.text*) in the *ctx_thidx* policy:

```
exec ctx_ddl.create_index('ctx_thidx')
```

After a theme index is created for a column, ConText servers with the Query personality can process theme queries for the column.

Create Result Tables (Two-Step Queries Only)

The structure of the result table for a theme query and the method for creating the table is identical to the result table used in a text query. In fact, you can use the same result table or you can create a separate result table for two-step theme queries.

Note: The two-step theme query example in "Theme Query Task Map" use a different result table than the result table used in the example for two-step text queries.

Theme Query Examples

The methods for performing theme queries are identical to the three text query methods presented in "Text Queries" in Chapter 2, with the exception that theme queries are always case-sensitive and the scoring methods are different.

"Theme Query Task Map" illustrates how to perform theme queries using all three of the supported query methods. In the examples provided, the theme *Oracle* is queried.

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

Case-Sensitivity

Unlike text queries, which can be case-sensitive or case-insensitive, theme queries are always case-sensitive. As a result, theme queries for places and names generally return different results than text queries for the same words / phrases.

For example, a theme query for the term *Oracle* will produce those documents in which ConText determined *Oracle Corporation* (the canonical form of *Oracle*) to be a major theme in the document. In contrast, a text query (in a case-insensitive index) for the term *Oracle* will return all documents that contain occurrences of either *Oracle* or *oracle*, regardless of how the term appears in the document.

Scoring

Similar to text queries, the documents returned by theme queries have a score. While scores in a theme query indicate the relevance of the selected documents to the query, each score is based on the weight of the queried theme in the document, rather than the number of occurrences of the theme.

Theme weights are generated during theme indexing. Theme weight measures the importance of a document theme relative to the other themes in the document.

Columns with Multiple Indexes

If a column has two or more indexes, which may be common with text indexes and theme indexes, you must specify the name of the policy for the appropriate index in the *pol_hint* argument for the CONTAINS function of a one-step query.

For example, if the QuickStart tasks for both text and theme queries have been performed, the column *ctxdev.docs.text* has both a text indexing policy (*ctx_docs*) and a theme indexing policy (*ctx_thidx*) and indexes for both.

The one-step theme query example in "Theme Query Task Map" identifies *ctx_thidx* as the index to be searched.

Linguistics

The setup tasks required for using the Linguistics to generate output for a document are:

- Create Linguistic Output Tables
- Generate Linguistic Output for Documents

Once these tasks have been performed, you can query the output tables for themes and Gists. "Example Queries for Linguistic Output" in this chapter provides examples of the types of queries you can perform.

Create Linguistic Output Tables

The Linguistics generate three types of output for a document:

- list of up to 50 document themes
- Gist (paragraphs or sentences that best represent all the document themes)
- theme summaries (paragraphs or sentences that best represent each document theme)

The output is stored in tables specified by the user. The linguistic output tables can have any name; however, they *must* have the following structure:

```
create table ctx_themes (
cid number, pk varchar2(64), theme varchar2(2000), weight number);

create table ctx_gist (cid number, pk varchar2(64),pov varchar2(80), gist long);
```

In these examples, two tables (*ctx_themes* and *ctx_gist*) are created for storing linguistic output. The *pk* column in each table stores the primary keys (textkeys) for each document. The *cid* column in each table stores policy IDs. The *pov* (point-of-view) column in the *ctx_gists* table stores the theme for each theme summary.

> **See Also:** *Oracle8 SQL Reference*, *Oracle8 ConText Cartridge Application Developer's Guide*

Generate Linguistic Output for Documents

To request a list of themes for a document, call the REQUEST_THEMES in the CTX_LING PL/SQL package. To request a document Gist and/or theme summaries for the document themes, call the REQUEST_GIST procedure in CTX_LING.

Then call the SUBMIT function in CTX_LING to submit the request(s) to the Services Queue and return a handle for each submitted request. If SUBMIT is called after REQUEST_THEMES and REQUEST_GIST are both called, the requests are submitted as a single batch request and a single handle is returned.

The requests in the Services Queue are picked up and processed by the first available ConText servers with the Linguistic personality. After linguistic output is generated, you can query for lists of themes in documents and use Gists / theme summaries to view summarized versions of your documents.

Note: REQUEST_THEMES and REQUEST_GIST must be called once for each document for which you want to generate the respective type of linguistic output.

In addition, you must specify a policy name for REQUEST_ THEMES and REQUEST_GIST. The specified policy can be *either* a text indexing policy *or* a theme indexing policy; however, if you are generating both themes and Gists / theme summaries for a document, you should use the same policy.

In the following example, themes and paragraph-level Gists / theme summaries are requested for document 1 (*pk*). The document is stored in *ctxdev.docs.text* for the *ctx_ thidx* theme indexing policy. The *ctx_themes* and *ctx_gist* tables are specified as the output tables.

Then, the SUBMIT function is called for both requests.

```
exec ctx_ling.request_themes('ctx_thidx',1,'ctx_themes')
exec ctx_ling.request_gist('ctx_thidx',1,'ctx_gist')

variable handle number
exec :handle := ctx_ling.submit
print handle
```

Note: In this example, the *ctx_thidx* policy is used to call REQUEST_THEMES and REQUEST_GIST; however, you could also use the *ctx_docs* text indexing policy discussed in this manual to generate the same results.

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

Example Queries for Linguistic Output

Theme and Gist information is stored in the linguistic output tables and can be used to present a specialized view of a document.

For example, you may want to display the themes for all the documents in a column or for a single document. You may also want to display the theme summary for all documents with a specific theme or the Gist for a specific document.

Note: The Linguistic personality is required *only* for generating linguistic output for documents; neither the Linguistic nor Query personalities are required for querying the linguistic output tables.

In addition, because linguistic information is stored in output tables separate from the base text table, you must join the text table and the linguistic output tables to return detailed information for the document hits.

The following example returns the themes (*theme)* and theme weight (*weight*), as well as the title (*title*) for each document in *docs* that has *Smith* as the author:

```
select theme, weight, title from ctx_themes, docs,
where docs.author='Smith'
and ctx_themes.pk=docs.pk
order by weight desc;
```

The following example returns the title (*title*) and theme summary (*gist*) for each document that has *computer software industry* as one of its themes (*pov*):

```
select title, gist from ctx_gist, docs
where ctx_gist.pk=docs.pk
and pov = 'computer software industry';
```

Note: ConText stores themes as plural nouns or noun phrases. When you perform queries using linguistic output, the query phrase must be entered exactly as it is stored in the output table or the query will likely return no results.

A list of available words or phrases for queries can be obtained by selecting all *unique* themes from the themes/Gist output table.

See Also: *Oracle8 ConText Cartridge Application Developer's Guide*

4

Implementing ConText

This chapter provides basic information for implementing ConText.

The chapter covers the following topics:

- Pre-implementation Checklist
- Implementation Questions

Pre-implementation Checklist

This QuickStart manual assumes that Oracle8 is installed and running. In addition, the manual assumes that basic Oracle implementation tasks have been performed.

Before you proceed to the ConText implementation questions, verify the following:

- ❏ Oracle8 installed and running
- ❏ Tablespaces created
- ❏ Oracle users created and granted necessary database roles

---

Note: The Oracle user who performs the QuickStart implementation must be granted the CONNECT and RESOURCE roles.

See Also: *Oracle8 SQL Reference*, Oracle8 installation documentation specific to your operating system

Implementation Questions

This section provides a list of ConText implementation questions that you need to answer before you can begin the QuickStart tasks:

1. Have you created tables to store your text?

2. Is text loaded into your tables?

3. Is ConText installed?

4. Is the ConText initialization parameter enabled?

5. Is CTXAPP granted to users who own tables?

The answers you provide depend on whether text is already stored in Oracle8 or you are storing text for the first time.

Note: This QuickStart manual presents an example Oracle user named *ctxdev* and an example text table named *docs*; however, the *ctxdev* user and the *docs* table are not provided by ConText. The QuickStart examples assume that *docs* (or its equivalent) is in the database schema for *ctxdev* (or its equivalent).

---

1. Have you created tables to store your text?

YES: Proceed to "2. Is text loaded into your tables?" >

NO: You need to create one or more tables for storing your text. Each table used to store text must have *at least* the following columns:

- primary key (one or more columns that uniquely identify each document)
- column for storing text (recommended datatypes: CHAR, VARCHAR2, LONG, LONG RAW, BLOB, CLOB, BFILE)

To create tables, log in to SQL*Plus as the Oracle user who will own the tables and use the CREATE TABLE command.

See Also: *Oracle8 SQL Reference*

2. Is text loaded into your tables?

YES: Proceed to "3. Is ConText installed?" >

NO: You need to load documents (text) into your table(s). ConText supports three types of storage for documents:

Storage Type	Description
Internal	A document is represented as either a single row or multiple rows in a database table. The text of the document is stored in one of the table columns. Each row is indexed by ConText as a separate entity.
Internal master-detail	A document is represented as multiple rows in a database table. The text of the document is stored in one of the table columns. The collection of rows for the document are indexed by ConText as a single entity
external	The text of the documents is stored in files and pointers to the files are stored in one of the table columns. The pointers can be operating system directories / file names or Uniform Resource Locations (URLs). Each row is indexed by ConText as a separate entity.

You can use the following methods for loading documents into a table:

- INSERT (SQL command) to load a single block of plain (ASCII) text into a row
- SQL*Loader to batch load plain (ASCII) text
- ctxload (utility provided with ConText) to batch load plain and formatted text into LONG and LONG RAW columns

You can also use a ConText server with the Loader personality to automate loading text from operating system files into a LONG or LONG RAW column.

The tool / utility you use depends on the structure and usage of the text column. For example, ctxload *only* works for tables containing a LONG or LONG RAW column.

> **See Also:** *Oracle8 SQL Reference, Oracle8 Utilities, Oracle8 ConText Cartridge Administrator's Guide*

3. Is ConText installed?

YES: Proceed to "4. Is the ConText initialization parameter enabled?" >

NO: You must install Oracle8 ConText Cartridge on a server machine.

> **See Also:** Oracle8 installation documentation specific to your operating system

4. Is the ConText initialization parameter enabled?

YES: Proceed to "5. Is CTXAPP granted to users who own tables?" >

NO: If you want to use one-step queries in ConText, you need to set the initialization parameter TEXT_ENABLE to TRUE.

You can set the TEXT_ENABLE parameter by adding the following line to your init*sid*.ora file and restarting the database:

```
text_enable = true
```

You can also use the SQL command, ALTER SESSION, to set TEXT_ENABLE for the current session.

> **See Also:** *Oracle8 ConText Cartridge Administrator's Guide*

5. Is CTXAPP granted to users who own tables?

YES: You are done with the required implementation tasks for ConText!

Go to Chapter 2, "Using Text Queries" >>

or

Chapter 3, "Using Theme Queries and CTX_LING" >>

NO: You must grant the ConText role, CTXAPP, to Oracle users who own the tables in which text is stored. The CTXAPP role allows a user to create a text index for a column.

To grant CTXAPP to users, log in to SQL*Plus as the CTXSYS user and use the GRANT command.

For example:

```
grant CTXAPP to CTXDEV;
```

In this example, CTXAPP is granted to a user named *ctxdev*.

Note: *ctxdev* is an example Oracle user used in QuickStart; the *ctxdev* user is not provided by ConText and must be created manually if you wish to use it in your installation.

See Also: *Oracle8 SQL Reference, Oracle8 ConText Cartridge Administrator's Guide*