IBM DB2 Universal Database

Net Search Extender Administration and User's Guide

Version 8.1

SH12-6740-02

IBM DB2 Universal Database

Net Search Extender Administration and User's Guide

Version 8.1

SH12-6740-02

Note!
Before using this information and the product it supports, be sure to read the general information under Appendix O, "Notices", on page 285.

Third Edition, June 2003

This edition applies to Version 8.1 of IBM DB2 Universal Database Net Search Extender, program number 5765-F38, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SH12-6740-01.

© Copyright International Business Machines Corporation 1995, 2003. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1995, 2003

About this book

This book describes how to use the IBM DB2 Universal Database™ Net Search Extender to prepare and maintain a DB2® database for retrieving text data. It also describes how you can use the SQL functions provided to access and manipulate these types of data. By incorporating DB2 Net Search Extender's functions in your program's SQL statements, you can create powerful and versatile text-retrieval programs.

References in this book to ″DB2″ refer to DB2 UDB.

Who should use this book

This book is intended for DB2 database administrators who are familiar with DB2 administration concepts, tools, and techniques.

This book is also intended for DB2 application programmers who are familiar with SQL and with one or more programming languages that can be used for DB2 application programs.

How to use this book

This book is structured as follows:

"Part 1. User's Guide"

This part gives an overview of DB2 Net Search Extender, describes how to install and set it up, and discusses planning considerations. It also describes how to prepare and maintain a DB2 database so that you can search for text.

Read this part if you are new to DB2 Net Search Extender and want to learn how to use the DB2 Net Search Extender functions to search for text.

"Part 2. Reference"

This part presents reference information for DB2 Net Search Extender functions and commands.

Read this part if you are familiar with DB2 Net Search Extender concepts and tasks, but need information about a specific DB2 Net Search Extender function or command.

"Part 3. Appendixes"

This part provides additional reference information for DB2 Net Search Extender. It contains information on migration, memory usage, views, document models, messages, and codes.

© Copyright IBM Corp. 1995, 2003

About this book

Read this part if you need specific reference information about DB2 Net Search Extender.

How to read the syntax diagrams

Throughout this book, syntax is described using the structure defined as follows:

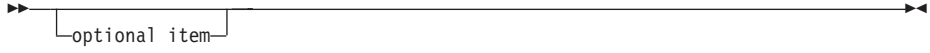
- Read the syntax diagrams from left to right and top to bottom, following the path of the line. The ►►— symbol indicates the beginning of a statement.

  The —► symbol indicates that the statement syntax is continued on the next line.

  The ►— symbol indicates that a statement is continued from the previous line.

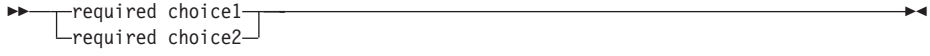
  The —►◄ symbol indicates the end of a statement.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.



- If you can choose from two or more items, they appear in a stack.

  If you *must* choose one of the items, one item of the stack appears on the main path.



  If choosing none of the items is an option, the entire stack appears below the main path.

  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items.

- Keywords appear in uppercase; they must be spelled exactly as shown. Variables appear in lowercase (for example, `srcpath`). They represent user-supplied names or values in the syntax.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Related information

IBM DB2 Universal Database Version 8

- *IBM DB2 Universal Database Quick Beginnings Version 8* for DB2 Servers (GC09-4836), for DB2 Clients (GC09-4832), for DB2 Connect Personal Edition (GC09-4834), for DB2 Personal Edition (GC09-4838), and IBM Data Links Manager (GC09-4829-00). These books describe how to plan for, install, configure, and migrate to DB2 Universal Database on the appropriate platform.
- *IBM DB2 Universal Database Administration Guide Version 8* Planning (SC09-4822), Performance (SC09-4821), and Implementation (SC09-4820). These books describe how to design and implement a DB2 database.
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1 Version 8* (SC09-4849). This book describes how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification.
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2 Version 8* (SC09-4850). This book describes how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification.
- *IBM DB2 Universal Database Command Reference Version 8* (SC09-4828). This book describes how to use the DB2 command line processor and gives reference information about DB2 commands.
- *IBM DB2 Universal Database Replication Guide and Reference Version 8* (SC27-1121). This book describes how to plan, set up, and maintain a DB2 data replication environment.

IBM DB2 Universal Database Enterprise Server Edition Version 8

- *IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition Version 8* (GC09-4833). This book describes how to plan for, install, and configure DB2 Universal Database Enterprise-Extended Edition on the pertinent platform.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 Extenders documentation. You can use the following method to provide comments:

- Send your comments by e-mail to swsdid@de.ibm.com. Be sure to include the name of the book, the part number of the book, the version of the product, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. This guide suggests information that you can gather to help DB2 Customer Support to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the United States, then you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product information

If you live in the United States, then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or get general information
- 1-800-879-2755 to order publications

http://www.ibm.com/software/data/db2/
The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more.

http://www.ibm.com/software/data/support/
The DB2 support Web pages provide access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

Note: This information may be in English only.

http://www.ibm.com/software/data/db2/extenders/
The DB2 Extenders Web pages provide information on all the currently available DB2 Extenders. These include DB2 XML Extender, DB2 Spatial Extender, and DB2 AIV Extender.

http://www.ibm.com/software/data/db2/extenders/support/
The DB2 Extenders support Web pages provide access to frequently asked questions, hints and tips, fixes, and documentation.

http://www.ibm.com/software/data/db2/extenders/netsearch/index.html
The DB2 Net Search Extender page provides information on the latest performance tips.

www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi
The Publications Center provides information on how to order or download publications.

http://www.ibm.com/certify/index.html
The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products, including DB2.

On Compuserve: GO IBMDB2
Enter this command to access the IBM DB2 Family forums. All DB2 products are supported through these forums.

For information on how to contact IBM outside of the United States, refer to Appendix A of the *IBM Software Support Handbook*. To access this document, go to the following Web page:
http://techsupport.services.ibm.com/guides/contacts.html

Note: In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

Summary of changes

DB2 Net Search Extender V8.1 Fix Pack 2, introduces the following new features:

- Support for partitioned databases
- The `db2ext.highlight` function
- Using DB2 Replication to create and maintain text indexes on nicknames with incremental index update
- Support for Outside-In filtering software by Stellent

© Copyright IBM Corp. 1995, 2003

Part 1. User's Guide

© Copyright IBM Corp. 1995, 2003

Chapter 1. Overview and concepts

DB2 Net Search Extender Version 8.1 is one of a family of DB2 Extenders™.

It replaces DB2 Text Information Extender Version 7.2, and Net Search Extender Version 7.2 and offers users and application programmers a fast, versatile, and intelligent method of searching full-text documents stored in DB2, other databases, and file systems using SQL queries.

Key concepts

To fully understand the capabilities of DB2 Net Search Extender, it is necessary to understand key terms, which are found **bold** in this section, and the various options available. It is also necessary to have a basic understanding of DB2 Universal Database concepts and terms.

Basically, DB2 Net Search Extender searches **text documents** that are held in the column of a database table.

The text documents must be uniquely identifiable. Net Search Extender uses the **primary key** of the table for this purpose.

The documents can be in various formats, such as HTML or XML.

Rather than sequentially searching through the text documents that would take a considerable amount of time, Net Search Extender creates a **text index** in order to make documents searchable.

A text index consists of significant **terms** that are extracted from the text documents.

© Copyright IBM Corp. 1995, 2003

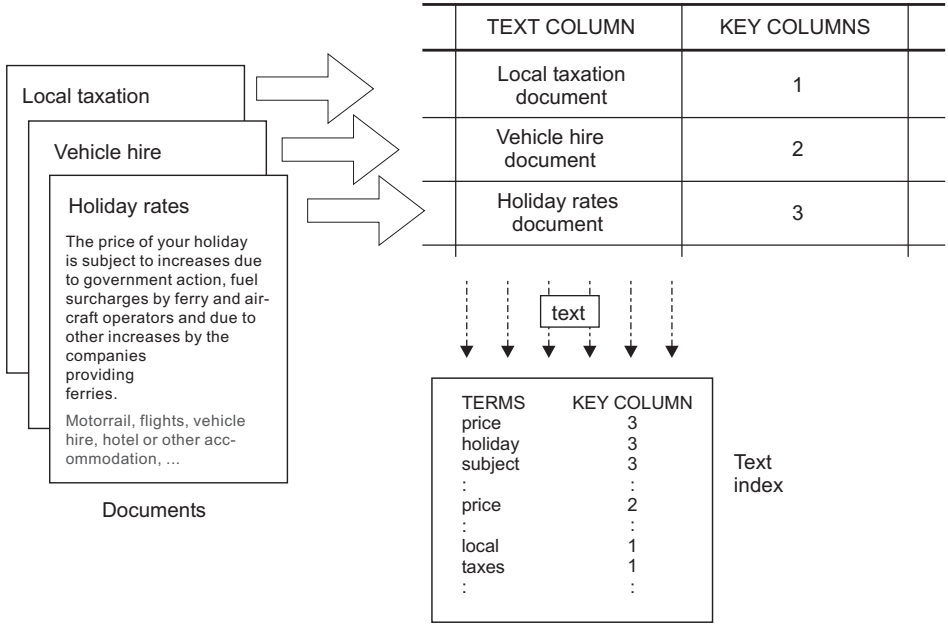


Figure 1. Creating a text index

Text index creation is the process of defining and declaring the properties of the index, such as the location of the index. After creation, the text index contains no data. **Index update** is the process of adding data to the text index. The first index update adds all text documents from the text column to the index. It is known as the **initial update**.

By using a text index for searching, there are synchronization issues between the table and the text index that must be taken into account, as any follow-up changes to the table, such as additions, deletions, and updates to the text documents must be reflected in the text index.

Synchronization in Net Search Extender is based on **triggers** that automatically store information about new, changed, and deleted documents in a **log table**. There is one log table for each text index. Applying the contents of the log table to its corresponding text index is called **incremental update**.

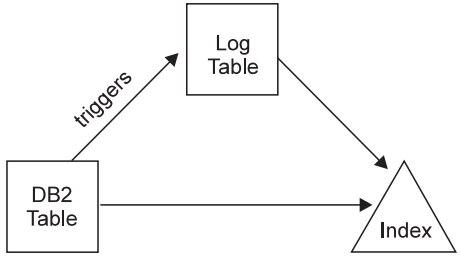


Figure 2. Incremental update process

You can update the text index by using a **manual** or **automatic** option. The automatic option uses an update schedule to set days and times.

Note that **neither** of these options synchronizes the text index within the scope of a transaction that updates, deletes, and inserts text documents. Net Search Extender's asynchronous text indexing improves performance and concurrency. The update is applied within a separate transaction to a copy of a very small part of the index. The index is only locked for read access during a very short period of time when the copy is put in place of the original. This is invisible to search operations, see Chapter 5, "Net Search Extender instance services", on page 29 for information.

A text index has certain properties, such as index file location and automatic update properties. If necessary, you can change some of the properties. This is known as **altering** the index.

One such property is whether the `ORDER BY` phrase should presort the text index on the table columns. In such a case, the initial update will index the text document in the order specified, and return the search results in this order.

For example, specifying presorted book abstracts according to the book price. When looking for the least expensive books about relational database systems, you can restrict your text search to return only the first couple of books as these will be the cheapest. However, without presorted indexes, you would have to search for all books and join these with the cheapest books, which would be a more costly operation.

Net Search Extender allows several presorted indexes per text column. For example, one index for presorting books according to the date of publication and, a second presorting books according to the price.

Usually the first update after creating a text index is an initial update, and the following updates are incremental. However, when working with presorted

indexes you want to keep the order in case of updates. This is addressed by the **Recreate Index on Update** option, which totally rebuilds the index each time an update is performed.

After the text index is updated, you can search using one of the following options:

- An SQL scalar search function
- A stored procedure search
- An SQL table-valued function

As the search options have different operating characteristics, they are explained in the following sections.

Using an SQL scalar search function

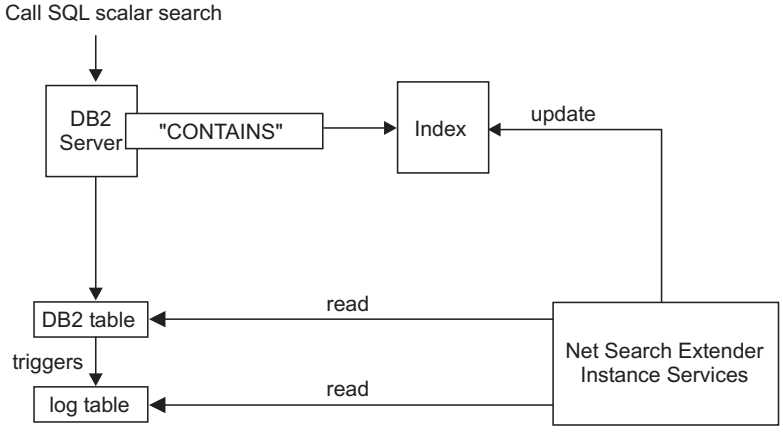


Figure 3. Using an SQL scalar search function for searching

Net Search Extender offers three scalar text search functions (`CONTAINS`, `NUMBEROFMATCHES`, and `SCORE`) that are seamlessly integrated within SQL. You can use the search functions in the same places that you would use standard SQL expressions within SQL queries. Typical queries are:

```
SELECT * FROM books WHERE CONTAINS (abstract,'"relational databases"') = 1
          AND PRICE <10

SELECT ISBN, SCORE (abstract, '"relational databases"') as SCORE
          from BOOKS
          where NUMBEROFMATCHES (abstract, '"relational databases"')
          >5 AND PRICE <10
          order by SCORE
```

The SQL scalar functions return an indicator to how well the text documents matched a given text search condition. Then the `SELECT` phase of the SQL query determines the information returned to the end user.

Use the SQL scalar search functions as the default search method. These search functions should be suitable for a majority of situations, especially when the text search expression is combined with other, different conditions.

Note that the DB2 Optimizer is aware of how many text documents can be expected to match a CONTAINS predicate and how costly different access plan alternatives will be. The optimizer will choose the cheapest access plan.

Using a stored procedure search

The stored procedure search works differently from the SQL scalar search functions. At text index creation, you have to specify which columns out of the table or view are returned to the end user. This data is stored in a **cache** in main memory. This enables the stored procedure search to return search results extremely quickly. The cache needs to be **activated** before it can be used and there is a corresponding **deactivate** command.

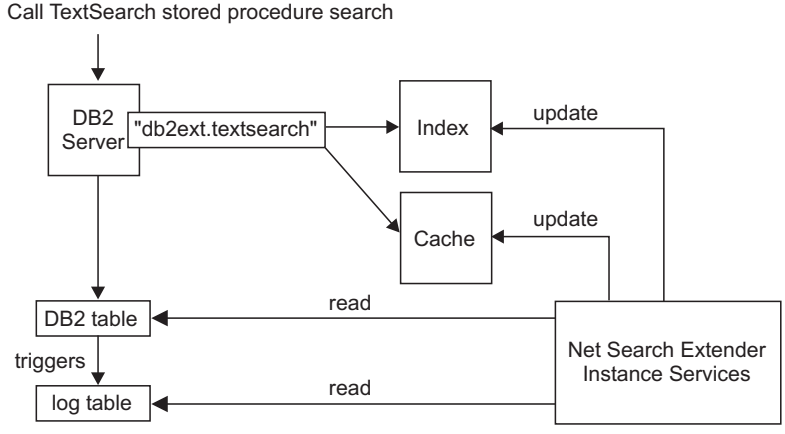


Figure 4. Using a stored procedure search

The `ACTIVATE` command loads data into either a temporary cache (which is created from scratch on activation), or a persistent cache, which is maintained on disk.

Using the stored procedure for searching requires memory calculations, such as how much memory is required and how much free memory should be left for index updates.

The stored procedure can work on text indexes that are created on views. However, as triggers can not be created on views, any changes are not automatically recognized. You can manually add the changed information to the log table, or you can work with the `RECREATE` option.

Use the stored procedure search for high performance/high scalability applications that are interested in text-search-only queries. For example, queries that do not need to join text search results with the results of other complex SQL conditions.

The main functional differences to the SQL scalar search functions are:

- The stored procedure search can not be used in arbitrary SQL queries, but is a query against a predefined cache table.
- The stored procedure search can exploit indexes on views.
- The stored procedure search can exploit multiple presorted text indexes on a column.

Note that for this option, a large amount of main memory must be available. For additional information, see Appendix B, “Using large amounts of memory”, on page 205.

Using an SQL table-valued function for searching

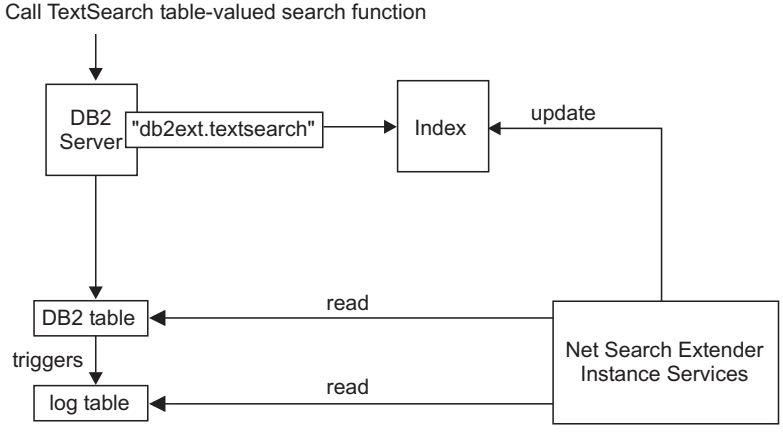


Figure 5. Using an SQL table-valued function for searching

The SQL table-valued function is a compromise between the SQL scalar search functions and the stored procedure search. With the SQL table-valued function you can also use a `db2ext.highlight` function to get information about why a document qualified as a search result.

The main functional differences to the stored procedure search are:

- No cache is necessary (and no cache is exploited).
- The table-valued function can be used in arbitrary SQL statements.
- A large amount of main memory is not necessary.

The main functional differences to the SQL scalar search functions are:

- The SQL table-valued function can exploit indexes on views.
- The table-valued function can exploit presorted text indexes.

Use the SQL table-valued function in those cases where you would normally use an SQL scalar function, but you want to exploit text indexes on views or presorted text indexes.

Additional concepts

As well as understanding the key concepts of DB2 Net Search Extender, there are also some additional concepts that need explaining.

For more information on developing Net Search Extender-based applications, see Chapter 4, “Planning”, on page 25.

Column transformation function

You can use your own function to convert a nonsupported format or data type into a supported format or data type. By specifying a User Defined Function (UDF), you can get the original text document as input. The output from the UDF should be a supported format, that can be processed during indexing.

You can also use this feature for indexing documents that are stored on external nonsupported data stores. In this case, the DB2 column contains document references and the function returns the document contents that have the relevant document reference.

See “Creating a text index on a nonsupported data type” on page 40.

Instance services

Net Search Extender Instance Services take care of index-specific locking services and text index update services (both automatic and manual).

See Chapter 5, “Net Search Extender instance services”, on page 29 for more information.

Externally stored data

In a majority of cases, the data on which you create a text index is stored within native DB2 table columns, such as in `CLOBS` or `VARCHARS`.

However, text documents that are stored externally, such as in files or other databases, are also supported. For documents that are stored in files, the DB2 Data Links feature is available. For documents that are stored on other databases, use DB2 nickname tables to create a text index.

See “Related information” on page ix.

Additional concepts

You can also use the column transformation function for data that are stored in nonsupported external data stores. See "Column transformation function" on page 9.

Administration views

There are several views available in DB2 Net Search Extender. They provide information on the text indexes and their properties.

See Appendix C, "Net Search Extender information catalogs", on page 209 for information.

Partitioned database support

The search functions of DB2 Net Search Extender use partitioned database support in the following ways:

- The SQL scalar function exploits indexes that are created on a partitioned table.
- The stored procedure search and the SQL table-valued function exploit only tables on one node in a partitioned environment.

Indexes on nicknames in a federated database

You can also create a text index on nicknames in a federated database that points to tables in a remote database. In this case, the role of the log table (for incremental index updates) differs from its role for an index on a regular table. Unlike regular tables, DB2 triggers can not be created on nicknames, so that change information about documents can not be inserted into a log table using triggers. Therefore, there are two different ways for incremental updates to create an index on a nickname:

- The log table is created locally in the federated database and the application is responsible that the log table contains correct change information on the nickname. For DB2 views, this is similar to the incremental index update. This option is the default option.
- DB2 Replication has been set up so that changes to the table referenced by the nickname are captured in a so-called "Change Data Table" (CD-table) for DB2 remote databases, or a "Consistent Change Data Table" (CCD-table) for non-DB2 relational databases. DB2 Net Search Extender can then use the CD or CCD table instead of creating a log table for an index on a nickname. In this case, you must specify the capture table characteristics in the DB2TEXT CREATE INDEX command.

Key features

DB2 Net Search Extender Version 8.1 has the following key features:

- Indexing
  - Fast indexing of very large data volumes
  - Dynamic updating of indexes

  - Storing table columns in main memory at indexing time to avoid expensive physical read operations at search time
  - A choice of command line or interface via the DB2 Control Center for indexing
  - Different text formats, for example HTML and XML
  - Support of third-party filtering software ″Outside-In″
  - Nickname table support
  - DB2 Data Link Manager support
  - Support of presorted text indexes
  - Partitioned database support
- Search
  - Boolean operations
  - Proximity search for words in the same sentence or paragraph
  - “Fuzzy” searches for words having a similar spelling as the search term
  - Wildcard searches, using front, middle, and end masking, for whole words and single characters
  - Free-text searches. For documents containing specific text, the search argument is expressed in natural language
  - A highlight function to show why a particular document qualified as a search result
  - Thesaurus support
  - Restrict searching to sections within documents
  - Numeric attribute support
  - High-speed searching through a large number of text documents with many concurrent users
- Search results
  - You can specify how the search results are sorted at indexing time
  - You can specify search result subsets when searching large data volumes and expecting large result lists
  - You can set a limit on search terms with a high hit count
  - Built-in SQL functionality combined with the DB2 Optimizer automatically selects the best plan according to the expected search results

DB2 Net Search Extender in the DB2 client/server environment

DB2 Net Search Extender search functionality is integrated into SQL and executed at the server. Therefore, you do not need to install Net Search Extender on the client to issue text search queries.

Key features

DB2 Net Search Extender supports administration calls to the server from the client side. Either install DB2 Net Search Extender on the client sides and the server sides, or alternatively, use the DB2 Control Center to administrate DB2 Net Search Extender from the client side.

Chapter 2. Installation

This chapter describes how to install DB2 Net Search Extender in UNIX® and Windows® systems.

After installation, run the DB2 Net Search Extender installation verification script.

System requirements

The following versions of software are required to run DB2 Net Search Extender:

- DB2 Version 8.1 Fix Pack 2
- Java Runtime Environment (JRE). The JRE version depends on the DB2 version.

DB2 Net Search Extender is available on the following operating systems:

Solaris Operating Environment:

- Solaris 7
- Solaris 8
- Solaris 9

AIX®:

- AIX Version 4.3.3. The following file set is also required: `xlC.aix43.rte 5.0.2.x`.
- AIX Version 5.1.0/5.2.0. The following file set is also required for 64-bit code support: `xlC.aix50.rte 5.0.2.3` or higher.

Note that you can download the AIX file sets from:
http://techsupport.services.ibm.com/server/fixes

Windows:

- Windows NT® Version 4
- Windows 2000
- Windows XP

HP:

- HP-UX 11i

© Copyright IBM Corp. 1995, 2003

Linux:

- Only Intel machines support DB2 Net Search Extender. The validation status for new Linux kernels and distributions is frequently updated. To obtain the latest information for supported Linux software levels, refer to: http://www.ibm.com/software/data/db2/linux/validate

For AIX, Solaris, and HP-UX, DB2 Net Search Extender is available as a 32–bit and 64–bit application. On Windows and Linux, Net Search Extender is only available as a 32–bit application.

For all DB2 Net Search Extender operating systems, the minimum memory requirement is 30MB. The minimum disk space for a typical DB2 Net Search Extender installation is 50MB.

DB2 Net Search Extender has the same minimum software and hardware requirements as DB2 Universal Database Version 8.1. For these requirements, as well as specific operating system patches, refer to the corresponding section in the *IBM DB2 Universal Database Quick Beginnings Version 8* documentation.

Any additional hardware requirements depend on the size and type of text index selected. For DB2 documentation, see "Related information" on page ix.

Installation overview for a partitioned DB2 server (UNIX)

Ensure the correct installation and configuration of DB2 on every node. After installing DB2, you need to install DB2 Net Search Extender on each computer.

Note

A fenced user ID that is different from the instance owner ID does not work with partitioned databases.

Installation on UNIX

To install on UNIX, follow these steps:

1. Install the product.
2. Update the DB2 instance.

Step 1 for UNIX: Install the product components

To install on UNIX, follow these steps:

1. Log on at the target machine as the root user.
2. Change to the correct directory for your platform:
   - `cd /<cdrom>` where `<cdrom>` is your CD-ROM driver path.
   - `cd /<platform>`

3. Call `./nsesetup.sh` and follow the instructions displayed on the screen. Note: Ignore any of the `'Exited with'` messages.

Step 2 for UNIX: Update the DB2 instance

To update the DB2 instance, follow these steps:

1. Ensure that you are active as the root user.
2. Depending on the platform, use one of the following commands:
   - For AIX: `cd /usr/opt/db2_08_01/instance`
   - For Solaris, Linux, HP-UX: `cd /opt/IBM/db2/V8.1/instance`
3. Run `db2iupdt` using `./db2iupdt <db2instance>`, where `<db2instance>` is an existing DB2 instance user ID that you would like to use with Net Search Extender.
4. Log out.

> **Note**
> Net Search Extender automatically creates new DB2 instances during `db2icrt`.

Windows installation

To install on Windows you must log on with a user ID that has administrative rights, and then follow these steps:

1. Use the `<cdrom>:\windows\install\setup.exe` to transfer the files from the package to the target machine. Note that for every DB2 service, you must enter a user ID and password to create the correct DB2 Net Search Extender service.
2. Reboot the system after data transfer.
3. Call `db2text start` to start the DB2 Net Search Extender Instance Services.

> **Note**
> Every DB2 instance creates a Windows service. Make sure that the DB2 instance services run under a user account and not under the systems account.

Directory names and file names

You must specify the directory names and file names in SBCS characters for all Net Search Extender commands. The maximum length of the path names (including the file name) is 256 bytes or less.

Installing the Outside-In libraries

To use Net Search Extender with the Outside-In software from Stellent, you must set up the libraries for each platform:

- On Windows, ensure that the directory where the libraries are located is added to the path environment variable.
- On UNIX, add all the Outside-In libraries into the `/opt/IBM/db2/V8.1/lib` directory.

Installation verification

Net Search Extender installation verification is available on Windows and UNIX platforms.

Note

For distributed databases, use the following verification sample:

```
nsesample_partitioned database_name [node_number][table_space_filename]
```

Installation verification on Windows

Complete the following steps to verify that Net Search Extender is correctly installed.

- Follow these steps to call the administration script `nsesample.bat` to set up the text indexes:
  1. Call `db2cmd` to open a DB2 command window.
  2. Change to `<sqllib>\samples\db2ext`
  3. From the DB2 command window, call `nsesample.bat <yourdb>` where `<yourdb>` is the name of a database. Note that this command creates the database if it does not already exist.
  4. Check the generated output file `nsesample.log` in the current directory.
- Then call the following sample queries to execute in the DB2 command window:
  1. Connect to your database using `db2 connect to <yourdb>`
  2. Execute the sample queries using `db2 -tvf search`
  3. Check the results of the queries contained in the script. Note that every query should return one or more hits.

If there are no errors in the `nsesample.log` file and all the queries are working, Net Search Extender is successfully installed.

Installation verification on UNIX

Complete the following steps to verify that Net Search Extender is correctly installed.

- Follow these steps to call the administration script `nsesample` to set up the text indexes:
  1. Change to `<instance_owner_home>/sqllib/samples/db2ext`
  2. Call `./nsesample <yourdb>`. Note that this command creates the database if it does not already exist.
  3. Check the generated output file `nsesample.log` in your home directory.
- Then call some sample queries to execute in the same DB2 command window:
  1. Connect to your database using `db2 connect to <yourdb>`
  2. Execute the sample queries using `db2 -tvf search`
  3. Check the results of the queries contained in the script. Note that every query should return one or more hits.

If there are no errors in the `nsesample.log` file and all the queries are working, Net Search Extender is successfully installed.

Note

For migration information, see Appendix A, “Migration”, on page 201.

Installation verification

Chapter 3. User scenarios

Use this chapter to learn about Net Search Extender by using the following walk-through examples:

The SQL scalar search example
This command line example demonstrates the indexing and search functions available.

The stored procedure example
This command line example uses the index command from the example above. With the addition of a cache however, the example demonstrates the different indexing and search functions available.

The SQL table-valued function example

For more information on using Net Search Extender, see the following chapters:

- Chapter 4, “Planning”, on page 25
- Chapter 6, “Creating and maintaining a text index”, on page 33
- Chapter 8, “Searching”, on page 77

Note
Before using the examples, ensure that Net Search Extender is installed successfully by using the installation verification procedure.

Simple example with the SQL scalar search function

Use the following steps in the DB2 Net Search Extender example:

1. Creating a database
2. Enabling a database for text search
3. Creating a table
4. Creating a full-text index
5. Loading the sample data
6. Synchronizing the text index
7. Searching with the text index

You can issue the sample commands on the command line of the operating system by using an existing database. For the following examples, the database name is `sample`.

© Copyright IBM Corp. 1995, 2003

Creating a database

You can create a database in DB2 by using the following command:

```
db2 "create database sample"
```

Enabling a database for text search

You can issue DB2 Net Search Extender commands in the same way as DB2 commands on the command line of the operating system. For example, use the following command to start Net Search Extender Instance Services:

```
db2text "START"
```

Use the following command to prepare the database for use with DB2 Net Search Extender:

```
db2text "ENABLE DATABASE FOR TEXT CONNECT TO sample"
```

You need to do this step only once for each database.

Creating a table

```
db2 "CREATE TABLE books (isbn VARCHAR(18) not null PRIMARY KEY,
      author VARCHAR(30), story LONG VARCHAR, year INTEGER)"
```

This DB2 command creates a table called `books`. It contains columns for the `author`, `story`, `isbn number`, and the `year` the book was published.

Creating a full-text index

```
db2text "CREATE INDEX db2ext.myTextIndex FOR TEXT ON books (story)
         CONNECT TO sample"
```

This command creates a full-text index for the column `story`. The name of the text index is `db2ext.myTextIndex`

Loading the sample data

```
db2 "INSERT INTO books VALUES ('0-13-086755-1','John', 'A man was
     running down the street.',2001)"
db2 "INSERT INTO books VALUES ('0-13-086755-2','Mike', 'The cat hunts
     some mice.', 2000)"
db2 "INSERT INTO books VALUES ('0-13-086755-3','Peter', 'Some men
     were standing beside the table.',1999)"
```

These commands load the `isbn`, `author`, `story`, and `publishing year` for these books into the table.

Synchronizing the text index

To update the text index with data from the `sample` table, use the following command:

```
db2text "UPDATE INDEX db2ext.myTextIndex FOR TEXT CONNECT TO sample"
```

Searching with the text index

To search the text index, use the following CONTAINS scalar search function:

```
db2 "SELECT author, story FROM books WHERE CONTAINS
    (story, '\"cat\"') = 1 AND YEAR >= 2000"
```

Note

Depending on the operating system shell you are using, you might need a different escape character in front of the double quotes surrounding the text search phrase. The above example, uses "\" as an escape character.

This query searches for all books about the term cat that are greater or equal to the year 2000. The query returns the following result table:

```
AUTHOR    STORY
Mike   The cat hunts some mice.
```

Other functions supported include SCORE and NUMBEROFMATCHES. SCORE returns an indicator on how well the search argument describes a found document. NUMBEROFMATCHES returns how many matches of the query terms are found in a resulting document.

Simple example with cache usage and stored procedure search

Use the following steps in the DB2 Net Search Extender stored procedure search example:

1. Creating a text index with cache option.
2. Synchronizing the index and activating the cache.
3. Searching with the TEXTSEARCH Stored Procedure.

Note

The stored procedure example assumes that the steps from the previous example are complete and that the database is still enabled.

Creating a text index with cache option

As the database is already enabled, use the following command to create a full-text index:

```
db2text "CREATE INDEX db2ext.mySTPTextIndex FOR TEXT ON books (story)
        CACHE TABLE (author, story) MAXIMUM CACHE SIZE 1
        CONNECT TO sample"
```

In this example, the full-text index is for the column `story` and it specifies a cache table on the columns `author` and `story`. The name of the text index is `mySTPTextIndex`.

Synchronizing the index and activating the cache

To update the index according to the data inserted into the table, use the following command:

```
db2text "UPDATE INDEX db2ext.mySTPTextIndex FOR TEXT CONNECT TO sample"
```

To activate the cache, use the following command:

```
db2text "ACTIVATE CACHE FOR INDEX db2ext.mySTPTextIndex FOR TEXT
        CONNECT TO sample"
```

Searching with the TEXTSEARCH Stored Procedure

You can only use the DB2 Net Search Extender stored procedure in certain cases. For details, see "Using a stored procedure search" on page 7.

```
 db2 "call db2ext.textSearch
        ('\"cat\"','DB2EXT','MYSTPTEXTINDEX',0,2,0,0,?,?)"
```

This query searches for all books about a `cat`, but only returns the first two results. In this case, the query returns the following result table:

```
Value of output parameters
--------------------------
Parameter Name  : SEARCHTERMCOUNTS
Parameter Value : 1
Parameter Name  : TOTALNUMBEROFRESULTS
Parameter Value : 1

AUTHOR    STORY
Mike      The cat hunts some mice.

Return Status = 0
```

For more samples about the search syntax check the following file: `sqllib\sample\db2ext\search`

For details about the other parameters used in the query, see "Searching for text using a stored procedure search" on page 85.

Simple example with the SQL table-valued function

You can use the SQL table-valued function on the text indexes created in the previous examples.

The SQL table-valued function query corresponds to the previously used CONTAINS query. See "Synchronizing the text index" on page 20 for information.

```
db2 "SELECT author, story FROM books b, table (db2ext.textsearch
     ('\"cat\"','DB2EXT','MYTEXTINDEX', 0, 2, CAST
     (NULL AS VARCHAR(18)))) T where T.primKey = b.isbn
```

For more details, see "Using an SQL table-valued function for searching" on page 8.

Note

The CAST (NULL AS VARCHAR(18)) calls the corresponding table-valued function to the primary key for the table books.

For each primary key type there is a table-valued function. This identifies the correct table-valued function for DB2.

User scenarios

Chapter 4. Planning

To use DB2 Net Search Extender in the most effective way, it is essential that some planning occurs prior to development. This development should involve several groups including those in database administration, interface and system designers, system architects, and developers.

The following sections provide a guide to the areas that you should consider.

For more information on developing DB2 Net Search Extender based applications, see the following chapters:

- Chapter 5, “Net Search Extender instance services”, on page 29
- Chapter 6, “Creating and maintaining a text index”, on page 33
- Chapter 8, “Searching”, on page 77

Directory locations and index storage

The disk space you need for an index depends on the size and type of data you want to index. As a guide, for indexing single-byte documents you need to reserve disk space of about 0.7 times the size of the documents you want to index. For double-byte documents, reserve the same disk space as the size of the documents you want to index.

The amount of space needed for the temporary files in the work directory is 1.0 to 4.0 times the amount of space needed for the final index file in the index directory.

If you have several large indexes, you should store them on separate disk devices, especially if you have concurrent access to the indexes during index update or search.

You must also specify the directory where you will store the text index. Ensure that there is enough disk space and that the DB2 instance owner has write access to the directory.

Document formats and supported code pages

DB2 Net Search Extender needs to know the format (or type) of text documents that you intend to search. This information is necessary for indexing text documents.

DB2 Net Search Extender supports the following document formats:

© Copyright IBM Corp. 1995, 2003

TEXT Plain text (for example, flat ASCII)

HTML Hypertext Markup Language

XML Extended Markup Language

GPP General Purpose Format (flat text with user-defined tags)

Outside-In (INSO)
Filtering software to extract textural content from PDFs and other common text formatting tools, for example, Microsoft Word. For further information, see "Outside-In filtering software" on page 27.

For the document formats HTML, XML, GPP, and the Outside-In filter formats, searching can be restricted to specific parts of a document. Chapter 9, "Working with structured documents", on page 91 explains how to define and work with document models.

Where Outside-In filters can not be used for nonsupported document formats, you can write a User Defined Function (UDF). This UDF must be specified at index creation time and converts the data from the nonsupported format to a supported format.

See "CREATE INDEX" on page 125 for more information.

You can index documents if they are in one of the supported Coded Character Set Identifiers (CCSIDs). These are also known as code pages. See Appendix D, "Supported CCSIDs", on page 219 for a list of these code pages.

To check the database code page, use the following DB2 command:

```
db2 GET DB CFG for <dbname>
```

For consistency, DB2 normally converts the code page of a document to the code page of the database. However, when you store data in a DB2 database in a column having a binary data type, such as `BLOB`, `FOR BIT DATA`, or a `datalink value`, DB2 does not convert the data, and the documents retain their original CCSIDs.

Note that having two different code pages might cause problems when creating a text index or searching. See "Creating a text index on binary data types" on page 39 for further information.

Outside-In filtering software

DB2 Net Search Extender supports the third-party document filtering software. Known as Outside-In from Stellent, you can use the software for extracting the textual contents from PDF files, or from documents written in the proprietary format of common text formatting tools without using native applications. Example formats include Microsoft Word and Lotus Word Pro.

You load the Outside-In libraries as plug-ins during UPDATE INDEX. The libraries are not part of Net Search Extender and need to be installed separately. You need to ensure that Net Search Extender can find the Outside-In libraries. For information on setting up and using the libraries, see "Installing the Outside-In libraries" on page 16.

The Outside-In software can not only generate textual content but also structural information, for example, fields. Net Search Extender can also customize which part of the Outside-In generated document information is to be stored in the index. To do this, you need to apply a specific type of document model, the Outside-In document model.

For information, see "Defining a document model for Outside-In filtered documents" on page 192.

The Stellent website is `http://www.stellent.com`.

To view a list of filtering formats, use the following url:

```
http://www.stellent.com/intradoc-cgi/nph-idc_cgi.exe/p31019225.pdf?IdcService=
GET_FILE&noSaveAs=1&Rendition=Web&RevisionSelectionMethod=
LatestReleased&allowInterrupt=1&dDocName=p31019225
```

Note

The Outside-In filtering software is only available on 32-bit instances. There is no 64-bit support available.

User roles

These are the different roles and authorizations for users of Net Search Extender:

DB2 instance owner

The DB2 instance owner user can start and stop the instance services for DB2 Net Search Extender and control the locking services. In addition, the DB2 instance user becomes DBADM for each enabled database. This enables a central point of control for all database changes driven by DB2 Net Search Extender.

Required DB2 authorizations
DBADM is granted on ENABLE DATABASE.

Required file system authorizations
Read and write access for all text index directories and read access to model files.

Commands for the instance owner
`DB2TEXT START`, `DB2TEXT STOP`, and `DB2TEXT CONTROL`.

The commands are only allowed on the server. In a distributed DB2 environment, this can be any of the servers. Each command checks if the user running the command is the DB2 instance owner. Note that using a separate fenced user ID on UNIX systems does not influence Net Search Extender processing in terms of authorization or performance.

Database administrators

Database administrators can enable and disable databases for use with DB2 Net Search Extender.

Required DB2 authorizations
DBADM (SYSADM for ENABLE DATABASE).

Commands for the database administrator
`DB2TEXT ENABLE DATABASE` and `DB2TEXT DISABLE DATABASE`.

Text table owners

The text table owner can create, drop, and change indexes. Note that they must be able to control the location of indexes and updates to the full-text indexes.

Required DB2 authorizations and privileges
Owner of text table.

Commands for the text table owner:
`DB2TEXT CREATE INDEX`, `DB2TEXT DROP INDEX`, `DB2TEXT ALTER INDEX`, `DB2TEXT ACTIVATE CACHE`, `DB2TEXT DEACTIVATE CACHE`, `DB2TEXT UPDATE INDEX`, `DB2TEXT CLEAR EVENTS`, and `DB2EXTTH`.

Note that the command implementation partially runs under the user ID of the DB2 instance owner. Therefore, grant the instance owner the necessary file system access before creating or altering the text indexes.

Using the command line or the DB2 Control Center interface for indexing

For indexing, you can either use the command line option, or the DB2 Control Center interface.

Chapter 5. Net Search Extender instance services

DB2 Net Search Extender Instance Services consist of the following services:

- Locking services
- Update services

This chapter explains how to start and stop the DB2 Net Search Extender Instance Services. It also discusses Locking Services and Update Services in detail.

Starting and stopping DB2 Net Search Extender

Before you can create a text index and search your documents, you have to start the DB2 Net Search Extender Instance Services.

To start the Instance Services, log on to the DB2 instance owner user ID (UNIX systems only) and enter the following command:

```
db2text start
```

To stop the Instance Services, enter the following command:

```
db2text stop
```

Note that there must be one Net Search Extender Instance Service per DB2 instance. The locking service maintains the locks for multiple databases.

Locking services

When you start DB2 Net Search Extender, the locking services automatically start. The locking services are needed to synchronize concurrent access to text indexes in Net Search Extender.

The locking services ensure that no two processes attempt to change a text index simultaneously, or that no process reads text index data while another process is making changes to the same text index data. Therefore, most processes request a lock on a text index before starting and release it again when processing has completed.

Note that the locking services for Net Search Extender text indexes must not be confused with DB2 locks that control access to DB2 tables.

Using the locking services

In Net Search Extender, there are different types of locks that control concurrent access to an index. The different locks depend on whether the text

© Copyright IBM Corp. 1995, 2003

index is only being read, as in the case of a search request, or if changes to the text index need to be computed and subsequently written to files.

During `db2text start`, the locking services automatically start. There are the following types of locks on a text index:

S-lock For shared read-only access. For example, search requests.

U-lock
For read and write access while computing changes to an index (update) with concurrent read access. For example, search requests.

X-lock For exclusive read/write access for a short period during which changes are actually written to the index.

IX-lock
For intended exclusive read/write access preventing any new S-locks while the update process is waiting for an X-lock.

There is one Net Search Extender locking service per DB2 instance. The locking service maintains the locks for multiple databases.

The locking services configuration file is `db2extlm.cfg`. It is stored on `<instance_owner_home>/sqllib/db2ext` for UNIX systems and on `<sqllib>\<DB2INSTANCE>\db2ext` for Windows.

Changes to the configuration file only take effect when Net Search Extender Instance Services are started during `db2text start`. See "CONTROL" on page 102 for further information. The user can set the following values:

- The maximum number of databases
- The maximum number of indexes per database
- The maximum number of allowed locks (concurrent users) per index
- Waiting times and the number of attempts to obtain a lock

The default values of the configuration file are as follows:

```
<default
         maxDbs         = "  8"
         maxIdxPerDb    = " 50"
         maxLocksPerIdx = "100"

         sWait = "  50"
         uWait = " 500"
         xWait = " 500"

         sAttempt = "50"
         uAttempt = "10"
         xAttempt = "60"
```

```
        latchTimeout = "80"

/>
```

The syntax is `<default attribute=value.../>` and the attributes have the following meanings:

maxDbs
The number of databases the locking services can handle (integer >1).

maxIdxPerDb
The number of indexes per database that can be locked (integer >1). This value is the same for all databases.

maxLocksPerIdx
The number of locks that can simultaneously exist on an index (integer >1). This value is the same for all indexes.

sWait/sAttempt
When requesting an S-lock, sAttempt is the number of attempts that are made if the lock is not granted immediately. sWait is the waiting time between these attempts (integer >1). These parameters also apply to IX-locks.

uWait/uAttempt
When requesting a U-lock, uAttempt is the number of attempts that are made if the lock is not granted immediately. uWait is the waiting time between these attempts (integer >1).

xWait/xAttempt
When requesting an X-lock, xAttempt is the number of attempts that are made if the lock is not granted immediately. xWait is the waiting time between these attempts (integer >1).

latchTimeout
This is additional waiting time for interval locking services. To determine the total waiting time for a lock, use the following calculation:

```
waiting time = # attempts * (# waits + (2 * # latchTimeout))
```

The waiting time is calculated in milliseconds. Note that with each attempt, the latchTimeout value is doubled when added to the overall waiting time.

Viewing a lock snapshot

You can view a lock snapshot by using one of the following commands:

- For a single text index:

  ```
  db2text CONTROL LIST ALL LOCKS FOR DATABASE mydatabase INDEX myindex
  ```

Viewing a lock snapshot

- For all the locked text indexes of a database:

  `db2text CONTROL LIST ALL LOCKS FOR DATABASE mydatabase`Note that only indexes that are actually locked are in the list.

The first time a text index is locked, memory is reserved for the database and the text index in the locking services. If further text indexes are locked, memory is also allocated for these indexes in the locking services. This memory is only freed again when the text index is dropped or the database disabled, or whenever the Net Search Extender services are restarted. This means that a text index or database consumes memory in the locking services, even if there are no locks currently set.

The command `"db2text CONTROL CLEAR ALL LOCKS"` forces the release of all the locks on a database or index. See "CONTROL" on page 102 for details on how to use this command. Note that this command does not free any memory allocated for the database or indexes. To free memory, you must either drop the index or disable the database, or restart the Net Search Extender services.

Update services

Update services start during `db2text start`. These services update the text index automatically at the specified times. Note that the text index is not immediately synchronized with the user table.

During index creation, you can specify how often Update Services check if an update of the index is required by using the following command:

```
db2text create index DB2EXT.TITLE for text on DB2EXT.TEXTTAB (TITLE)
     UPDATE FREQUENCY D(1,3) H(0,12) M(0) update minimum 5
```

In this example, every Monday and Wednesday at 12 p.m. and 12 a.m. the Update Services wake up and check if there is some work to be completed on index `db2ext.title`. Note that there need to be at least five changes before the automatic index update can start to synchronize with the database.

See "CREATE INDEX" on page 125 for more details on the parameters.

In a partitioned database environment, the update services only start on one node.

Note

Using UPDATE FREQUENCY every minute brings a high workload on your machine. To avoid this, only use UPDATE FREQUENCY for a very limited number of indexes.

Chapter 6. Creating and maintaining a text index

This chapter provides information on creating and maintaining a text index and covers the following areas:

- Introducing the `db2text` commands
- Enabling a database for text search
- Creating a text index for different data types
- Creating a text index on a nickname with incremental index update using DB2 Replication
- Creating a text index which a stored procedure can use
- Text indexes on views
- Maintaining an index

There is also information on avoiding code page problems which might occur, and performance considerations which you might need to take into account.

Before creating a text index, ensure that you have considered the prerequisites found in Chapter 4, “Planning”, on page 25. Other indexing prerequisites include starting DB2 Net Search Extender Instance Services using the `db2text start` command.

For examples of creating a text index and making text searchable, see Chapter 3, “User scenarios”, on page 19.

Note

You can also create and maintain a text index by using the DB2 Control Center. See Chapter 7, “Using DB2 Control Center”, on page 55.

Introducing the db2text commands

Here is an example of a DB2 Net Search Extender command:

```
db2text ENABLE DATABASE FOR TEXT
```

Note

The `db2text` commands, such as `db2text ENABLE DATABASE FOR TEXT` and `db2text CREATE INDEX`, are also called commands.

© Copyright IBM Corp. 1995, 2003

Tip

For every create and index maintenance command, you can specify the database, user, and password.

```
db2text ... connect TO <database> USER <userID> USING <password>
```

Note that if you leave out the connect options in the `db2text` command, the environment variable `DB2DBDFT` specifies the database.

To display a list of commands, enter the following command:

```
db2text ?
```

To display the syntax of an individual command, enter the following command:

```
db2text ? command
```

For example, to display the syntax of the `CREATE INDEX` command, use the following command:

```
db2text ? CREATE INDEX
```

`db2text` returns 0 if the command has been processed successfully, and 1 if the command has not been processed. Note that if there are document errors but the index still updates, the `db2text` command returns 0.

Note

The system shell interprets special characters such as ?, (,), *, !, and ″. Therefore, if the command contains these characters, use quotation marks or an escape character.

Here is an example of a UNIX command that uses special characters:

```
db2 "SELECT * FROM sample WHERE CONTAINS (DESCRIPTION, '\"enable\"') = 1"
```

Enabling a database

Summary

When	Once for each database that contains columns of text to be searched in.
Command	ENABLE DATABASE FOR TEXT
Authorization	SYSADM

This command prepares the connected database for use by DB2 Net Search Extender.

This command also declares DB2 Net Search Extender search functions and procedures that are described in Chapter 15, "SQL scalar search function and the SQL table-valued function", on page 163.

When you enable a database, you create the following tables and views:

db2ext.dbdefaults
Shows the database default values for index, text, and processing characteristics.

db2ext.textindexformats
Shows the list of supported formats and the model files used.

db2ext.indexconfiguration
Shows the index configuration parameters.

db2ext.textindexes
A catalog view that keeps track of all text indexes.

db2ext.proxyinformation
Shows proxy information for accessing files using a proxy server.

For information on all the views, see Appendix C, "Net Search Extender information catalogs", on page 209.

When a database is enabled, it remains enabled until you disable it.

Note
A fenced user ID that is different from the instance owner ID does not work with partitioned databases.

Disabling a database

Summary

When When you no longer intend to make text searches in this database.

Command
DISABLE DATABASE FOR TEXT

Authorization
DBADM on the database

When DB2 Net Search Extender prepares the database for use, certain administrative changes are made. This section describes functions that help you to reverse this process.

To disable the connected database, use the following command:

```
db2text DISABLE DATABASE FOR TEXT
```

When you disable a database, you also delete the following objects:

- The DB2 Net Search Extender catalog views that were created when the database was enabled.
- The declaration of DB2 Net Search Extender's SQL functions (UDFs).

To disable the database and remove all the text indexes, use the following command:

```
db2text DISABLE DATABASE for text force
```

Note

Disabling a database will fail if there are any text indexes defined in the database. It is recommended to remove these indexes one by one and then check if any problems occur. If you use the disable database for text force command, it only guarantees that Net Search Extender catalog tables in the database are removed.

However, if some of the indexes can not be completely dropped, there may still be resources that need to be manually cleaned up. These include:

- Files in the index, work and cache directory
- Scheduler entries in ctedem.dat
- Where an index was created using the replication capture option, the IBMSNAP_SIGNAL, IBMSNAP_PRUNE_SET, and IBMSNAP_PRUNCNTL entries in the tables of the remote database must be manually deleted. These entries can be easily identified using `APPLY_QUAL="NSE"||<instance name> and TARGET_SERVER= <database name>` command.

  In the following example, the instance is `DB2` and the database is `SAMPLE`.

```
DELETE FROM <ccSchema>.IBMSNAP_SIGNAL
WHERE SIGNAL_INPUT_IN IN
        (SELECT MAP_ID FROM <ccSchema>.IBMSNAP_PRUNCNTL
        WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE');

DELETE FROM <ccSchema>.IBMSNAP_PRUNCNTL
WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE';

DELETE FROM <ccschema>.IBMSNAP_PRUNE_SET
WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE';
```

Creating a text index

Summary

When	Once for each column that contains text to be searched.
Command	CREATE INDEX ... FOR TEXT ... (See the following examples)
Authorization	CONTROL on the table

Creating a text index

You can create a text index on supported data types, although there are different requirements for the following data types:

- Binary data types
- Nonsupported data types
- Datalink data types

There are also different requirements for creating a text index for a stored procedure search.

When you create a text index, you also create the following objects:

A log table
: This keeps track of all changed rows in the user table. Note that if you select the **Recreate index on Update** option or use replication capture tables, the log table is not created.

An event table
: This collects information about problems during an update of the text indexes.

Triggers on the user table
: These add information to the log table whenever a document in the column is added, deleted, or changed. The information is necessary for index synchronization when indexing time next occurs.

  Note that you only create triggers if you create a log table, and the text index is created on a base table and not on views or nickname tables.

To optimize performance and disk space, use the `CREATE INDEX` command to specify a different tablespace for the tables.

> **Note**
> Using the `DB2 LOAD` command to import your documents can cause problems, as triggers do not fire and incremental indexing of the loaded documents is not possible.
>
> Therefore, it is preferable to use the `DB2 IMPORT` command as this activates the triggers.

The following example creates a text index on text column `HTMLFILE` in table `htmltab`.

```
db2text create index DB2EXT.HTMLIDX for text on DB2EXT.HTMLTAB
          (HTMLFILE) format HTML
```

A primary key on this table is necessary.

The default values for index creation are from the `db2ext.dbdefaults` view.

If errors occur during indexing, so-called **index update events** are added to the event table. For example, when a document queued for indexing can not be found. For additional information, see the "Event view" on page 215.

To reverse the changes made by `CREATE INDEX`, use the `DROP INDEX` command. See "Dropping a text index" on page 53 for this information.

To synchronize the text index with the database, use the following command:

```
db2text update index DB2EXT.HTMLIDX for text
```

Note that you can only find documents after synchronization.

Search summary

Depending on the options selected during index creation, different ways of searching are possible:

- The SQL scalar search functions work on all text indexes, except those created on views.
- The stored procedure search function only works on text indexes that are created with a cache.
- The SQL table-valued function works on all text indexes, including those created on views.

Creating a text index on binary data types

When you store data in a column having a binary data type, for example `BLOB`, `FOR BIT DATA`, or a `datalink` value, DB2 does not convert the data. This means that the documents retain their original code pages (CCSIDs), which can cause problems when creating a text index as you might have two different code pages. Therefore, you need to determine whether you are using the code page of the database, or the code page specified in the CREATE INDEX command.

To avoid this problem, specify the code page when creating the text index:

```
db2text CREATE INDEX db2ext.comment FOR TEXT ON db2ext.texttab (comment)
        CCSID 1252
```

If the code page is not specified, check which CCSID has been used to create the index, by calling:

```
db2 SELECT ccsid FROM db2ext.textindexes WHERE INDSCHEMA = 'COMMENT'
                     and INDNAME = 'DB2EXT'
```

Note that there is no support for having documents with different code pages in one text index. For information on how DB2 converts document code page settings, go to the *DB2 Universal Database Administration Guide*. See "Related information" on page ix.

Note that there is no such problem with creating indexes on character data types.

Creating a text index on a nonsupported data type

To create an index, the text columns must be one of the following data types:

- CHAR
- VARCHAR
- LONG VARCHAR
- CLOB
- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC
- DBCLOB
- BLOB
- DATALINK

If the documents are in a column of a different type, such as a user-defined type (UDT), you must provide a function that takes the user type as input and provides as an output type one of the above-mentioned types.

Specify the name of this transformation function. See "CREATE INDEX" on page 125 for further information.

Example: You intend to store compressed text in a table.

1. Create a user-defined type (UDT) for the text in an interactive SQL session:
   ```
   db2 "CREATE DISTINCT TYPE COMPRESSED_TEXT AS CLOB(1M)"
   ```
2. Create a table and insert the text into it:
   ```
   db2 "CREATE TABLE UDTTABLE (author VARCHAR(50) not null,
                               text COMPRESSED_TEXT, primary key (author))"
   db2 "INSERT ..."
   ```
3. Create a user-defined function (UDF) called, for example, `uncompress`. This receives a value of type COMPRESSED_TEXT and returns the corresponding uncompressed text as, for example, a CLOB(10M) value.
4. Create your text index in the following way to specify the `uncompress` UDF:
   ```
   db2text "CREATE INDEX UDTINDEX for text ON UDTTABLE
                                (uncompress(text))
                                ..."
   ```

Creating a text index for DATALINK data types

DB2 Net Search Extender supports the data type DATALINK.

1. If you are using proxies, add one row to the `db2ext.proxyinformation` table, counting the host name, a timeout value in seconds, and either port 'proxy' or 'socks'.

   ```
   db2 INSERT into db2ext.proxyinformation values
             ('hostname', '80' 'proxy', 10)
   ```

   Note that only one row is allowed in this table. A trigger assures this.

2. See the DB2 Universal Database Version 8 documentation for details on how to set up the Java environment on different platforms, for example in the DB2 Information Center. Basically, you must adjust the database manager configuration.

   The Data Link UDF returns blob (100 KB). To change the return size, use `DB2EXTDL` and update the database manager configuration parameter, `java_heap_sz`.

Installing the Data Links jar file

With Net Search Extender, you can index data stored in files that are referenced using the DB2 Data Links feature. For this, you need to install the Data Links jar file `ctedludf.jar`. In the db2 command line processor, run the following command:

- For UNIX:

  ```
  call sqlj.install_jar
       ('file:/<instance_owner_home>/sqllib/java/ctedludf.jar','ctedludf_jar')
  ```

- For Windows:

  ```
  call sqlj.install_jar
       ('file:D:\sqllib\java\ctedludf.jar','ctedludf_jar')
  ```

You also need to update your Java heap size using the following command:

```
db2 update dbm cfg using JAVA_HEAP_SZ 1024
```

Use the following command to unregister the jar file on all platforms:

```
call sqlj.remove_jar('ctedludf_jar')
```

For a list of error messages, see Appendix K, "Data Link messages", on page 273.

Creating a text index on a nickname with incremental index update using DB2 Replication

Before creating a text index on a nickname using a replication capture table, you must perform the following steps:

Note

The following steps only provide an overview of the process and are not an example.

1. Set up the DB2 federated database with all server definitions and wrapper definitions.
2. Set up the replication control tables and the capture programs at the remote server. This is where the source table for the nickname resides. See Chapter 2, "Setting up for Replication" in the *DB2 Replication Guide and Reference, Version 8*. If DB2 does not automatically create nicknames, you must create nicknames in the federated DB2 database using one schema name for the following tables:
   - IBMSNAP_SIGNAL
   - IBMSNAP_PRUNE_SET
   - IBMSNAP_PRUNCNTL
   - IBMSNAP_REGISTER
   - IBMSNAP_REG_SYNC (Non-DB2 remote sources only)

   After this step, nicknames for the replication control tables are available as nicknames under one "capture control schema" on the federated DB2 database. This schema name is important for the DB2TEXT CREATE INDEX command.
3. Register the table as a replication source. For details, see Chapter 3, "Registering tables and views as replication sources" in the *DB2 Replication Guide and Reference, Version 8*. For restrictions on registering the nickname the index is to be created on, see page 132.
4. If DB2 does not automatically create a nickname in the registration step, create a nickname for the replication capture table in the federated database. The replication capture table can either be a Change Data (CD) table or a Consistent Change Data (CCD) table. This nickname is a parameter for the DB2TEXT CREATE INDEX command.

   Note that the column names IBMSNAP_OPERATION, IBMSNAP_COMMITSEQ, IBMSNAP_INTENTSEQ, and the names of the primary key columns must not be changed.
5. If you are using DB2 replication source, ensure that your capture program is running. We strongly recommend not to use a cold start for the capture program. If a cold start is used, all rows in the IBMSNAP_SIGNAL table for APPLY_QUAL LIKE 'NSE%' have to be reinserted. In the following SQL statement you can see how this is done:

```
INSERT INTO <capture control schema>.IBMSNAP_SIGNAL
SELECT CURRENT TIMESTAMP, 'CMD', 'CAPSTART', MAP_ID, 'P'
FROM <capture control schema>.IBMSNAP_PRUNCNTL
WHERE APPLY_QUAL LIKE 'NSE%';
```

6. You can use the following example to create a text index on a nickname using replication:

```
DB2TEXT
CREATE INDEX <indexname> FOR TEXT ON <nickname> (< text column>)
REPLICATION CAPTURE TABLE <capture nickname>
CONTROL TABLE SCHEMA <capture control schema>
```

Creating a text index which the stored procedure search can use

To use the stored procedure search, you need to specify cache options during the CREATE INDEX command. This enables high performance, by moving all the specified data into main memory.

However, before the first index update for searching, ensure that your table contains documents to avoid updating an index on a non-populated table. This provides a better indexing performance and a solid estimation of cache memory requirements.

The stored procedure search allows you to quickly return predefined data that is associated with a document. Use the cache table option to define this in the CREATE INDEX command. The ACTIVATE CACHE command then moves the specified data into a memory cache.

Note

The SQL scalar search functions can also use this text index, if the text index is not created on a view.

In a distributed DB2 environment, you must explicitely specify a tablespace for administration tables on a single node for the stored procedure and explicitely call on this node.

To ensure that you connect to the correct node, use the DB2NODE environment variable.

When creating a text index for stored procedure search, you must determine and calculate the following parameters:

- The type of cache.
- How to update the index.
- The maximum amount of memory that Net Search Extender can use, the MAXIMUM CACHE SIZE.

- The amount of free memory necessary for subsequent document updates, the PCTFREE. Note that this is only for incremental updates.

The following types of cache are available:

A temporary cache
This is rebuilt with each DB2TEXT ACTIVATE CACHE command, and requires loading the data from your DB2 table to memory. This takes longer than activation of a persistent cache, especially for large indexes. However, it might provide slightly better search performance.

A persistent cache
This is maintained on disk and can be quickly mapped to memory by means of the operating system on each DB2TEXT ACTIVATE CACHE command. In incremental index update scenarios, it must remain activated to allow synchronization between the index and the cache. If this does not occur, the next DB2TEXT ACTIVATE CACHE command recreates the cache from scratch.

The following methods of updating a text index are available:

Without the Recreate index on update option
Avoid deleting and re-inserting a document in the table as the slot for a deleted document cannot be reused in the cache. As a consequence, the changing of key columns should be avoided on an activated index.

This is also known as incremental update.

With the Recreate index on update option
This recreates the index on each update. Use variable data types in the cache column expressions wherever possible. This will save cache space. Use the corresponding cast expressions in the CACHE TABLE clause.

Use this option if you expect to insert more than 50% of your documents after the initial index activation.

Net Search Extender provides two SQL functions to help you determine the CREATE INDEX memory parameters. These are: MAXIMUM CACHE SIZE and PCTFREE.

For incremental and recreate updates
The following UDF function returns the recommended MAXIMUM CACHE SIZE value in megabytes (MB):

```
DB2EXT.MAXIMUM_CACHE_SIZE(maximumNumberDocs INTEGER,
       averageRowLength INTEGER, numberOfCacheColumns INTEGER)
```

The following command returns the average row length parameter from your table:

```
SELECT AVG(LENGTH(cache column_1) + ... + LENGTH(cache column_n))
```

Note that the average may change significantly when further documents are inserted into your table. The number of cache columns relates to the number of column expressions used in the CACHE TABLE clause of the DB2TEXT CREATE INDEX command.

For additional information, see Appendix B, "Using large amounts of memory", on page 205.

For incremental updates only

The following UDF function returns the recommended PCTFREE value based on the actual and maximum numbers of documents.

```
DB2EXT.PCTFREE(actualNumberDocs INTEGER, maximumNumberDocs INTEGER)
```

The actual numbers of documents are the number of rows in your table at the time of the first ACTIVATE CACHE command, which creates the memory cache.

The maximum number of documents is an estimate of the maximum number of documents in your table before the next DB2TEXT ACTIVATE command (for a temporary cache), or DB2TEXT ACTIVATE CACHE RECREATE command (for a persistent cache) is run.

If you are recreating the index on each update, set the PCTFREE value to 0.

Examples

Assume that you have 10 000 rows in your table and you do not expect more then 20 000. Use the following call to calculate the PCTFREE value you require:

```
db2  "values DB2EXT.PCTFREE(10000,20000) "
```

Assume that your maximum row size is 20 000 and that you have 2 columns in your cache with an average size of 76. Use the following call to return the size:

```
db2 " values DB2EXT.MAXIMUM_CACHE_SIZE(20000,76,2) "
```

After determining suitable parameters, you can create your index and cache table by using the following call:

```
db2text CREATE INDEX db2ext.comment FOR TEXT ON db2ext.texttab (comment)
          CACHE TABLE (docid) PCTFREE 10 MAXIMUM CACHE SIZE 5
```

Creating a cached text index which the stored procedure search can use

In this example, the docid column is built in addition to the index, using main memory for fast result table return. Ten percent of the cache memory is reserved for future documents and the cache is limited to a maximum of 5 MB.

Updating the text index

To search on this index, you need to update and then activate the index. This copies the specified table cache expression from the database into memory.

If during ACTIVATE or UPDATE operations, the MAXIMUM CACHE SIZE is exceeded, the following actions are recommended:

- Rebuild the cache by using the following sequence of DB2TEXT commands: DEACTIVATE CACHE, ALTER INDEX, MAXIMUM CACHE SIZE, and ACTIVATE CACHE RECREATE.

If you expect frequent updates on documents, consider using fixed-size data types for the cache column expressions in the CACHE TABLE clause. The following example shows how you can use the same cache storage during update operations:

```
CACHE TABLE(cast(C1 as char(20)), cast(substr(C2,1,10) as char(10))....
```

In this case, you ensure that only the non-variable data types are used.

Activating and deactivating the cache for a text index

Before cache activation, perform any pending incremental updates to avoid a poor PCTFREE calculation.

To activate the text index, use the following command:

```
db2text ACTIVATE CACHE FOR INDEX db2ext.comment FOR TEXT
```

This command retrieves the specified cache table data out of the database and stores this in memory. The time taken depends on the size of the table.

Note

If you call update index when an index is activated, this will also update the cache tables. As deleted documents may take slots in cache memory, ensure that you set PCTFREE with a high enough value.

The ACTIVATE CACHE call needs to be redone every time you stop your system. If you use the persistent cache, the new activate will be quicker.

Note

If an update occurs when the persistent cache is not activated, the persistent cache is dropped and recreated during the activate call.

To save resources, you should also deactivate any indexes that are not currently required.

To check how much memory is left, use the following call:

```
db2text control show cache status for database cte index db2ext.comment
```

This displays whether the index has been activated and how much of the specified cache space is left.

Text indexes on views

When using the stored procedure, you are able to create text indexes on views. However, one major drawback is that you cannot create triggers on views, so any changes in the underlying base tables are not recognized.

So with incremental index updates, the user has to know which document has been added, updated, or deleted in order to synchronize the text index with the database. To do this, you must add all the changes to the log table. This process is shown in the following sample:

1. To create the base table, use the following command:

   ```
   db2 "create table sample (key INTEGER not null  PRIMARY KEY, name
             VARCHAR(50) not null, comment VARCHAR(90))"
   ```

2. To add some entries, use the following commands:

   ```
   db2 "insert into sample values(1,'Claus','works in room 301')"
   db2 "insert into sample values(2,'Manja','is in the same office
                                  as Juergen')"
   db2 "insert into sample values(2,'Juergen','has the longest way to
                                  Raiko')"
   db2 "insert into sample values(3,'Raiko','is sitting in the office
                                  besides Claus ')"
   ```

3. To create the view, use the following command:

   ```
   db2 "create view sampleview as select key, comment from sample"
   ```

4. Use the following commands to create, update, and activate the text index:

   ```
   db2text "create index indexview for text on hde.sampleview(comment)
              cache table (comment) maximum cache size 1 key columns
              for index on view (key)"
   db2text "update index indexview for text"
   db2text "activate cache for index indexview for text"
   ```

Note

You need to specify the cache table to be able to create a text index on a view. To create the correct log table, you must specify the key columns for the index on view. If you create an index in this way, you can also use the index with the table-valued function.

When you use the stored procedure search in a distributed DB2 environment, you must explicitly specify a tablespace for administration tables on a single node and explicitly call on this node. To ensure that you connect to the correct node, use the DB2NODE environment variable.

5. To update the table, use the following commands:

```
db2 "insert into sample values(4,'Bernhard','is working in the same floor
           as Manja, but not as Claus')"
db2 "insert into sample values(5,'Guenter','shares the office with Raiko')"
```

6. Then update the log table. To get the name of the log table, use the following command:

```
db2 "select INDSCHEMA,INDNAME,LOGVIEWSCHEMA,LOGVIEWNAME
           from db2ext.textindexes"
```

This is the layout of the log table:

```
sqltype              sqllen  sqlname.data            sqlname.length
-------------------  ------  ----------------------  --------------
496   INTEGER             4  OPERATION                            9
392   TIMESTAMP          26  TIME                                 4
497   INTEGER             4  PK01                                 4
```

To add the entries into the log table, use the following commands:

```
db2 "insert into sample values(0,CURRENT TIMESTAMP,4)"
db2 "insert into sample values(0,CURRENT TIMESTAMP,5)"
```

The first value describes the operation (0 = insert, 1 = update, 2 = delete). The second should always be the CURRENT TIMESTAMP and the last value, the key which has been inserted.

7. Use the following command to update the index again:

```
db2text "update index indexview for text"
```

You can now search with the stored procedure on the new values.

Performance considerations

To enhance performance during indexing, consider the following issues:

- Using a VARCHAR data type to store the text documents instead of LONG VARCHAR or CLOB.
- Using different hard disks to store the text index and the database files.
- Use small primary key columns, such as TIMESTAMP and INTEGER instead of VARCHAR types.
- Ensuring that your system has enough real memory available for all this data. If there is insufficient memory, the operating system uses paging space instead. This decreases the search performance.

  For information on configuring the memory requirements of different platforms, see Appendix B, "Using large amounts of memory", on page 205.
- The update commitcount parameter, used during the automatic or manual updating of the index, slows down the indexing performance during incremental indexing. Note that the parameter is not used during the initial update process.

  If COMMITCOUNT is not set, then the NUMBER_DOCS parameter from the db2ext.textindexes is not updated. Therefore, to view the number of documents during the updating process, use the CONTROL LIST command. See "CONTROL" on page 102 for information.

Note

For the latest performance tips, go to the DB2 Net Search Extender Web site: www.ibm.com/software/data/db2/extenders/netsearch/index.html

Maintaining text indexes

This chapter describes how to maintain text indexes and get useful information about them. The maintenance tasks are:

1. Updating and reorganizing a text index
2. Altering a text index
3. Deleting index update events
4. Dropping a text index
5. Showing index status

You can run these tasks at any time and in any sequence. The chapter also includes information on how to back up and restore indexes and enabled databases.

For commands that display information on text indexes, directory names, and updates, see Appendix C, "Net Search Extender information catalogs", on page 209.

You can also maintain a text index by using the DB2 Control Center. See "Maintaining a text index" on page 71.

Updating and reorganizing a text index

After creating and updating the text index for the first time, you must keep the text index up to date. For example, when you add a text document to a database, or change an existing document in a database, you must index the document to keep the content of the index synchronized with the content of the database. Likewise, when you delete a text document from a database, its terms must be removed from the index.

If the text index was created without the `RECREATE INDEX ON UPDATE` option, triggers automatically store information about new, changed, or deleted documents in an internal log table. So, the next time an index update takes place, the documents referenced in the log table are indexed. For a text index on views, see "Text indexes on views" on page 47 for additional information.

If you specify the `RECREATE` option in the `CREATE INDEX` command, the index is totally rebuilt for each update. This option creates no log table or triggers.

Typically you update an index at intervals. You can change the update frequency for an existing index by using the `ALTER INDEX` command.

You specify the index update frequency in terms of when the update is to be made, and the minimum number of text changes that must be queued. If there are not enough changes in the log table at the day and time given, the index is not updated.

You should plan periodic indexing carefully; to index text documents is a time- and resource-consuming task. The time taken is dependent on many factors. These include the size of the documents, how many text documents have been added or changed since the previous index update, and how powerful the processor is.

Note

On a DB2 table, rollback and deadlock situations might occur in the following cases:

- High update frequencies
- High frequency change transactions
- Long transactions

The `UPDATE INDEX` command lets you update an index immediately on request.

Summary

When When an index must be updated immediately without waiting for periodic indexing to occur.

Command
UPDATE INDEX

Authorization
CONTROL on the table

The following command updates the index:

```
db2text UPDATE INDEX comment FOR TEXT
```

This command is useful when you have added several text documents to a database and want to search them immediately.

To determine if manual reorganization is necessary, query the `db2ext.textindexes` view by using the following command:

```
db2 "select reorg_suggested from db2ext.textindexes where INDNAME = 'comment'"
```

If you specify `MANUAL REORGANIZATION` and often update a column, the update process becomes slower. To manually reorganize, use the following command:

```
dbtext UPDATE INDEX comment FOR TEXT reorganize
```

However, if you specify `AUTOMATIC REORGANIZE` during `CREATE INDEX`, the index will be automatically reorganized when necessary.

Altering a text index

Summary

When When the update frequency or index and work directories have to be changed.

Command
ALTER INDEX

Authorization
CONTROL on the table

Use this command to change the index work directory, the update frequency of an index, or the cache characteristics, principally the `MAXIMUM CACHE SIZE` or `PCTFREE`. If you do not specify an update frequency, the current settings are

left unchanged. If an index update or search is running, an error message displays. This states that the index is currently locked and no changes can be made.

The following example changes the update frequency for the index.

```
db2text ALTER INDEX comment FOR TEXT
         UPDATE FREQUENCY d(1,2,3,4,5) h(12,15) m(00) UPDATE MINIMUM 100
```

In this example, the index is to be updated at 12:00 or 15:00, on Monday to Friday, if a minimum of 100 text documents are in the queue.

Use the following command to stop the periodic updating of an index:

```
db2text ALTER INDEX comment FOR TEXT
                      UPDATE FREQUENCY NONE
```

If the index is copied from one directory to another, the index is locked during this process. However, after copying, the index is unlocked and can be used again.

Clearing index events

Summary

When When you no longer need the messages in an index's event table.

Command
CLEAR EVENTS FOR INDEX

Authorization
CONTROL on the table

Information about indexing events, such as the update start and end times, the number of indexed documents, or document errors that occurred during the update, are stored in the index's event table. This can help you determine the cause of the problem. When you no longer need these messages, you can delete them.

The following example deletes messages from the specified text index:

```
db2text CLEAR EVENTS FOR INDEX comment FOR TEXT
```

Dropping a text index

Summary

When When you no longer intend to make text searches in a text column.

Command
DROP INDEX FOR TEXT

Authorization
CONTROL on the table

Example:

```
db2text DROP INDEX comment FOR TEXT
```

When dropping a text index, you also drop the following tables and views:

- The log table and view
- The event table and view
- The log table triggers (if present)

Note
Always drop the indexes on the table before dropping the table. If you drop the table first, the indexes still exist.

Viewing text index status

To get information about the current text indexes within the database, use the views. For example, if you want to know about the current database defaults, use the following command:

```
db2 "select * from db2ext.dbdefaults"
```

For information about the currently available indexes, their corresponding tables, and the number of indexed documents, use this command:

```
db2 "select indschema, indname, tabschema, tabname, number_docs
     from db2ext.textindexes"
```

Use this command for information about the formats of a specific index:

```
db2 "select format, modelname from db2ext.textindexformats where
     indschema = 'DB2EXT' and indname = 'TITLE'"
```

For further information, see Appendix C, “Net Search Extender information catalogs”, on page 209.

Viewing text index status

If COMMITCOUNT is not set, then the NUMBER_DOCS parameter from the db2ext.textindexes is not updated. To view the number of documents updated during the update process, use the following command:

```
db2text CONTROL LIST ALL LOCKS FOR DATABASE sample INDEX db2ext.title
```

For further information, see the "CONTROL" on page 102.

Backing up and restoring indexes

Use the following steps to back up enabled databases and text indexes created by DB2 Net Search Extender:

1. To find out which indexes DB2 Net Search Extender has created and where they are stored, call a select statement on the db2ext.textindexes view:

   ```
   db2 "select indschema, indname, indexdirectory from db2ext.textindexes"
   ```

2. Ensure that no index update is running, and then stop DB2 Net Search Extender services with the following command:

   ```
   db2text stop
   ```

3. After backing up the database, back up the index directories and subdirectories.
4. Restart DB2 Net Search Extender services with the following command:

   ```
   db2text start
   ```

Use the following steps to restore the enabled databases and text indexes created by DB2 Net Search Extender:

1. Stop DB2 Net Search Extender with the following command:

   ```
   db2text stop
   ```

2. Restore the backup copies of the index directories to the same path as before.
3. Restart DB2 Net Search Extender with the command:

   ```
   db2text start
   ```

Chapter 7. Using DB2 Control Center

Use DB2 Control Center to manage DB2 Net Search Extender administration functions, DB2 instances, databases and database objects such as tables, views, and user groups.

You can invoke the commands on different DB2 Control Center objects, for example:

- Instance objects
- Database objects
- Text index objects

The main elements of DB2 Control Center are the menu bar, tool bar, object tree, and content pane.

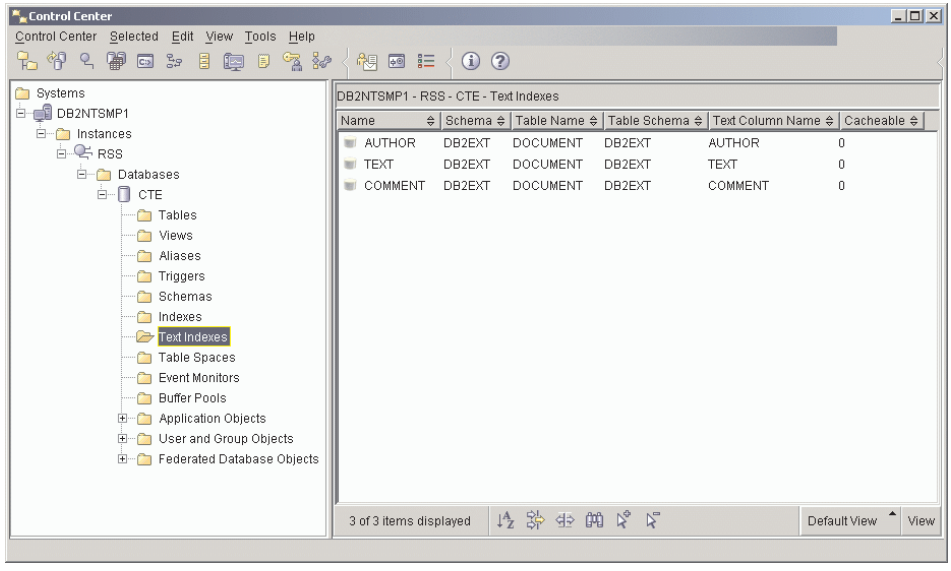


Figure 6. DB2 Control Center

Alternatively, you can use the command line. For more information, see the following chapters:

- Chapter 6, “Creating and maintaining a text index”, on page 33
- Chapter 5, “Net Search Extender instance services”, on page 29

© Copyright IBM Corp. 1995, 2003

Note

To use the examples and the DB2 Control Center for Net Search Extender, a valid Net Search Extender licence must be installed on the database server.

Only DB2 Net Search Extender indexing and administration functions are found in this chapter. For information on using DB2 Control Center, see "Related information" on page ix.

Starting and stopping DB2 Net Search Extender Instance Services

From the object tree, click on a system to display the available instances. Highlight the instance and right-click to display the instance object pop-up menu. Highlight **Net Search Extender** and select one of the following commands from the pop-up menu:

Start DB2 Net Search Extender Instance Services
This starts the instance services if they are not already started.

Stop DB2 Net Search Extender Instance Services

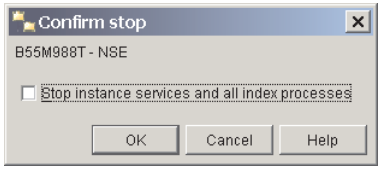


Figure 7. Stop Net Search Extender Services dialog

This displays a dialog. Use the check box to stop the instance services and index processes. In the command syntax, this is known as the `FORCE` option. Click on the **OK** button.

Instance status
This displays a dialog showing the status of the instance.

See Chapter 11, "Administration commands for the instance owner", on page 101 for further information.

Enabling a database

In the object tree, click on the instance object to display the available databases. Highlight the database and right-click to display the pop-up menu. Highlight **Net Search Extender** and select one of the following commands from the extended menu:

Enable the database for text
This displays a dialog if the database is not enabled. Click on the **OK** button to enable the database. If the database is enabled, a message box displays.

Disable the database for text
This displays a dialog if the database is not disabled. Click on the **OK** button to disable the database. If the database is already disabled, a message box displays.

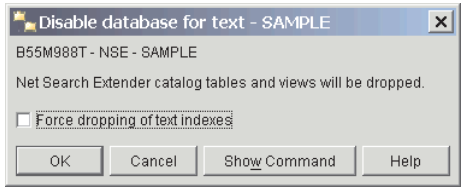


Figure 8. Disable database for text dialog

Click on the check box if you want to disable the database and drop all the text indexes.

Change the Data Link return size
This displays a dialog showing the current Data Link return size. Enter the new Data Link value in kilobytes (KBs) and click on the **OK** button.

See Chapter 12, "Administration commands for the database administrator", on page 107 for further information.

Note that in all the dialogs, the **Show Command** button displays the command line alternative.

Text index administration

In the object tree, below the database object you can see the text index object. Click on the text index object to view the text indexes in the content pane.

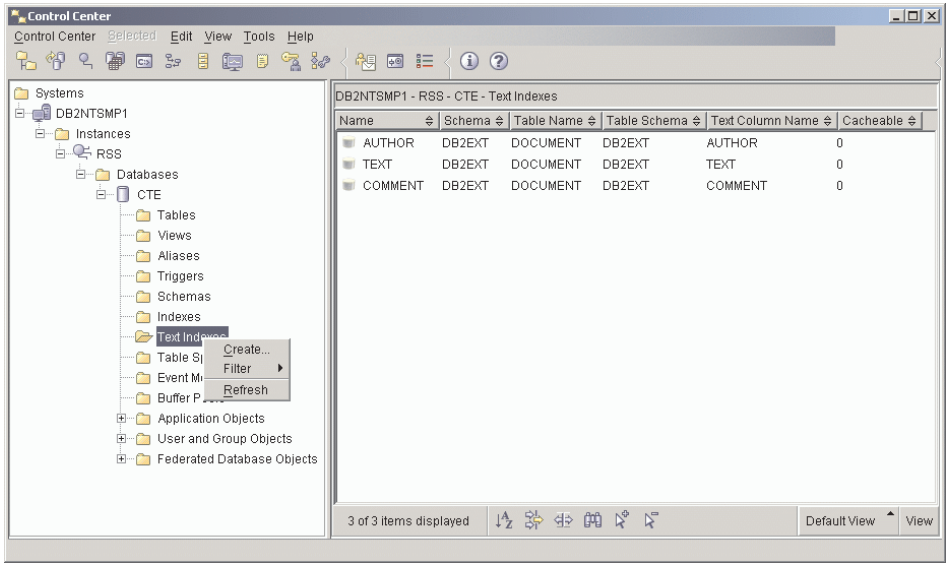


Figure 9. DB2 Control Center

Right-click on the text index object and select one of the following commands from the pop-up menu:

Create This displays a wizard for creating a text index. See "Creating a text index" on page 59 for more information.

Filter This displays a dialog where you can select which text index objects display in the control pane view.

Refresh
This refreshes information in the object tree and control pane.

To maintain text indexes, see "Maintaining a text index" on page 71.

Note

To access the instance, database and text index object commands without using the right-click option, click on the **Selected** menu command and highlight **Net Search Extender** to access the relevant commands.

Before creating a text index, ensure that you have considered the prerequisites found in Chapter 4, "Planning", on page 25.

Other indexing prerequisites include:

- Starting DB2 Net Search Extender Instance services
- Enabling the database

Creating a text index

Select the **Create** command, and a Create Text Index Wizard pops up. Use the wizard to specify the configuration options for the text index in a number of panels.

To move between the panels, enter all the mandatory information and click on the **Next** button until the **Finish** button is enabled. Click on the **Finish** button to create the text index.

To create a text index on views, use the CREATE INDEX command described in "CREATE INDEX" on page 125.

Name panel

This panel allows you to specify the schema and name for the text index. You can also specify a work and index directory for the text index files. Create the administrative tables for the index on the administration tablespace.

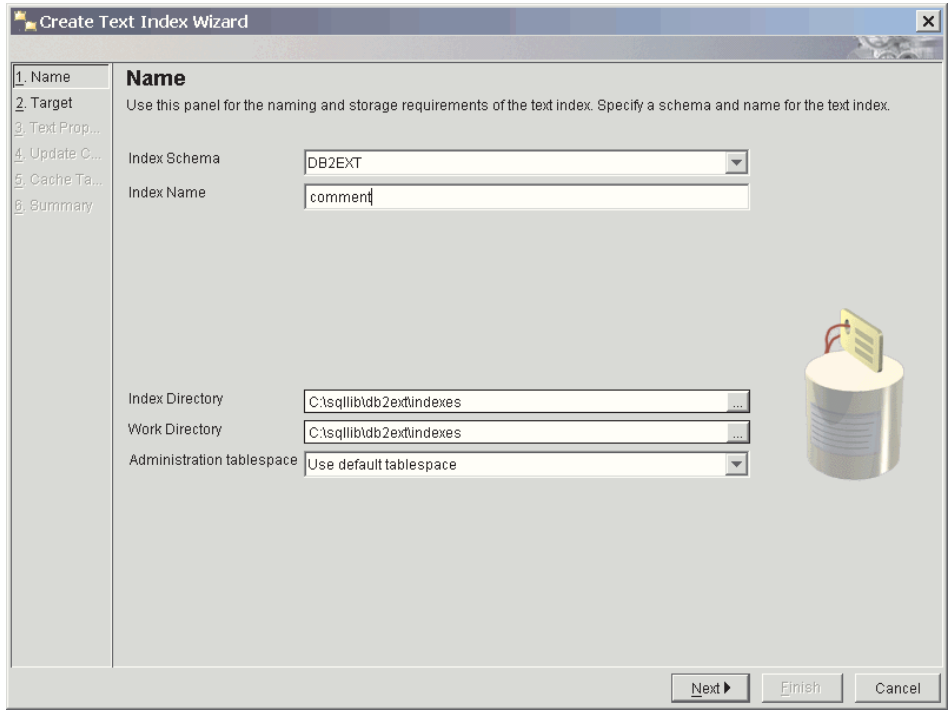


Figure 10. Create Text Index Wizard: Name panel

Here is a description of the fields in the panel:

Table 1. Name panel text fields

Field Name	Mandatory/ Optional	Default	Description
Index schema	Mandatory	user ID	Select a schema name of the text index. This is the DB2 schema name for the index-specific administration tables.
Index name	Mandatory	N/A	Enter a valid DB2 index name for the text index. With the index schema, this uniquely identifies a full-text index in the database.
Index directory	Optional	See the path name	Specify the directory path where you will store the text index. The directory must exist with read, write, and run permissions for the DB2 instance owner user ID.
Work directory	Optional	See the path name	Specify the work directory where you will store temporary files during search and administration operations. The directory must exist with read, write, and run permissions for the DB2 instance owner user ID.
Administration tablespace	Optional	Use the default tablespace	Select a tablespace name for the text index administration tables. You must define the tablespace on the same node group as the tablespace for the user table.

Target panel

This panel allows you to specify the schema and name of the table or nickname table, and the name of the text column containing the data you want to index. You can use a transformation function to modify the content of the text column. In addition to the text column, you can also specify numeric attributes if you want to index content of a table column expression.

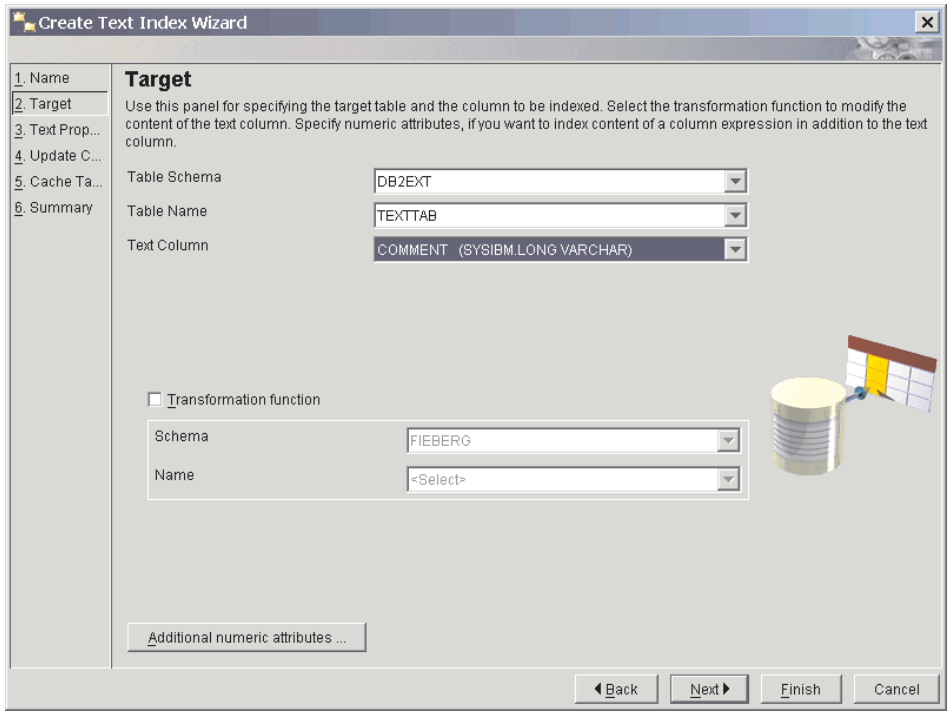


Figure 11. Create Text Index Wizard: Target panel

Here is a description of the fields in the panel:

Table 2. Target panel text fields

Field Name	Mandatory/ Optional	Default	Description
Table schema (1)	Mandatory	user ID	Select the schema of the table or nickname table on which you are creating a text index.
Table name (2)	Mandatory	N/A	Select the name of the table or nickname table on which you are creating an index. The table must have a primary key.
Text column (3)	Mandatory	N/A	Select the name of the column used for creating the text index. The column must be transformed to, or be one of the following types: CHAR (for bit data), VARCHAR (for bit data), LONG VARCHAR (for bit data), CLOB, DBCLOB, BLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, and DATALINK.
Transformation function	Optional	Disabled	Select to use a transformation function.

Table 2. Target panel text fields (continued)

Transformation function: Schema	Mandatory (if function selected)	user ID	Select the schema of the UDF used to access the text documents.
Transformation function: Name	As above	N/A	Select the name of a UDF used to access the text documents.

Note that you can only specify the table schema (1), table name (2) and text column (3) in this order.

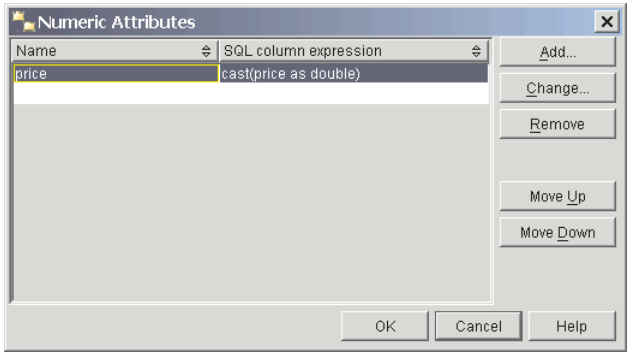


Figure 12. Numeric Attributes dialog

To view or add attributes, click on the **Numeric Attributes** button. A window displays. To add numeric attributes to the index, click on the **Add** button and a further window displays. Specify the SQL column expression and name for the attribute.

Alternatively, select an attribute and press the appropriate buttons to change, move, or remove an entry.

Explaining Numeric Attributes

Use Numeric Attributes to index column expressions in addition to the text column. For example, if you want to index the column date of type `TIMESTAMP` in addition to the text column, specify a numeric attribute `"cast(julian_day(date) as double)"` and specify a name for the attribute.

Specify a numeric attribute if you want to use a numeric expression inside a search query. If you are searching with SQL queries, you can use a combined search instead of using numeric attributes, for example: `WHERE numattrib = 123 AND contains('...')`.

Text Properties panel

This panel allows you to specify the language and format of the text documents. If the documents are not the same CCSID as the database and the text column is of binary type, specify the CCSID. Note that the database CCSID is initially selected. If your documents are of a GPP, HTML, Outside-In, or XML structured format, you can specify a document model.

> **Note**
> In the format list box, the Outside-In filtering format is known as INSO.

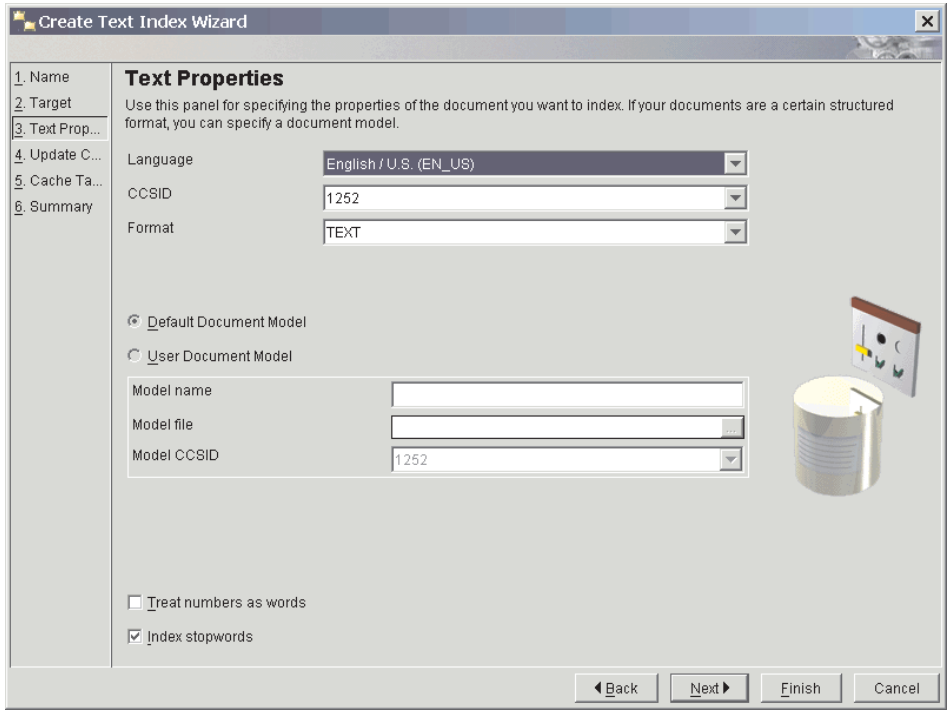


Figure 13. Create Text Index Wizard: Text Properties panel

Here is a description of the fields in the panel:

Table 3. Text Properties panel text fields

Field Name	Mandatory/ Optional	Default	Description
Language	Optional	EN_US	Select a language to determine end-of-sentence and end-of-paragraph delimiters when indexing documents.

Table 3. Text Properties panel text fields (continued)

CCSID	Optional	CCSID of database	Select the CCSID for indexing text documents.
Format	Optional	TEXT	Select the text document format: HTML, XML, TEXT, INSO or GPP.
Default Document Model	Optional	Enabled	Use the default document model.
User Document Model	Optional	Disabled	Use your document model.
Model name	Mandatory (if User Document Model selected)	N/A	Enter the name of the document model. For HTML, XML, Outside-In, and GPP formats, you can specify a document model. Note that the name is only found in the model file.
Model file	As above	N/A	Specify the document model file. The file must be readable by DB2 instance owners.
Model CCSID	As above	Database CCSID	Select the CCSID to interpret the contents of the document model file.
Treat numbers as words	Optional	Disabled	Select to interpret sequences of digits as separate words, even if they are adjacent to characters.
Index stopwords	Optional	Enabled	Select to enable language-specific stopword processing. The `<language>.tsw` in the directory `sqllib/db2ext/resources` contains the stopword list.

Update characteristics panel

This panel allows you to specify whether the index updates incrementally or is recreated from scratch. You can specify update settings so that the index automatically updates at the specified time.

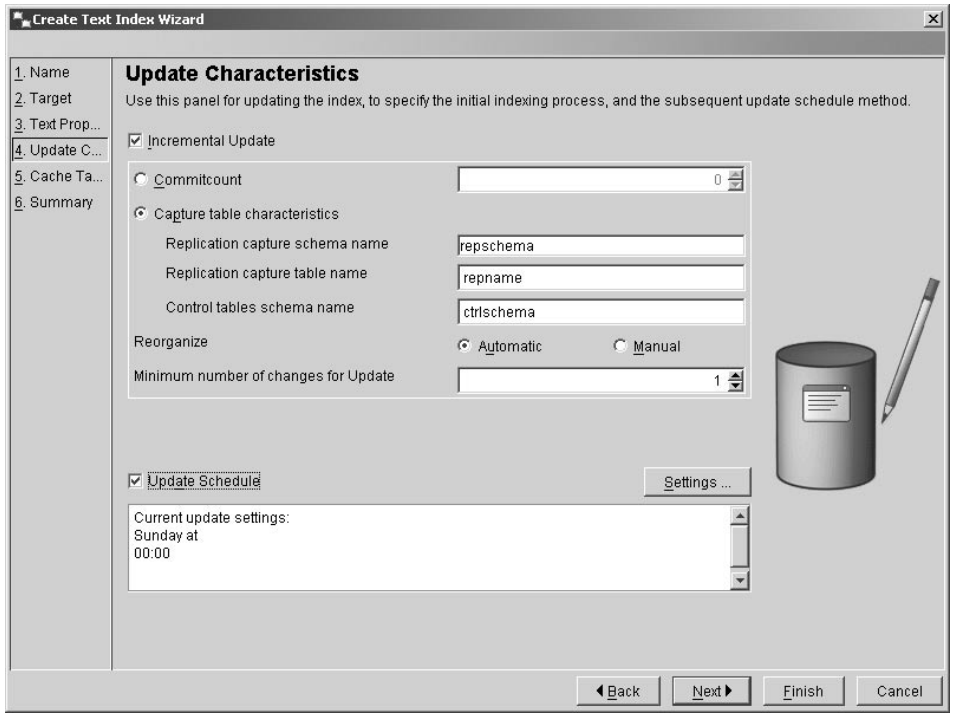


Figure 14. Create Text Index Wizard: Update Characteristics panel

Here is a description of the fields in the panel:

Table 4. Update Characteristics panel text fields

Field Name	Mandatory/ Optional	Default	Description
Incremental update	Optional	Enabled	Select for incremental index updates. If you do not enable the check box, you recreate the index when an update operation is performed.
Commitcount	Optional	0	Number of changes processed during an update in one transaction. Commitcount has implications on performance. For information, see "Performance considerations" on page 48.

Table 4. Update Characteristics panel text fields (continued)

Capture table characteristics	Optional	N/A	Select to use a replication capture table for capturing changes on the source table. The replication capture table must either be a Capture Data (CD) table , or a Capture Change Data (CCD) table and replaces the DB2 Net Search Extender generated log table.
Replication capture schema name	Optional	User ID	The schema name of the replication capture table. Note that the table must have been previously created using DB2 Replication.
Replication capture table name	Mandatory, if Capture table characteristics enabled	N/A	The table name of the replication capture table. Note that the table must have been previously created using DB2 Replication.
Control table schema name	Mandatory, if Capture table characteristics enabled	N/A	The control table schema name. Note that the tables must have been previously created using DB2 Replication.
Reorg automatic or manual radio button	Optional or Mandatory	Enabled/ disabled	Completes index reorganization automatically or manually.
Minimum number of changes for Update	Optional	1	Specify the minimum number of changes to the text documents before the index incrementally updates at the specified time.
Update schedule	Optional	Disabled	Select to add automatic update settings.

To add index update settings, click on the **Settings** button. Note that this button is only enabled if you select **Update Schedule**. In the dialog, select the days, hours, and minutes for the update time. Note that if you select multiple days, the update occurs at the same time on all the selected days.

Cache table panel

This panel allows you to specify a cached table in addition to the index. You can specify the result columns to be cached and you can search the cache using a stored procedure. You can also specify other cache parameters, such as type, maximum size, and the order in which you retrieve the contents of the user table during initial indexing.

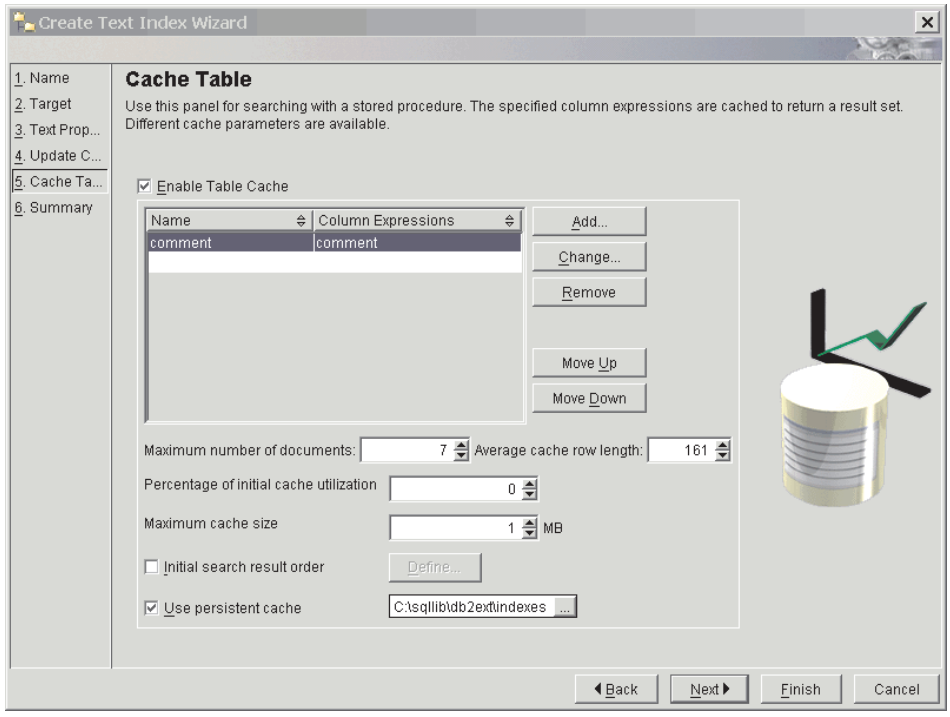


Figure 15. Create Text Index Wizard: Cache Table panel

Here is a description of the fields in the panel:

Table 5. Result Cache panel text fields

Field Name	Mandatory/ Optional	Default	Description
Enable table cache	Optional	Disabled	Select to enable the building of a cached table.
Result column table	Mandatory (if Enable table cache selected)	N/A	Displays a list of SQL column expressions specifying the search result columns.
Maximum number of documents	Mandatory	Row count of table	See the following section: Determining cache utilization and cache size.
Average cache row length	Mandatory	N/A	See the following section: Determining cache utilization and cache size.
Percentage of initial cache utilization	Optional	50%	Select the percentage of the cache held free for additional documents.

Table 5. Result Cache panel text fields (continued)

Maximum cache size	Optional	N/A	Specify a maximum size for the cached table built during index activate. If the number is too small, the activation will fail.
Initial search result order	Optional	Disabled	Select to define the search result order. Documents are returned in the same indexing order as in the cached table. This order can not be ensured after incremental update.
Use persistent cache	Optional	Enabled	This option enables a fast activate execution after a deactivation or system reboot. Note that you must specify a directory path for the persistent cache. Leave disabled if the cache should be temporary.

Determining cache utilization and cache size

The Percentage of initial cache utilization specifies the percentage of the cache to be held free for additional documents. The Maximum cache size specifies the maximum size of the cached table to be built during activate cache. These options depend on the following factors:

- The actual number of documents in the table.
- The expected number of updates.
- The average size of the SQL expressions you want to cache.

You can enter the recommended values for the Percentage of initial cache utilization and Maximum cache size. Alternatively, you can let them be calculated each time you enter values in the Maximum number of documents or Average cache row length fields.

The Maximum number of documents value is initially set to the row count of the table. Modify this according to the number of documents and expected number of changes. Include all document updates, additions, and deletions.

When you add an SQL expression to the cache table list, the Average cache row length is calculated according to the length of the result. As this is based on the number of rows in your table, the calculation can take a considerable amount of time. If you know that on average this value is smaller, modify the value.

For example, if your table has 10 entries and the sum of your column expressions is 100, then these values are initially set. If you expect that the maximum number of documents (including deleted ones) is 10 000, enter this figure. If you know that column expressions on average are smaller than the calculated value, such as a VARCHAR(100) and a filled-in text size of 10, use this figure for the average row size.

To define the initial search result order, click on the **Define** button. Note that this button is only enabled if you select the Initial Search Result Order check box. A dialog displays all the specified SQL column expressions. To add a result order, click on the **Add** button and, in the dialog, specify the SQL result order.

To change, move, or remove an entry, select the expression and click on the appropriate buttons.

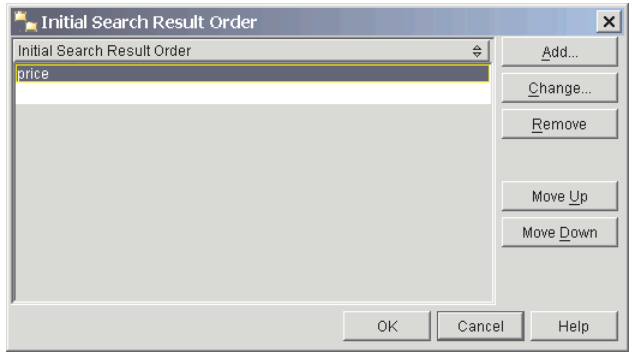


Figure 16. Initial Search Result Order dialog

To add SQL column expressions, click on the **Add** button, next to the Result Column table. In the dialog, specify the result column expression and name.

To change or remove an entry, click on the column expression which enables the appropriate buttons.

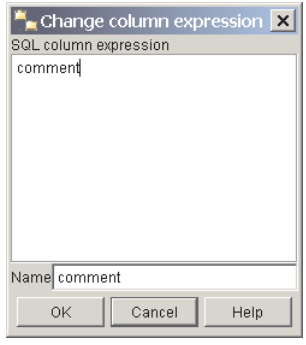


Figure 17. Change column expression dialog

Summary panel

This panel provides an overview of the previously selected parameters.

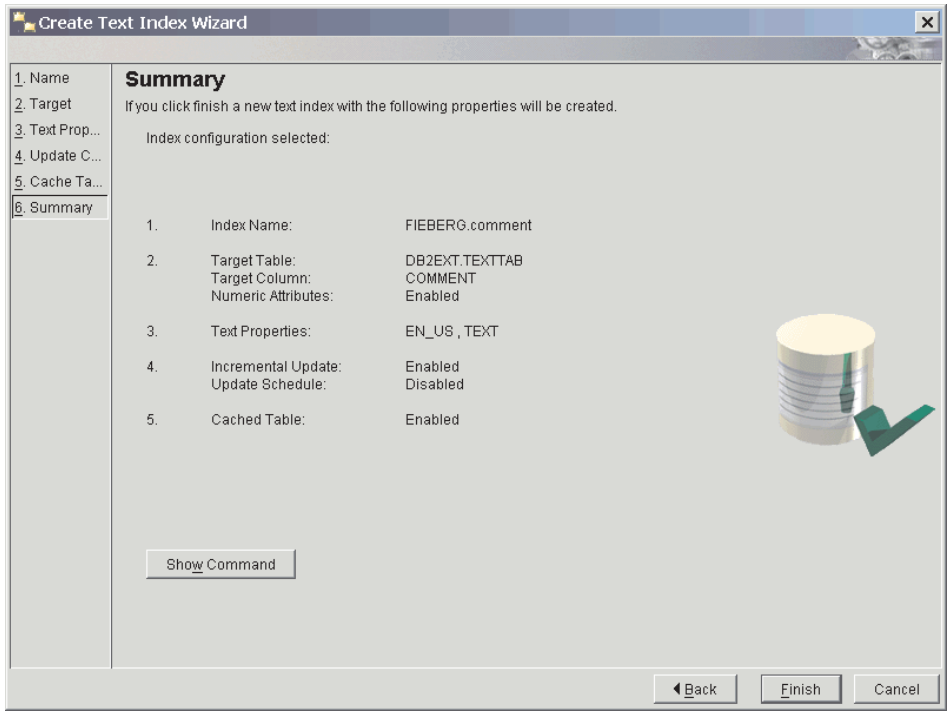


Figure 18. Create Text Index Wizard: Summary panel

Click on the **Show Command** button to view the commands that are run when you click on the **Finish** button. This action creates the text index.

Maintaining a text index

To maintain the text indexes, select the text index in the control pane and click on the **Select** menu command. You can select one of the following commands from the menu:

1. The `ALTER` command, for altering a text index.
2. The `DROP` command, for dropping a text index.
3. The `UPDATE` command, for updating a text index.
4. The `SHOW INDEX EVENTS` command, for showing index events.
5. The `ACTIVATE INDEX MEMORY` command, for activating an index cache.
6. The `DEACTIVATE INDEX MEMORY` command, for deactivating an index cache.
7. The `SHOW STATUS` command, for showing the index status.

Note that the Activating and Deactivating commands only display if you create the index with a cache option.

Altering a text index

Select the **Alter** command and a dialog displays a series of panels. These provide an overview of the text index parameters. Note that you can **not** change all of the parameters.

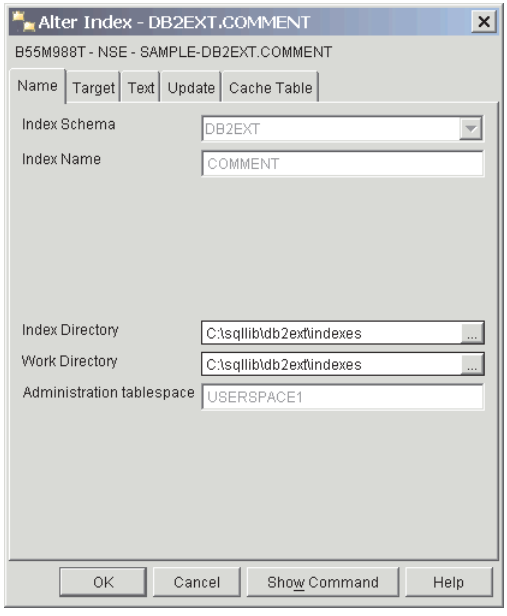


Figure 19. Alter Index dialog: Name tab

The **Name** panel displays the name and storage configurations for the index. You can change the index and work directories.

The **Target** panel displays the target and numeric attribute settings of the index. You cannot change these configurations.

The **Text** panel displays the text document configurations. You cannot change these settings.

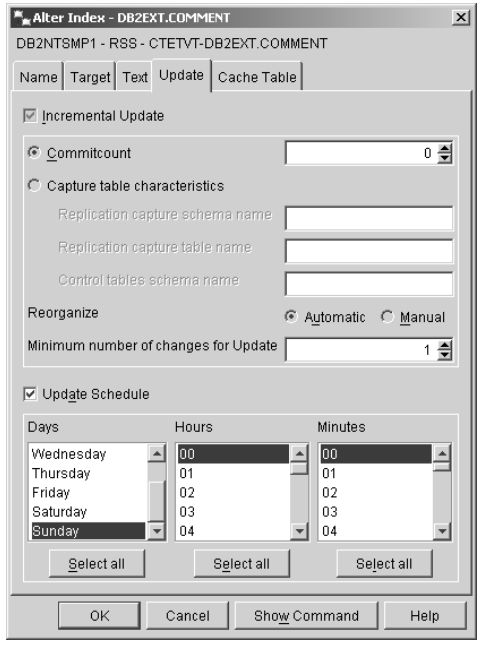


Figure 20. Alter Index dialog: Update tab

The **Update** panel displays the update characteristics of the index configuration. You can alter the update schedule. If the index was created for incremental updates, you can also modify the minimum number of changes. If the index was created with the commitcount option, you can also modify the commitcount value.

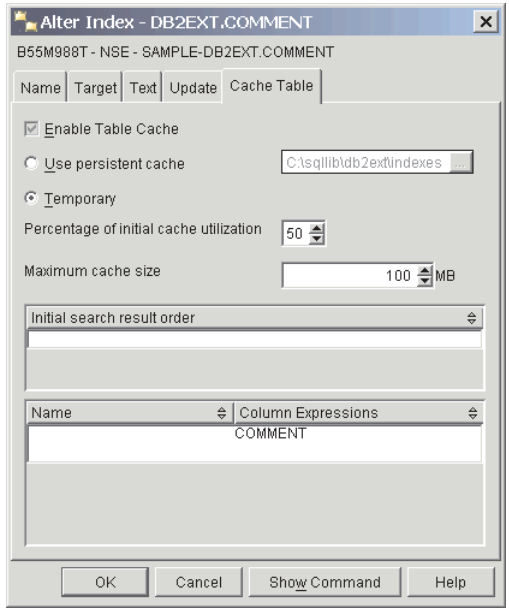


Figure 21. Alter Index dialog: Cache Table tab

The **Cache Table** panel displays the cache option settings. If the result cache is already enabled, you can modify the persistent directory, or make the index cache temporary. You might also change the Maximum cache size and the Percentage of initial cache utilization.

Dropping a text index

Select the **Drop** command and a dialog displays the available text indexes.

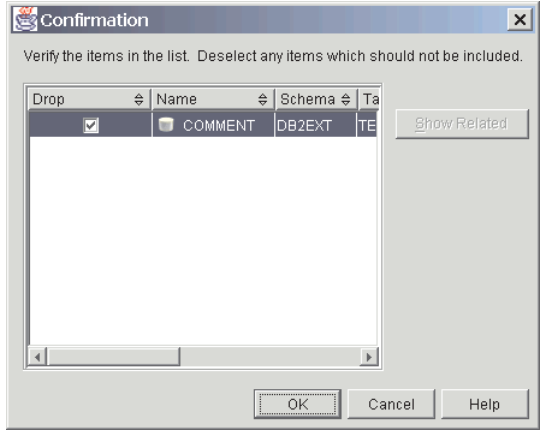


Figure 22. Drop Index dialog

Select the index and click on the **OK** button.

Updating a text index

Select the **Update** command and a dialog displays a number of update options.

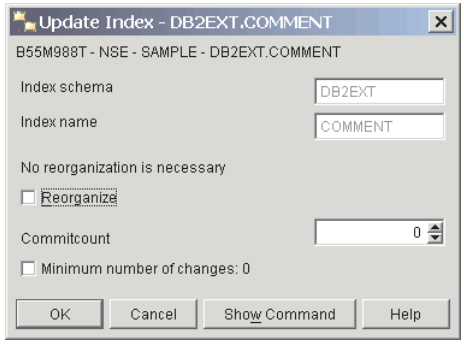


Figure 23. Update Index dialog

You can specify a commitcount for the update operation. If you want to update the minimum specified during create or alter index, select the Minimum number of changes check box. To reorganize the index, select the check box. Note that if reorganization is suggested, the check box is enabled.

Showing Index events

Select the **Show index events** command and the contents of the event table display in the dialog.

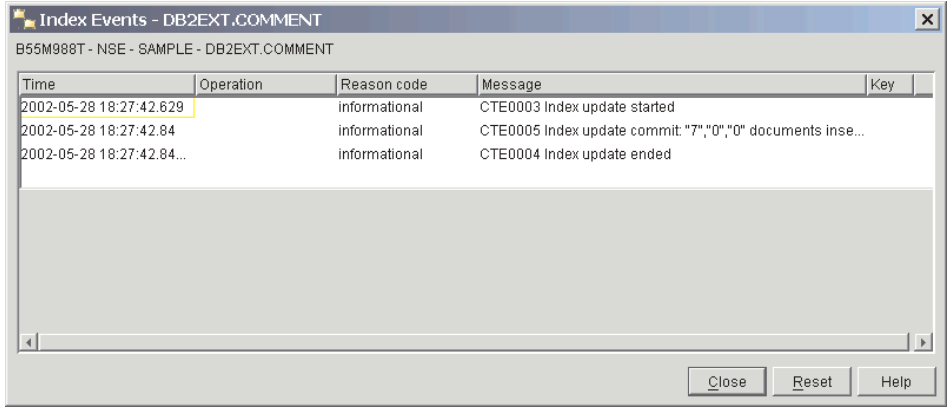


Figure 24. Index Events dialog

To clear the index events, click on the **Reset** button.

Activating a text index cache

Select the **Activate index memory** command and a dialog displays.

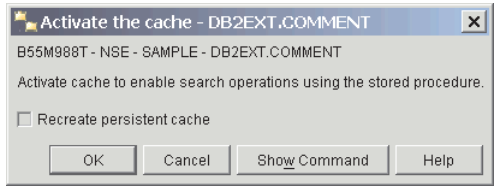


Figure 25. Activate the cache dialog

To activate the cache, click on the **OK** button. If you want to build the cache from scratch, select the check box.

Deactivating a text index cache

Select the **Deactivate index memory** command and a dialog displays.

Figure 26. Deactivate the cached table dialog

To release the cache, click on the **OK** button.

Showing Index status

Select the **Show status** command and a dialog displays the status of the text index.

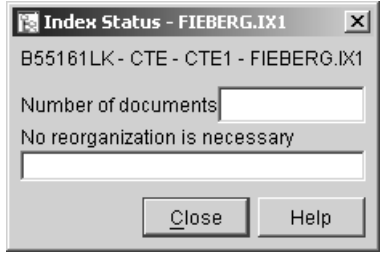


Figure 27. Index Status dialog

This includes information on the number of indexed documents, the reorganization suggested flag, and additional index information.

Chapter 8. Searching

DB2 Net Search Extender provides the following methods for searching text:

SQL scalar search functions
These enable you to include text search subqueries in SQL queries. Net Search Extender provides these functions in addition to those normally available in SQL.

A stored procedure search function
This enables you to return predefined cached result tables.

An SQL Table-Valued Function
You can use this search in a similar way to the stored procedure.

For SQL scalar search functions, the chapter describes the following areas:

- Searching for text, using the CONTAINS, NUMBEROFMATCHES, and SCORE functions.

  Refer to Chapter 15, "SQL scalar search function and the SQL table-valued function", on page 163 for a description of the syntax.
- Specifying search arguments by using examples with the CONTAINS function.

  Refer to Chapter 14, "Syntax of search arguments", on page 153 for a description of the syntax.

For the stored procedure search function, the chapter describes the following areas:

- Searching for text using the stored procedure search.
- For specifying search arguments, refer to Chapter 14, "Syntax of search arguments", on page 153 for a description of the parameters.

For the SQL Table-Valued Function, the chapter describes the following areas:

- Searching for text using the SQL Table-Valued Function and the HIGHLIGHT function.

  Refer to Chapter 15, "SQL scalar search function and the SQL table-valued function", on page 163 for a description of the syntax.
- For specifying search arguments, refer to Chapter 14, "Syntax of search arguments", on page 153 for a description of the parameters.

There is also information on search performance considerations that you may need to take into account.

© Copyright IBM Corp. 1995, 2003

Searching

Before searching, ensure that all the appropriate indexing steps, described in Chapter 6, “Creating and maintaining a text index”, on page 33, involving the different data types are considered.

Note

The system shell interprets special characters such as ?, (,), *, !, and ″. Therefore, if the command contains these characters, you must use quotation marks or an escape character.

Here is an example of a UNIX command that uses special characters:

```
db2 "SELECT * from sample WHERE CONTAINS (DESCRIPTION, '\"enable\"') = 1"
```

Searching for text using SQL scalar search functions

Using examples, this section describes how to use the SQL scalar search functions in the following ways:

- Using the function `CONTAINS` to make a query.
- Using the function `NUMBEROFMATCHES` to determine how many matches are found in a text document.
- Using the function `SCORE` to get the relevancy of a found text document.

Refer to Chapter 15, “SQL scalar search function and the SQL table-valued function”, on page 163 for a description of the syntax.

Making a query

This example demonstrates how the `CONTAINS` function searches for text in column `comment` in table `texttab`. It returns 1 if the text satisfies the search argument, otherwise it returns 0.

```
SELECT AUTHOR,TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT, '"book"') = 1
```

In this example, you search for the term `book` in the column `COMMENT`.

Note

To increase performance, it is beneficial to add restrictive search criteria, for example:

```
SELECT AUTHOR,TITLE
        FROM db2ext.texttab
        WHERE CONTAINS(COMMENT, '"book"') = 1 AND PRICE < 20
```

Searching and returning the number of matches found

Use the NUMBEROFMATCHES function to determine how often the search value is found in each text document.

```
SELECT AUTHOR,TITLE,
       NUMBEROFMATCHES(COMMENT, '"book"')
       FROM DB2EXT.TEXTTAB
```

NUMBEROFMATCHES returns an integer value.

Searching and returning the score of a found text document

SCORE is an absolute value that indicates how well the document meets the search value relative to other found documents. The value indicates the number of matches that are found in the document in relation to the document's size. In the following example, you can get the score of a found document by using the SCORE function:

```
WITH TEMPTABLE(docid,score)
       AS (SELECT docid,
                  SCORE(COMMENT,'"book"')
       FROM DB2EXT.TEXTTAB)
SELECT *
       FROM TEMPTABLE
       WHERE score > 0
       ORDER BY score ASC
```

SCORE returns a DOUBLE value between 0 and 1.

Note

You cannot use the CONTAINS, SCORE, and NUMBEROFMATCHES search functions for indexes created on views.

In a distributed DB2 environment, the SCORE values are different:

- In a non-distributed environment, all the documents are in a single table. The SCORE value is based on a single table, and the documents relationship to all the other documents in the table.
- In a distributed DB2 environment, all the documents are located on different nodes. During indexing, only the local documents are used to build the text indexes, which are local on every node. In this case, the SCORE value is based on the documents relationship to all documents in only one of the multiple nodes.

Specifying SQL search arguments

The CONTAINS, NUMBEROFMATCHES, and SCORE functions all use search arguments. This section uses the CONTAINS function to show different examples of search arguments in DB2 Net Search Extender functions.

Refer to Chapter 14, “Syntax of search arguments”, on page 153 for a description of the syntax.

Searching for terms in any sequence

You can have more than one term in a search argument. One way to combine several search terms is to connect them together using commas, like this:

```
SELECT AUTHOR,TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
          '("kid", "dinosaur")') = 1
```

This form of search argument finds text that contains any of the search terms. In logical terms, an OR operator connects the search terms.

Searching with the Boolean operators AND and OR

You can combine search terms with other search terms using the Boolean operators “&” (AND) and “|” (OR):

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
                              '"author" | "pulitzer"') = 1
```

You can also combine several terms by using Boolean operators:

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
                       '"author" | "pulitzer" & "book"') = 1
```

If you use more than one Boolean operator, these are evaluated from left to right. However, the logical AND operator (&) binds stronger than the logical OR operator (|). You can see this evaluation in the following example, which does not include parentheses:

```
"book" & "pulitzer"| year" & "author"
```

Therefore, Net Search Extender evaluates the boolean operators in the following way:

```
("book" & "pulitzer") | (year & "author")
```

So, to correctly evaluate the boolean operators, you must include parentheses:

```
"book" & ("pulitzer" | year") & "author"
```

You can also combine Boolean operators with search terms that are chained together using the comma separator:

```
("author", "pulitzer") & "book"
```

In this case, however, the comma is interpreted as a Boolean OR operator:

```
("author"| "pulitzer") & "book"
```

For additional information, also see "Searching with the Boolean operator NOT" on page 83.

Fuzzy search

"Fuzzy" search searches for words that are spelled in a similar way to the search term.

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
                    'fuzzy form of 80 "pullitzer"') =1
```

In this example, the search could find an occurrence of the misspelled word `pulitzer`.

The match level, in the example "80", specifies the degree of accuracy. Use a fuzzy search when misspellings are possible in the document. This is often the case when a Optical Character Recognition device, or phonetic input creates the document. Use 1 to 100, where 100 is an exact match and below 80 is "fuzziness".

Note

If the fuzzy search does not provide the appropriate degree of accuracy, search for parts of a term using character masking.

Searching for parts of a term (character masking)

Masking characters, otherwise known as "wildcard" characters, offer a way to make a search more flexible. They do this by increasing the number of text documents that are found by a search.

DB2 Net Search Extender uses two masking characters: percent (%) and underscore (_).

- % represents **any number of arbitrary characters**. Here is an example of % used as a masking character in the middle of a search term:

  ```
  SELECT AUTHOR,TITLE
          FROM DB2EXT.TEXTTAB
          WHERE CONTAINS(COMMENT, '"thr%er"') = 1
  ```

  This search term finds text documents containing the word "thriller".

- _ represents **one character** in a search term. The following example also finds text documents containing the word "thriller".

  ```
  SELECT AUTHOR, TITLE
          FROM DB2EXT.TEXTTAB
          WHERE CONTAINS(COMMENT, '"th_iller"') = 1
  ```

Note

Use wildcard characters sparingly as they can increase the size of your result list significantly, thus decreasing performance and returning unexpected search results.

Searching for terms that already contain a masking character

If you want to search for a term that contains the "%" character or the "_" character, you must precede the character with a so-called *escape* character. Then you can identify the escape character using the ESCAPE keyword.

In the following example, the escape character is a "!":

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
                    '"100!%" ESCAPE "!"') = 1
```

Searching for terms in a fixed sequence

If you search for "primary key", you will only find the two terms if they are adjacent and occur in the sequence shown:

```
SELECT AUTHOR,TITLE
     FROM DB2EXT.TEXTTAB
     WHERE CONTAINS(COMMENT, '"primary key"') =1
```

Searching for terms in the same sentence or paragraph

Here is an example of a search argument that finds text documents in which the search terms occur in the same sentence:

```
SELECT AUTHOR,TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
           '"web" IN SAME SENTENCE AS "disk"') = 1
```

You can also search for more than two words occurring together. In the next example, a search is for two words occurring in the same paragraph:

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
           '"computer" IN SAME PARAGRAPH AS "web"') = 1
```

Searching for terms in sections of structured documents

Here is an example of a search argument that finds text documents where search term IBM occurs in the subsection H2 of structured documents.

```
SELECT CATEGORY, DATE
        FROM DB2EXT.HTMLTAB
        WHERE CONTAINS(HTMLFILE,
                          'SECTIONS ("H2") "IBM"') = 1
```

Note that section names are case-sensitive. Ensure that the section name in the model file and in the query are identical. See Chapter 9, “Working with structured documents”, on page 91 for more information.

Searching with the Boolean operator NOT

You can use the Boolean operator NOT to exclude particular text documents from the search:

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
                              '("author", "pulitzer") & NOT "book"') = 1
```

This example excludes any text documents containing the term “book” from the search for “author” or “pulitzer”.

Thesaurus search

Thesaurus search is a powerful search-term expansion function in DB2 Net Search Extender. The additional terms you search for are taken from a thesaurus that you build yourself, so you have direct control over the terms. For example, a search for “database”, and could find terms like “repository” and “DB2”.

Use this type of search for specific areas of interest in which you make frequent searches and produce significantly more effective search results.

See Chapter 10, “Using a thesaurus to expand search terms”, on page 93 for more information and a description of how to build a thesaurus. The following example demonstrates the syntax for using thesaurus expansion.

This example takes the term “product” and expands it, adding all relations of this term found in the thesaurus “nsesamplethes”. Here, “marketing” is added to the search.

```
SELECT CATEGORY, DATE
        FROM DB2EXT.HTMLTAB
        WHERE CONTAINS(HTMLFILE,
           'THESAURUS "nsesamplethes"
           EXPAND RELATED
           TERM OF "product"') = 1
```

The next example takes the search term “product”. The search then expands with all the *synonyms* of the search term.

```
SELECT CATEGORY, DATE
        FROM DB2EXT.HTMLTAB
        WHERE CONTAINS(HTMLFILE,
           'THESAURUS "nsesamplethes"
           EXPAND SYNONYM
           TERM OF "product"') = 1
```

Specifying search arguments

Numeric attribute search

On numeric attributes that are stored in a structured document, you can search using the following syntax:

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
        'ATTRIBUTE "PRICE" between 9 and 20') = 1
```

Note that attribute names are case-sensitive. Ensure that the attribute name in the model file and in the query are identical. See Chapter 9, “Working with structured documents”, on page 91 for more information.

Free-text search

“Free-text search” is a search in which you express the search term as free-form text. A phrase or a sentence describes in natural language the subject to be searched for. The sequence of words in a free-text query is not relevant. However, for a set of query terms, at least one of the terms must occur in the documents to be searched.

Note that there is no support for the masking of characters or words for search strings in a free-text argument.

For example:

```
SELECT AUTHOR, TITLE, SCORE(COMMENT,
        'IS ABOUT EN_US "something related to dinosaur"')
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
        'IS ABOUT EN_US "something related to dinosaur"') = 1
```

By combining the query with SCORE search function, the search will return documents containing only the word ″something″ as well.

Additional search syntax examples

To become familiar with additional search syntax examples, use the command line processor input file called `search`. This contains examples of DB2 Net Search Extender search functions that run against the sample table.

To run the example, use the following syntax:

```
db2 -tvf search
```

If the table and indexes have not been created, run one of the following:

- On UNIX platforms: `nsesample` in the `<instance_owner_home>/<sqllib>/samples/db2ext` directory.
- On Windows platforms: `nsesample (.bat)` in the `<sqllib>/samples/db2ext` directory.

Searching for text using a stored procedure search

Use the stored procedure search interface when your application needs a subset of the text search result, but in a high performance way. Do not use the stored procedure if you require all the results, or you need to index a large number of documents. The main reason being that as parts of the user table are copied into memory, a lot of real memory needs to be available.

You can use the stored procedure to first request results from 0 to 20, then 21 to 40, and so on, in a similar way to cursor navigation. Combining this cursor capability with the use of a cache (calculated during indexing), searching is extremely fast, especially as no join is necessary.

If you are going to use the stored procedure, ensure that you consider the following options:

- In a distributed DB2 environment, you must explicitly specify a tablespace on a single node for the stored procedure and explicitly call on this node.
- The cache-search-result options have been specified during `CREATE INDEX`.
- The present and future shared memory requirements, possibly involving incremental updates, have been fully considered. See "Creating a text index which the stored procedure search can use" on page 43 for further information.
- The cache of the index has been activated using `db2text activate` command.

The following is an example of a stored procedure search:

```
db2 "call  db2ext.textSearch('\"book\"','DB2EXT','COMMENT',0,2,1,1,?,?)"
```

The first parameter is the search term. The syntax is exactly the same as in the SQL functions. Then specify the index name and index schema. If you have not masked the name, it is translated to uppercase. The following two numbers give you the probability of getting the result in slices. The next two integer values specify if score and hit information are requested. The last two values are output values.

See Chapter 16, "Stored procedure search function", on page 175 for further details on the parameters.

Note

If you request larger result sets, you need a user tablespace. If there is none available, create a tablespace. The following example creates a tablespace on a UNIX platform:

```
db2 "create user temporary tablespace tempts managed by system
            using ('/work/tempts.ts')"
```

In a distributed DB2 environment, you must explicitly specify a tablespace for administration tables on a single node for the stored procedure and explicitly call on this node.

Searching for text using an SQL Table-Valued Function

Use the SQL Table-Valued Function where you implement an interface, but do not need all results back, or do not have all the real memory required to use the stored procedure interface.

There are two SQL table-valued functions available, both called `db2ext.textsearch`. One has additional parameters for use with the `db2ext.highlight` function. See "Using the highlight function" on page 87 for further information.

The SQL Table-Valued Function gives you the same cursor interface as the stored procedure to get only parts of the result. However, you still need to join the results with the user table. You can see this in following example:

```
db2 "select docid , author, score from TABLE(db2ext.textsearch('\"book\" ',
     'DB2EXT','COMMENT',3,2,cast(NULL as integer))) as t, db2ext.texttab u
      where u.docid = t.primkey"
```

The following are the values you could return from the SQL Table-Valued Function:

```
--> primKey <single primary key type>
the primary key

-->  score           DOUBLE
the score value of the found document

--> NbResults       INTEGER
the total number of found results (same value for all rows)

--> numberOfMatches  INTEGER
the number of hits in the document
```

Note

Note that only a single primary key column is allowed. See the "DB2EXT.TEXTSEARCH" on page 167 for further details on the parameters.

Using the highlight function

There are two SQL table-valued functions available, both called `db2ext.textsearch`. To use the `db2ext.highlight` function, you must use the `db2ext.textsearch` function with the additional `numberOfHits` and `hitInformation` parameters.

In this example, call the `db2ext.highlight` function to display the whole document without highlighting any hits found by the `db2ext.textsearch` function.

```
select p.docid,
     db2ext.highlight(p.comment, t.hitinformation, 'WINDOW_NUMBER = 0')
          as highlight
from DB2EXT.TEXTTAB p,
     table (db2ext.textsearch('"bestseller" | "peacekeeping" | "soldiers"
          | "attention"', 'DB2EXT', 'COMMENT', 0, 20,
          cast(NULL as INTEGER), 10)) t
where p.docid = t.primkey and p.docid = 2
```

The search argument returns the following result:

```
DOCID HIGHLIGHT

2     A New York Times bestseller about peacekeeping soldiers called
      "Keepers" who devise a shocking scheme to get the worlds
      attention after their tour of duty ends.

1 record(s) selected.
```

Note

In all the `db2ext.highlight` examples, the table function `db2ext.textsearch` searches for any of the following words: ″bestseller″, ″peacekeeping″, ″soldiers″, or ″attention″.

In this example, call the `db2ext.highlight` function to display the whole document and highlight all hits found by the `db2ext.textsearch` function.

```
select p.docid,
     db2ext.highlight(p.comment, t.hitinformation, 'WINDOW_NUMBER = 0,
          TAGS = ("<bf>", "</bf>" ) ') as highlight
from DB2EXT.TEXTTAB p,
     table (db2ext.textsearch('"bestseller" | "peacekeeping" | "soldiers"
```

```
          | "attention"', 'DB2EXT', 'COMMENT', 0, 20,
          cast(NULL as INTEGER), 10)) t
where p.docid = t.primkey and p.docid = 2
```

The search argument returns the following result:

```
DOCID HIGHLIGHT

2     A New York Times <bf>bestseller</bf> about <bf>peacekeeping</bf>
     <bf>soldiers</bf> called "Keepers" who devise a shocking scheme to
      get the worlds <bf>attention</bf> after their tour of duty ends.

1 record(s) selected.
```

In this example, call the db2ext.highlight function to display at maximum 10 parts (windows) of the document. Each window size is 24, which is approximately 12 bytes of data on each side of the hit. In addition, hits found by the table function db2.textsearch are highlighted.

```
select p.docid,
     db2ext.highlight(p.comment, t.hitinformation, 'WINDOW_NUMBER = 10,
          WINDOW_SIZE = 24, TAGS = ("<bf>", "</bf>" ) ') as highlight
from DB2EXT.TEXTTAB p,
     table (db2ext.textsearch('"bestseller" | "peacekeeping" | "soldiers"
          | "attention"', 'DB2EXT', 'COMMENT', 0, 20,
          cast(NULL as INTEGER), 10)) t
where p.docid = t.primkey and p.docid = 2
```

The search argument returns the following result:

```
DOCID HIGHLIGHT

2     York Times <bf>bestseller</bf> about <bf>peacekeeping</bf> ...
      <bf>peacekeeping</bf> <bf>soldiers</bf> called "Keepers" ... the
      worlds <bf>attention</bf> after their

1 record(s) selected.
```

The first hit found is <bf>bestseller</bf> and this hit determines the first window. The second hit, <bf>peacekeeping</bf> is only 8 bytes away from the first hit and is completely taken into the first window. The third hit, <bf>soldiers</bf> is outside the first window and determines a new window. As the second hit <bf>peacekeeping</bf> is only 2 bytes away from the left side of the <bf>soldiers</bf> hit, it is also taken into the second window and highlighted. The fourth hit <bf>attention</bf> is outside the second window and so determines a new window. As no previous or additional hit is contained in the size of this window, only data surrounding the hit is contained in the window.

Additionally, as no WINDOW_SEPARATOR is specified, the default window separator, " ... " is taken to separate the three document windows.

Note

To ensure high performance when using the `db2ext.highlight` function, the user should limit the search results in the `db2ext.textsearch` table-valued function.

See the "DB2EXT.HIGHLIGHT" on page 171 for further details on the parameters.

Searching on more then one column

In cases where you need to create an text index on more then one column, the easiest way is to use the SQL scalar function and combine the searches on those columns. You can see this in the following example:

```
SELECT AUTHOR,TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
        '"book"')=1 and CONTAINS(AUTHOR,'"Mike"')=1
```

For a table-valued function it is more difficult, as you may need to use a union for performance reasons. Another possibility with the table-valued function is to use a view and combine your table columns in a view column to create a text index. In this way, you avoid having two separate text search calls.

Combining your text columns may provide an improvement in performance. However, this strongly depends on your individual search requirements.

Performance considerations

To enhance performance during search, consider the following issues:

- When searching within SQL:
  - If you notice a decrease in performance, Use the `explain` statement to check the processing plan of the DB2 Optimizer.
  - Parametric search can make searching faster, especially if you use other search predicates to reduce the result size.
  - Use the result limit keyword if you do not require all of the results.
- When searching with the stored procedure:
  - As the specified cache table expression is copied from the database into memory, ensure that your workstation has enough memory available for this data. If there is insufficient memory, paging space is used, which decreases search performance.

Note

For the latest performance tips, go to the DB2 Net Search Extender Web site: www.ibm.com/software/data/db2/extenders/netsearch/index.html

Chapter 9. Working with structured documents

DB2 Net Search Extender allows you to index and search text or numeric fields, such as title, author, or description in a structured document. The documents can be in XML, Outside-In, HTML format, or contain user-defined tags (GPP).

Use markup tags and their field names in a *document model* to define which fields in the documents are indexed and, therefore, are available for searching. You can use the name of the field (also known as the section name) in queries against that field.

To be able to search in these fields, you must specify a `FORMAT AND MODEL FILE` when you create the text index containing the documents. See "CREATE INDEX" on page 125 for further information.

For more information on creating and defining document models, see Chapter 17, "Structured document support", on page 181.

For the document model syntax, see Appendix G, "Document model reference", on page 253.

© Copyright IBM Corp. 1995, 2003

Chapter 10. Using a thesaurus to expand search terms

You can broaden a query by searching not only for a specific search term, but also for terms that are related to it. You can automate this process by using Net Search Extender's functions for looking up and extracting the related search terms from a thesaurus. A thesaurus is a controlled vocabulary of semantically related terms that usually covers a specific subject area.

DB2 Net Search Extender lets you expand a search term by adding additional terms from a thesaurus that you have previously created. Refer to Chapter 14, "Syntax of search arguments", on page 153 to find out how to use thesaurus expansion in a query.

To create a thesaurus for using it in a search application requires a thesaurus definition file that has to be compiled into an internal format, the thesaurus dictionary.

This chapter describes:

- **The structure of a thesaurus**

  A thesaurus is structured like a network of nodes linked together by relations. This section describes Net Search Extender's predefined relations and how to define your own relations.
- **Creating and compiling a thesaurus**

  Here is a description of the syntax of a thesaurus definition file, and of the tools that you use to compile it into a thesaurus dictionary.

The structure of a thesaurus

A thesaurus is structured like a network of nodes linked together by relations. Net Search Extender looks up a term in a thesaurus by starting at the term, then following a path through the term relations and delivering the terms found in the process.

© Copyright IBM Corp. 1995, 2003

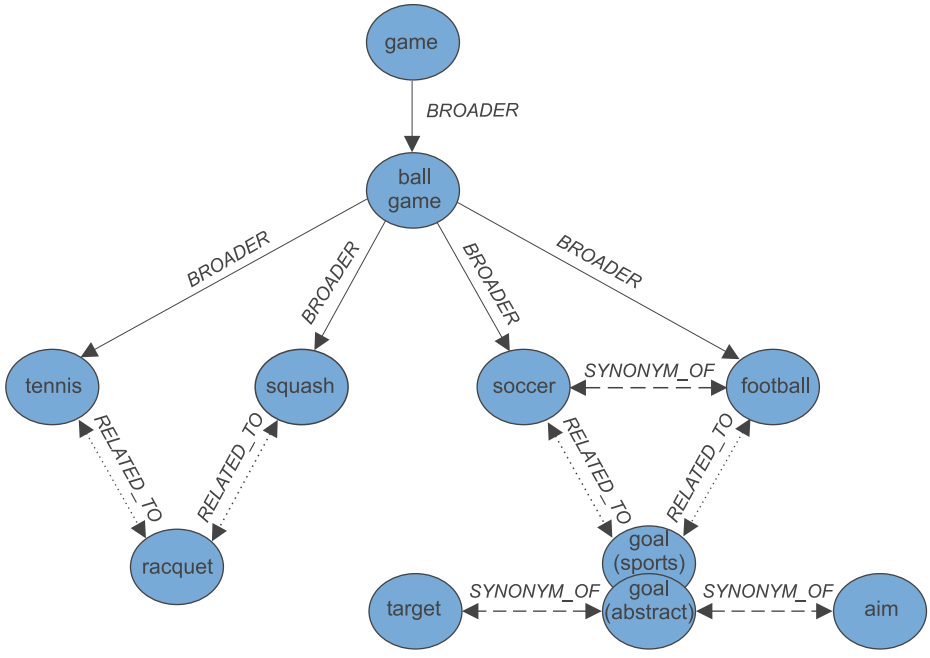


Figure 28. An example of the structure of a thesaurus

Thesaurus entries are connected by relations. Relation names, such as `BROADER`, let you restrict an expansion to certain named lines in the relation hierarchy. Some relations are bidirectional, others are unidirectional; `BROADER`, for example, is the name of a unidirectional relation.

Predefined thesaurus relations

These are the relations that are predefined in the Net Search Extender:

- **Associative relations**

  An associative relation is a bidirectional relation between two terms that do not express the same concept but relate to each other.

  Predefined associative relation: `RELATED_TO`

  Examples:

  ```
  tennis RELATED_TO racket
  football RELATED_TO goal (sports)
  ```

- **Synonym relations**

  A synonym relation is a bidirectional relation between two terms that have the same or similar meaning and can be used as alternatives for each other. This relation can, for example, be used between a term and its abbreviation.

  Predefined synonym relation: `SYNONYM_OF`

  Examples:

```
spot SYNONYM_OF stain
US   SYNONYM_OF United States
```

Figure 28 on page 94 shows two `goal` terms in the same thesaurus. One is specified with the comment `(sports)`, the other with the comment `(abstract)`. Even if terms have the same spelling, synonym relations can connect different word groups. You can model this by using different relations when defining the thesaurus.

- **Hierarchical relations**

  A hierarchical relation is a unidirectional relation between two terms, one of which has a broader (more global) meaning than the other. Depending on its direction, the relation can be used to look up either more specialized or more global terms.

  Predefined hierarchical relations:

  – `NARROWER` to model narrowing relations

    `NARROWER` relations are for modelling a sequence of more specialized terms. The deeper you follow a narrowing relation, the more specific the terms become. For example, if you look up the term `house` along a `NARROWER` relation, the result could be `skyscraper palace church chapel cathedral` and so on, in a list of increasingly specialized terms.

  – `BROADER` to model broadening relations

    `BROADER` relations are for modelling a sequence of more and more global terms. The deeper you follow such a relation, the less specific the terms become. For example, if you look up the term `house` along a `BROADER` relation, the result could be `building construction object` and so on, in a list of increasingly global terms.

Defining your own relations

Net Search Extender lets you define your own `RELATED_TO`, `NARROWER`, and `BROADER` thesaurus relations. Because each relation name must be unique, you must qualify such relations names by the addition of a unique number, like this: `RELATED_TO(42)`.

You can use the same relation number to define a relationship of a different type, such as `NARROWER(42)`. The number 0 is used to refer to Net Search Extender's predefined relations.

Creating and compiling a thesaurus

Use the following steps to create a thesaurus that can be used by the Net Search Extender functions:

1. Create a thesaurus definition file.
2. Compile the definition file into a thesaurus dictionary.

Creating a thesaurus definition file

To create your own thesaurus, your first step is to define its content in a definition file using a text editor.

Restrictions. The length of the file name, including the extension, must not exceed 256 characters. You can have several thesauri in the same directory, but it is recommended that you have a separate directory for each thesaurus.

A sample English thesaurus definition file `nsesamplethes.def` is provided. The thesaurus directory for Windows systems is:

```
<sqllib>\db2ext\thes
```

On UNIX systems, the thesaurus directory is:

```
<instance_owner_home>/sqllib/db2ext/thes
```

Here are the first few definition groups from that file:

```
:WORDS
   accounting
 .RELATED_TO account checking
 .RELATED_TO sale management
 .SYNONYM_OF account
 .SYNONYM_OF accountant

:WORDS
   acoustics
 .RELATED_TO signal processing

:WORDS
   aeronautical equipment
 .SYNONYM_OF turbocharger
 .SYNONYM_OF undercarriage

:WORDS
   advertising
 .RELATED_TO sale promotion
 .SYNONYM_OF advertisement
:
:
:
```

Figure 29. An extract from the sample thesaurus definition file

For the syntax of each definition group, see Chapter 18, “Thesaurus support”, on page 195.

Each member must be written to a single line. Each associated term must be preceded by the relation name. If the member terms are related to each other, specify a member relation.

The length of the member terms and associated terms is restricted to 64 characters. Single-byte characters and double-byte characters of the same letter are regarded as the same. Capital and small letters are not distinct. A term can contain a blank character and either a single-byte character period ″.″ or colon ″:″ can be used.

The user-defined relations are all based on the *associative* type. They are identified by unique numbers between 1 and 128.

Compiling a definition file into a thesaurus dictionary

To compile a thesaurus definition file, run the `db2extth` command. For the command syntax, see "DB2EXTTH (Utility)" on page 145.

To use a thesaurus dictionary within a partitioned environment, ensure that all the physical nodes can access the created files.

Tip

See Appendix M, "Messages returned by the thesaurus tools", on page 277.

Working with a thesaurus

Part 2. Reference

© Copyright IBM Corp. 1995, 2003

Chapter 11. Administration commands for the instance owner

This chapter describes the syntax of administration commands for the instance owner. Instance owner administration consists of checking the status of DB2 Net Search Extender locking and update services, and starting and stopping these services.

For additional information, see Chapter 5, “Net Search Extender instance services”, on page 29.

The commands are a variation of the `DB2TEXT` command and allow for the administration of DB2 Net Search Extender Services that are specific to a DB2 instance.

Command	Purpose	Page
CONTROL	Lists and deletes full-text index locks. Also lists the cache states.	102
START	Starts the DB2 Net Search Extender instance services.	104
STOP	Stops the DB2 Net Search Extender instance services.	105

© Copyright IBM Corp. 1995, 2003

CONTROL

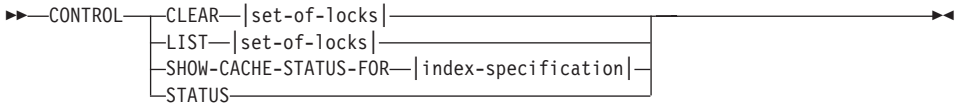
This command lets you list and delete full-text index locks managed by the Net Search Extender Instance Services. If the locking and update services are running, you can view their status as well as information on the activated cache.

In a distributed DB2 environment this only affects the current partition. The user is responsible for invoking the DB2 command, `db2_all` for the desired partitions.

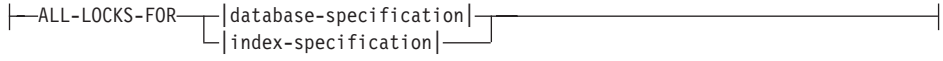
Authorization

You must run this command as a DB2 instance owner on the server.

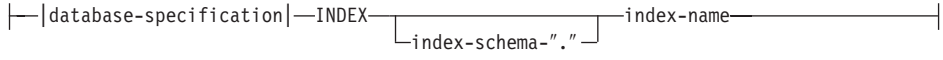
Command syntax



set-of-locks:



index-specification:



Database-specification:

Command parameters

CLEAR
Use CLEAR to force a cleanup for a set of locks. Use this command carefully after checking what leads to a locking problem.

LIST Use LIST to get information about the current locks held for a specific index or database. If there is an update lock, you can get information about the documents that have been processed.

You can also use LIST to view the number of documents updated during the update process. Note this is only for the time the lock is holding the index.

When using a replication capture table, there are no update operations. Instead, insert operations can be either from an insert or an update operation on the source table the index was created on.

set-of-locks
Works with locks only in the specified database or index.

SHOW CACHE STATUS FOR
Shows the activation status for a cached table of the specified index. This can be either: ″Not Activated″ or ″Currently Activated″. If the cache is activated, it displays details about cache memory usage. For example, the maximum cache size (in megabytes), the maximum number of documents to insert, and the space left in the cache table (in kilobytes).

STATUS
By using the STATUS keyword, the command displays whether the locking and update Net Search Extender Instance Services are up and running.

DATABASE database name
The name of the database on the server that is being used.

INDEX index-schema.index-name
The schema and name of the text index that is currently being used. This is specified in the `CREATE INDEX` command.

Usage

When an administration command error message indicates that there is a locking problem, ensure that no conflicting task is running. For example, attempting an `ALTER` command while an `UPDATE` command is running. Then free all the locks for the index.

Use SHOW CACHE STATUS for an incremental index update to check that the specified memory size is still large enough to hold all the update information during the next update, or to check if an activation has been done.

START

This command starts a demon that controls the locking of full-text indexes and the automatic updating of full-text indexes on the DB2 server.

> **Note**
> As the command does not activate any temporary cached table for indexes, individual ACTIVATE CACHE commands are necessary for searching with a stored procedure.

Authorization

You must run this command as a DB2 instance owner on a server, or any of the servers in a distributed DB2 environment.

Command syntax

Command parameters

None.

Usage

On Windows, the command starts a service `db2ext-<InstanceName>`. You can also start this command using normal Window methods.

For locking of full-text indexes, you can modify a configuration file to meet your requirements. See “Locking services” on page 29 for further information.

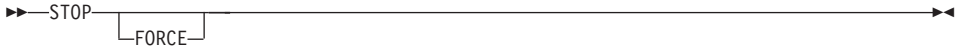
STOP

This command stops the locking and update services of Net Search Extender.

Authorization

You must run this command as a DB2 instance owner on a server, or any of the servers in a distributed DB2 environment.

Command syntax



Command parameters

FORCE
Stops services even if processes are holding locks or if the cached table is activated for any index. If you do not specify `FORCE`, the command will fail in these cases.

Usage

Stopping the Net Search Extender Instance Services will not allow any further use of specific Net Search Extender commands. When restarting the services, you must activate the temporary cache again if you previously used an activated cache with your index.

Note that the activated cache or running Net Search Extender commands will not stop services.

For locking of full-text indexes, you can modify a configuration file to meet your requirements. See “Locking services” on page 29 for further information.

Chapter 12. Administration commands for the database administrator

This chapter describes the syntax of administration commands for the database administrator. Database administration consists of setting up databases for use by DB2 Net Search Extender and then disabling this setup.

Chapter 6, "Creating and maintaining a text index", on page 33 describes how to use these commands.

Only the `ENABLE DATABASE` and `DISABLE DATABASE` commands are a variation of the `DB2TEXT` command, although all these commands allow for administration on the database level.

Command	Purpose	Page
ENABLE DATABASE	Enables the current database to create full-text indexes.	108
DISABLE DATABASE	Resets preparation work completed by DB2 Net Search Extender for a database.	110
DB2EXTDL (utility)	The default UDF to retrieve the content of a data link text column.	112
DB2EXTHL (utility)	The default UDF takes a 100 KB document and returns a 200 KB CLOB.	113

Tip

If no database connection has been specified as part of the `db2text` command, the `db2text` executable causes an implicit connection to be made to the default database specified in the environment variable DB2DBDFT.

© Copyright IBM Corp. 1995, 2003

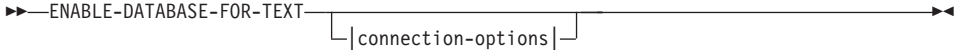
ENABLE DATABASE

This command enables a database to create and exploit full-text indexes on text columns.

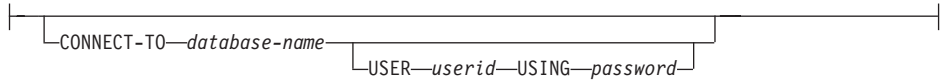
Authorization

You must run this command as a database administrator to enable the database. This requires you having SYSADM authority to be able to grant DBADM to the DB2 instance owner.

Command syntax



connection-options:



Command parameters

CONNECT TO database-name
The name of the database that is a target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command under a user ID with the necessary DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database.

Usage

This command prepares the connected database for use by DB2 Net Search Extender. It is a mandatory step before you can create a DB2 Net Search Extender index on tables/columns in the database.

You can view the database defaults established after running the command by using the `DB2EXT.DBDEFAULTS` catalog view.

Changes to the database
This command grants DBADM authority to the DB2 instance owner associated with the DB2 instance of the enabled database.

The ENABLE DATABASE command creates various database objects in the schema `DB2EXT`, such as DB2 Net Search Extender catalogs, UDFs, and stored procedures. After running the command, the following catalog views are available:

```
db2ext.dbdefaults
db2ext.textindexes
db2ext.textindexformats
db2ext.indexconfiguration
db2ext.proxyinformation
```

Note that DB2 Text Information Extender views are also available for backward compatibility reasons. See Appendix C, “Net Search Extender information catalogs”, on page 209 for further information.

Also note that the above tables are located in the default tablespace of the database, known as `IBMDEFAULTGROUP`. This is distributed over all the nodes defined in `db2nodes.cfg`

Changes to the file system

None.

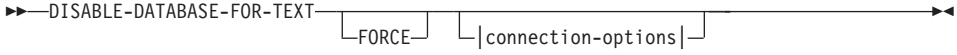
DISABLE DATABASE

This command undoes DB2 Net Search Extender changes to a database.

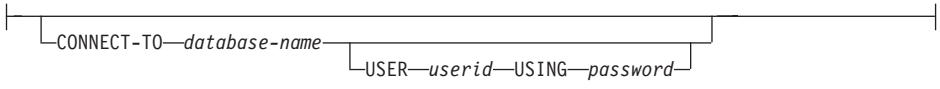
Authorization

You must run this command as a database administrator to disable the database. This requires you having DBADM authority.

Command syntax



connection-options:



Command parameters

CONNECT TO database-name
The name of the database that is a target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command under a user ID with the necessary DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database.

FORCE
Forces the dropping of all DB2 Net Search Extender indexes in the database. See "DROP INDEX" on page 143 for more information.

Usage

This command resets the connected database, so that it can no longer be used by other DB2 Net Search Extender commands. If full-text indexes exist in the database, this command fails unless the FORCE option is used.

This command does not remove DBADM authority from the DB2 instance owner.

Note

Disabling a database will fail if there are any text indexes defined in the database. It is recommended to remove these indexes one by one and then check if any problems occurr. If you use the disable database for text force command, it only guarantees that Net Search Extender catalog tables in the database are removed.

However, if some of the indexes can not be completely dropped, there may still be resources that need to be manually cleaned up. These include:

- Files in the index, work and cache directory
- Scheduler entries in ctedem.dat
- Where an index was created using the replication capture option, the IBMSNAP_SIGNAL, IBMSNAP_PRUNE_SET, and IBMSNAP_PRUNCNTL entries in the tables of the remote database must be manually deleted. These entries can be easily identified using `APPLY_QUAL="NSE"||<instance name> and TARGET_SERVER= <database name>` command.

  In the following example, the instance is `DB2` and the database is `SAMPLE`.

```
DELETE FROM <ccSchema>.IBMSNAP_SIGNAL
WHERE SIGNAL_INPUT_IN IN
        (SELECT MAP_ID FROM <ccSchema>.IBMSNAP_PRUNCNTL
        WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE');

DELETE FROM <ccSchema>.IBMSNAP_PRUNCNTL
WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE';

DELETE FROM <ccschema>.IBMSNAP_PRUNE_SET
WHERE APPLY_QUAL= 'NSEDB2' AND TARGET_SERVER= 'SAMPLE';
```

Changes to the database

The following modifications made in the database to enable DB2 Net Search Extender are deleted:

- The DB2 Net Search Extender catalog views in the database.
- All the database objects created by DB2 Net Search Extender.

Changes to the file system and shared memory

If you use the `FORCE` option, the index files are deleted.

If you use the `FORCE` option, the cache is deleted for any activated cache of indexes. See “DROP INDEX” on page 143 for further information.

DB2EXTDL (utility)

By default, the UDF that retrieves the content of a Data Link text column returns a 100 KB BLOB. Depending on the size of the largest document in the database referenced by the Data Link, you might increase or decrease this value.

Authorization

You must run this command as a database administrator to enable the database. This requires you having SYSADM authority to be able to grant DBADM to the DB2 instance owner.

Command syntax

Command parameters

new-result-size
The new result size of the UDF to retrieve the Data Link content in kilobytes. This is a positive integer of <2097152.

DB2EXTHL (utility)

By default, the highlight UDF takes as input a document up to a maximum size of 100 KB and returns a 200 KB CLOB. Depending on the size of the largest document in the database, you can increase the input value to a maximum size of 1 GB.

Authorization

You must run this command as a database administrator to enable the database. This requires you having SYSADM authority to be able to grant DBADM to the DB2 instance owner.

Command syntax

Command parameters

new-highlight-input-size
The new result size of the highlight UDF in kilobytes. This is a positive integer of <1048576.

Chapter 13. Administration commands for the text table owner

This chapter describes the syntax of administration commands for the text table owner.

Chapter 6, "Creating and maintaining a text index", on page 33 describes how to use these commands.

The commands are a variation of the DB2TEXT command. These allow the owner of a table to create and manipulate full-text indexes on columns of the table.

Command	Purpose	Page
ACTIVATE CACHE	Activates the cache so that search operations using the stored procedure are possible	117
ALTER INDEX	Changes the characteristics of an index	119
CLEAR EVENTS	Deletes index events from an index event table used during index update	123
CREATE INDEX	Creates a full-text index	125
DEACTIVATE CACHE	Deactivates the cache so that search operations using the stored procedure are no longer possible	141
DB2EXTTH (Utility)	Compiles the thesaurus definition file	145
DROP INDEX	Drops a full-text index for a text column	143
ENABLE DATABASE	Enables the current database to create full-text indexes	108
UPDATE INDEX	Starts the indexing process based on the current contents of the text columns	147
HELP	Displays the list of DB2TEXT command options	151
COPYRIGHT	Displays the Net Search Extender product and copyright information	152

© Copyright IBM Corp. 1995, 2003

Tip

If no database connection has been specified as part of the `db2text` command, the `db2text` executable causes an implicit connection to be made to the default database specified in the environment variable DB2DBDFT.

ACTIVATE CACHE

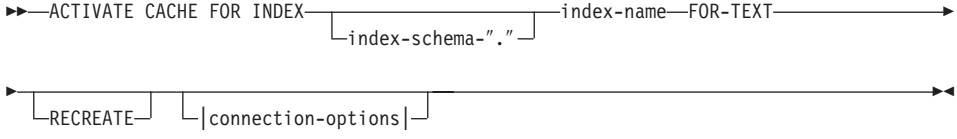
This command activates the cached table from either the DB2 user table or the persistent cache. After completion, search operations using the stored procedure are possible. See Chapter 16, “Stored procedure search function”, on page 175 for further information.

This command is only available if the index was created with a `CACHE TABLE` option. See “CREATE INDEX” on page 125 for further information.

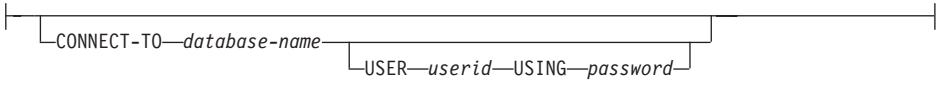
Authorization

According to the DB2 catalog views, the user ID in this command must have CONTROL privilege on the table for which the full-text index was created.

Command syntax



connection-options:



Command parameters

index-schema
The schema of the text index, as specified in the `CREATE INDEX` command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
The name of the text index, as specified in the `CREATE INDEX` command.

RECREATE
Applies only to indexes using a persistent cache; an existing cache is deleted. If an update without activation has completed, the persistent cache is automatically reconstructed from the database.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

You cannot issue the command if one of the following commands is running on the index:

- UPDATE INDEX
- ALTER INDEX
- DROP INDEX
- CLEAR EVENTS
- DEACTIVATE CACHE

Note
Activation of a cached table may require its recreation from scratch, even though a persistent cache was used. This occurs if an update operation was performed whilst the persistent cache was deactivated.

The amount of memory taken to build the cache is dynamically calculated from the current number of documents and the size of the result columns. Use the `PCTFREE` value to increase the calculated minimal amount of memory by a factor of 100/(100-PCTFREE). The `PCTFREE` value is specified in the `CREATE` or `ALTER INDEX` command.

Thereby, `PCTFREE` describes the percentage of the allocated cache that is reserved for insert operations while the cache is activated. Note that for each `ACTIVATE CACHE` command, the actual memory size is re-evaluated.

Changes to the file system
Files for implementing the persistent cache are created.

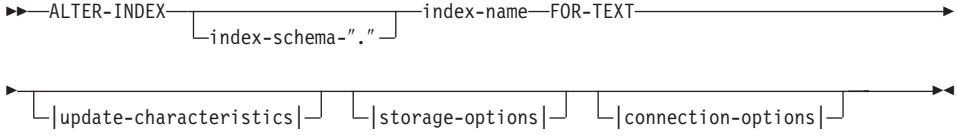
ALTER INDEX

The command changes the characteristics of a full-text index, for example, the update options and the storage options.

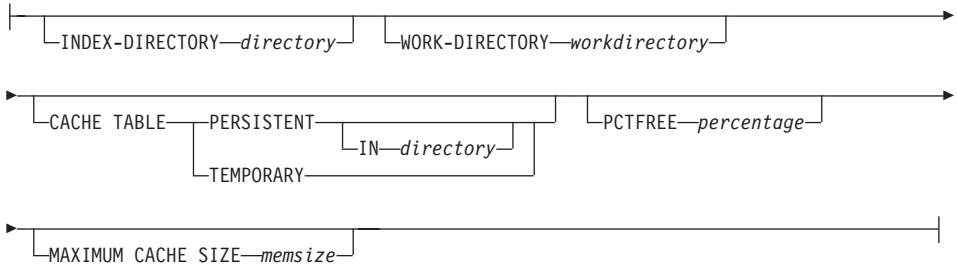
Authorization

According to DB2 catalog views, the user ID in this command must have CONTROL privilege on the table for which the full-text index was created.

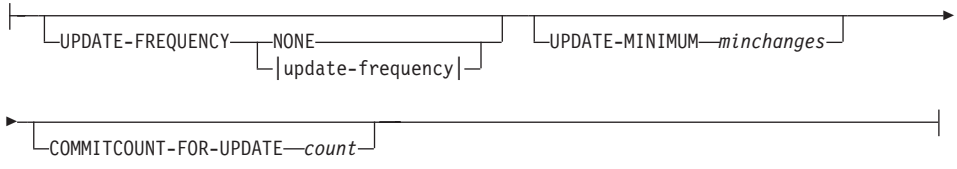
Command syntax



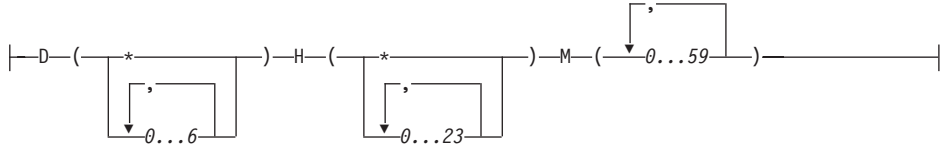
storage-options:



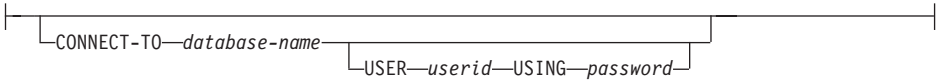
update-characteristics:



update-frequency:



connection-options:



Command parameters

index-schema
The schema of the text index as specified in the `CREATE INDEX` command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
The name of the text index as specified in the `CREATE INDEX` command.

INDEX DIRECTORY directory
The directory path where the text index is stored. As the directory will contain index data, ensure that the directory has read/write and run permissions for the DB2 instance owner user ID.

Note that in a distributed DB2 environment, this directory has to exist on every node. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server. Any index files from the previous index directory are deleted.

WORK DIRECTORY workdirectory
Stores temporary files during search and administration operations. You can change the separate work directory independently of a new index directory.

If the directory does not exist, it is created for the DB2 instance owner user ID. If it exists, ensure that the directory has read/write permissions on UNIX platforms for the instance owner.

Note that in a distributed DB2 environment, this directory has to exist on every node. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server. Any temporary index files from the previous index directory are deleted.

CACHE TABLE PERSISTENT IN directory
Specifies that after a deactivation or system reboot, the cached table in `CREATE INDEX` is persistent. In either case, this allows for a fast `ACTIVATE CACHE` execution. The persistent cache is stored in the specified directory.

The previously created persistent cache is moved to a new location. This location always requires a deactivated index.

CACHE TABLE TEMPORARY
Specifies that the cached result table is now temporary and any previously existing persistent cache has been deleted. Note that this change requires a deactivated index.

MAXIMUM CACHE SIZE memsize
Specifies the new maximum size of the cached table to be built during `ACTIVATE CACHE`. Specify the memsize parameter in megabytes as a positive integer.

If the integer is too small, the `ACTIVATE CACHE` command fails. The actual cache size is calculated during the `ACTIVATE CACHE` command. This change requires a deactivated index.

PCTFREE percentage
Specifies the percentage of the cache held free for additional documents. The percentage must be an integer value less than 100 and greater or equal to 0. Note that the previous persistent cache is deleted and that this change requires a deactivated index. See "ACTIVATE CACHE" on page 117.

UPDATE FREQUENCY
Using the following parameters, the index update frequency determines when the update occurs:

- **`D.`** The day(s) of the week when the index is updated: * (everyday) or 0..6 (0=Sunday)
- **`H.`** The hour(s) when the index is updated: * (every hour) or 0..23
- **`M.`** The minute(s) when the index is updated: 0..59
- **`NONE.`** No further index updates occur. This is intended for a text column in which no further changes are made.

If you do not specify the UPDATE FREQUENCY keyword, the frequency settings are left unchanged.

UPDATE MINIMUM minchanges
The minimum number of changes allowed for text documents before the index is incrementally updated. If you do not specify the UPDATE MINIMUM keyword, the setting does not change.

Note that you can only change the UPDATE MINIMUM if you did not create the index using the RECREATE ON UPDATE option.

COMMITCOUNT FOR UPDATE count
For update processing, you can specify a commitcount. See "UPDATE INDEX" on page 147 for further information. This applies to both the UPDATE command and the UPDATE FREQUENCY specification, which schedules update processing.

Note that you can only change COMMITCOUNT if you did not create the index using the RECREATE ON UPDATE option.

Also note that you cannot change COMMITCOUNT if you did create the index with the REPLICATION clause.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

You cannot issue the command if one of the following commands is running on the index:

- ALTER INDEX
- CLEAR EVENTS
- ACTIVATE CACHE
- DROP INDEX
- UPDATE INDEX
- DEACTIVATE CACHE

In a distributed DB2 environment, a text index with cache options is only allowed on a single-noded tablespace.

Changes to the database
Change DB2 Net Search Extender catalog views.

Changes to the file system

- Creation of `NODE<nr>` subdirectories in the index, and work directories
- Moving of index files
- Creation of persistent cache directories
- Moving of persistent cache files

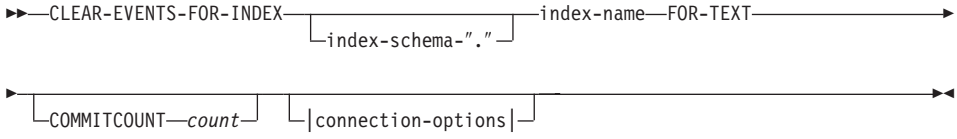
CLEAR EVENTS

This command deletes indexing events from an index's event view. Use the event view for administration purposes. The name of the event view is found in the EVENTVIEWNAME column of the DB2EXT.TEXTINDEXES view.

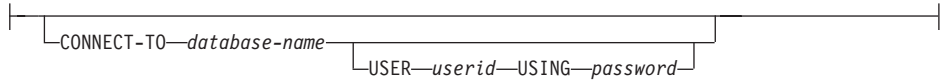
Authorization

According to DB2 catalog views, the user ID in this command must have CONTROL privilege on the table for which the full-text index was created.

Command syntax



connection-options:



Command parameters

index-schema
The schema of the text index as specified in the CREATE INDEX command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
The name of the text index as specified in the CREATE INDEX command.

COMMITCOUNT count
An INTEGER value >=0 displays the number of rows deleted in one transaction by DB2.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a password and userid to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

When you schedule regular updates using the UPDATE FREQUENCY option in the `CREATE` or `ALTER INDEX` commands, regularly check the event table. Use `CLEAR EVENTS` to clean up the event tables, after you have checked the reason for the event and removed the source of the error.

Try to ensure consistency between the contents of the text columns in the table and the index, especially when re-indexing documents.

You cannot issue the command if one of the following commands is running on the index:

- UPDATE INDEX
- ALTER INDEX
- ACTIVATE CACHE
- DEACTIVATE CACHE
- DROP INDEX

CREATE INDEX

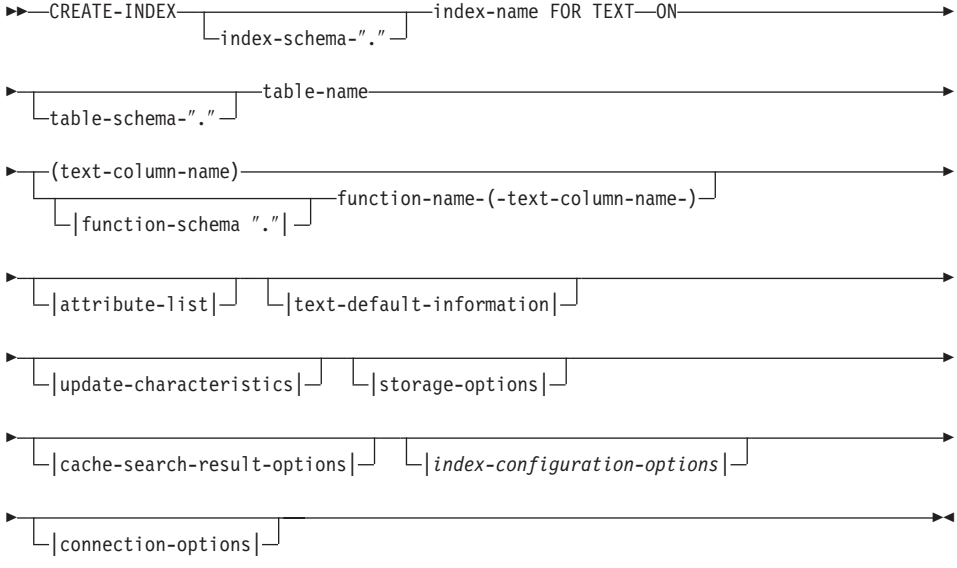
This command creates a full-text index on a text column for use in DB2 Net Search Extender full-text queries.

In a distributed DB2 environment, a full-text index is created on every partition of the tablespace the user table is defined on. Subsequent changes to the distribution of the tablespace are not allowed and will lead to unexpected behavior in the administration commands and during the search process.

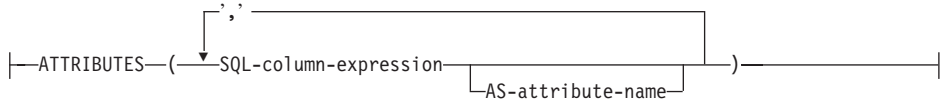
Authorization

According to DB2 catalog views, the user ID in this command must have the CONTROL privilege on the table where the full-text index was created.

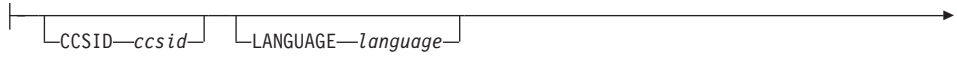
Command syntax

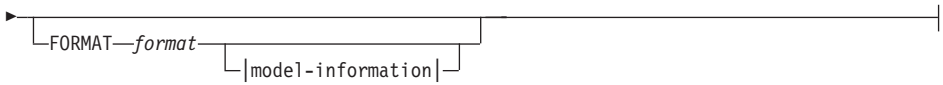


attribute list:

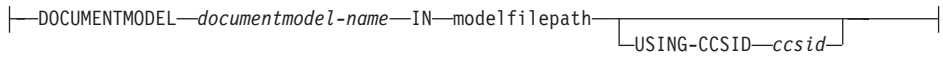


text-default-information:

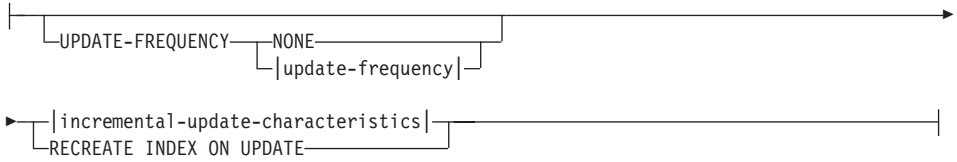




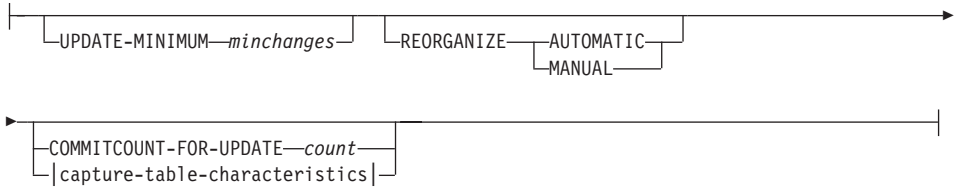
model-information:



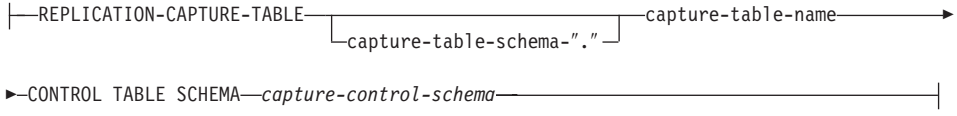
update-characteristics:



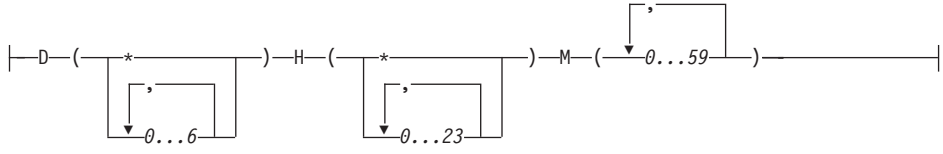
incremental-update-characteristics:



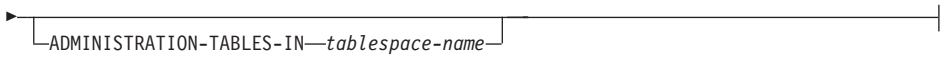
capture-table-characteristics:



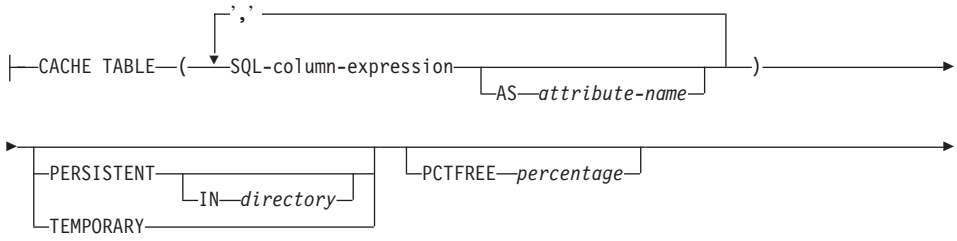
update-frequency:



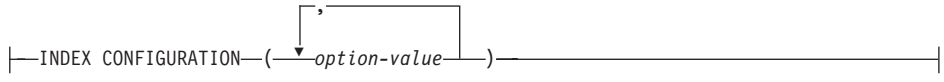
storage-options:



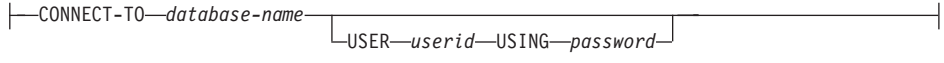
cache-search-results-options:



index-configuration-options:



connection-options:



Command parameters

index schema

The schema of the text index. Use this as a DB2 schema name for the index-specific administration tables. If no schema is specified, the user ID of the DB2 connection is used. Note that the index schema must be a valid DB2 schema name.

index name
The name of the index. Together with the index schema, this uniquely identifies a full-text index in a database. It also serves as the name of the index event table.

See Appendix C, "Net Search Extender information catalogs", on page 209 for details. Note that the index name must be a valid DB2 index name.

table schema
The schema for which of the table, nickname, or view the index is created. If no schema is specified, the user ID of the DB2 connection is used.

table name
The name of the text table, nickname, or view in the connected database that contains the column the full-text index is created for.

Note that when the table name does not refer to a DB2 base table, there are the following restrictions:

- A view only allows a stored procedure or table-valued function search. Therefore, you must specify the key columns for the index or views using the KEY COLUMNS FOR INDEX ON VIEW clause.
- For incremental index updates on nicknames without capture tables, a log table is created. If any changes occur to the data in the nickname table or view, you must manually fill in the log table. With base tables this is done automatically, so the user **must not** touch the log table. For the layout of the log table, see Appendix C, "Net Search Extender information catalogs", on page 209.
- The DB2 predicates `CONTAINS`, `SCORE`, and `NUMBEROFMATCHES` are only allowed for indexes on base tables or nicknames, but not on views.
- Indexes on views are only allowed if you specify cache-search-result options in the command.

text-column-name
The name of the column containing the text used for creating the full-text index. The column must be one of the following types:

- CHAR (FOR BIT DATA)
- VARCHAR (FOR BIT DATA)
- LONG VARCHAR (FOR BIT DATA)
- CLOB
- DBCLOB
- BLOB
- GRAPHIC
- VARGRAPHIC

- LONG VARGRAPHIC
- DATALINK

If the column type is none of these, specify a transformation function using **function-schema.function-name** to convert the column type.

Note that, if you use a Data Link column, the referenced content **is** be fetched for indexing. This is via the protocol that is part of the Data Link value, for example, `Http`. When using protocols other than "file" or "unc", ensure that you support these with servers that are part of the Data Link values. As proxy servers might be necessary to get the file content, the database administrator can specify them in the `DB2EXT.PROXYINFORMATION` table before index creation.

Note that several indexes on the same columns are allowed, but **only** with the following conditions:

The index is created on a view
Therefore, you can not use the index in the `CONTAINS`, `SCORE`, or `NUMBEROFMATCHES` search arguments.

The index is created on a table
If all the indexes are synchronized, they have identical properties on the same column in the following CREATE INDEX command details:

- Function name and schema
- ATTRIBUTES
- CCSID
- LANGUAGE
- FORMAT
- DOCUMENTMODEL
- INDEX CONFIGURATION

Therefore, it does not matter which index is chosen by the `CONTAINS`, `SCORE`, or `NUMBEROFMATCHES` arguments.

function-schema.function-name
The schema and the name of a user-defined function used to access text documents that are in a column of an unsupported type. The function performs a column type conversion, using the one input parameter of an arbitrary column type. It returns the value of one of the Net Search Extender supported types.

ATTRIBUTES (SQL-column-expression AS Attribute-name, ...)
Ensures that the content of a column expression is indexed in addition to the text column. This content can also be searched by the ATTRIBUTE clause in a search statement. The SQL-column

expressions have to be defined using unqualified column names of the table on which the index is created. The only data types allowed are double. Cast operators can be used in the column expressions, but implicit casting of DB2 is **not** possible. The attribute-names must follow the rules for attribute-names in document models and must be different from attribute names in the indexes model-definition file.

Determine the attribute names for expressions by using the following rules:

- If explicitly named by the SQL `AS` clause in the column expression, use the specified name. An example would be: ATTRIBUTES (C1+C2 AS myname)
- If a column of the specified table is used without `AS`, the name of the column is used. For example: CACHE TABLE (C1)
- If an expression is used without `AS` and which does not refer to a named column, CREATE INDEX reports an error.

For example: ATTRIBUTES (CAST(JULIAN_DAY(date) AS DOUBLE) as day, (price1+price2)/2 as avg_price)

Note that attributes without quotes are mapped to uppercase and must be specified in this way during search.

CCSID ccsid
The Coded Character Set Identifier is used when indexing text documents. The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='CCSID'.

LANGUAGE language
For a list, see Appendix E, "Supported languages", on page 225. The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='LANGUAGE'.

FORMAT format
The format of text documents in the column, for example, HTML. This information is necessary for indexing documents. See "Document formats and supported code pages" on page 25 for a list of document formats that are supported for structured documents.

For structured document formats, you can specify information in a document model file. If no document model is specified, the text of the document is indexed using a default document model. See "Document models" on page 181.

If the format keyword is not specified, the default value is from the DB2EXT.DBDEFAULTS view where the DEFAULTNAME='FORMAT'.

DOCUMENTMODEL documentmodel-name IN modelfilepath
The modelfilepath specifies the location of a model file. This contains

a model definition for the format in the FORMAT clause. It must be readable by the DB2 instance owner. A document model enables you to index and search specific sections of a document. You can define markup tags and section names in a document model. A document model is bound to a document format that supports HTML, XML, or GPP structures. You can only specify one document model in a model file.

As document models do not need to be referenced in search conditions, use all the section names in the model file instead. For details on document models, see Chapter 9, "Working with structured documents", on page 91. Note that as the document model is only read during the CREATE INDEX command, any later changes are not recognized for this index.

Note that in a distributed DB2 environment, use a shared file system to ensure the `modelfilepath` is accessible on every node.

USING CCSID ccsid

Specify a CCSID to interpret the contents of the model file. The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='MODELCCSID'.

UPDATE FREQUENCY

The index update frequency determines when the update occurs. If changes to the user table are less than that specified by the UPDATE MINIMUM option, the index is not updated. If you do not specify the UPDATE FREQUENCY, the default NONE is used, so that no further index updates are made. This is useful when there are to be no further changes to a text column.

- **`D.`** The day(s) of the week when the index is updated: * (everyday) or 0..6 (0=Sunday)
- **`H.`** The hour(s) when the index is updated: * (every hour) or 0..23
- **`M.`** The minute(s) when the index is updated: 0..59
- **`NONE.`** No further index updates are made. The update must be started manually.

The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='UPDATEFREQUENCY'.

UPDATE MINIMUM minchanges

The minimum number of changes allowed to text documents before the index is updated automatically by the UPDATE FREQUENCY. Positive integer values are allowed. The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='UPDATEMINIMUM'.

Note that this value is ignored in a DB2TEXT UPDATE command. This option cannot be used with the RECREATE INDEX ON UPDATE option, as the number of changes is not available without a log table and triggers for incremental update.

For distributed databases, the UPDATE MINIMUM is checked on every node.

REORGANIZE AUTOMATIC/MANUAL

Updates performed using the update frequency will only recognize the index if REORGANIZE AUTOMATIC is specified. This step is completed automatically according to the value of `select REORGSUGGESTED from DB2EXT.TEXTINDEXES` after the update.

REORGANIZE MANUAL can only be performed with a manual UPDATE command, using the REORGANIZE option.

If the REORGANIZE clause is omitted, the default is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='AUTOMATICREORG'.

For further information on the REORGANIZE option, see "UPDATE INDEX" on page 147.

REPLICATION CAPTURE TABLE capture-table-schema.capture-table-name CONTROL TABLE SCHEMA capture-control-schema

For incremental update processing, the specified replication capture table is taken instead of a log table that is normally created for the index. Therefore, `schemaname`, `tablename`, and the replication capture table name relate to objects in the local DB2 (federated) database.

The capture-control-schema is the schema name of the replication control tables, for example IBMSNAP_PRUNE_SET on the local DB2. The replication control tables must be available as nicknames on the local DB2 system after setting up the replication.

At minimum, there must be nicknames available for the following capture control tables:

- IBMSNAP_SIGNAL
- IBMSNAP_PRUNE_SET
- IBMSNAP_PRUNCNTL
- IBMSNAP_REGISTER
- IBMSNAP_REG_SYNC (Non-DB2 remote sources only)

As DB2 Replication Center does not automatically guarantee to create local nicknames for a remote capture table and capture control tables, this can be a manual task. The task is similar to creating a nickname for the table that the text index is to be created on.

The column names of primary key columns in the user table nickname and the capture table nickname must match. In addition, the names of the columns IBMSNAP_OPERATION, IBMSNAP_COMMITSEQ and IBMSNAP_INTENTSEQ must not be changed in the capture table nickname.

After index creation, the column names DB2EXT.TEXTINDEXES(LOGVIEWNAME) and DB2EXT.TEXTINDEXES(LOGVIEWSCHEMA) both refer to the local name of the replication capture table.

As Net Search Extender does not require all the functionality of the DB2 Replication Center, the Change Data table (CD) or the Consistent-Change Data (CCD) table must obey following rules:

- Use change capture registration and not the full refresh copying option.
- No horizontal subsetting of capturing changes is allowed. For example, by triggers. See Chapter 6, "Subsetting data in your replication environment" in the *DB2 Replication Guide and Reference, Version 8*.
- Registering changes for a subset of columns is only allowed if the primary key columns, the text column, and all columns involved in the attribute and cache table expressions of the DB2TEXT CREATE INDEX command are included.
- The primary key columns must be included in the capture table. Note that the after-image is sufficient.
- The capture tables must not be condensed. For each primary key there must be one entry with the latest data. However, DB2 Net Search Extender requires a full history to be available.
- The table must use the D/I option. This enables updates to primary keys on the source table to be transformed into a pair of inserts/deletes.

Other prerequisites include:

- The server type and version of the source table the index is created on is one of the following:
  - DB2/AIX V8.1 or later
  - DB2/NT V8.1 or later
  - DB2/HP V8.1 or later
  - DB2/LINUX V8.1 or later
  - DB2/SUN V8.1 or later
  - DB2 z/OS® V7.2 or later

  - DB2 OS/400 V5.2 or later
  - Informix® IDS 9.3
  - ORACLE 9i
  - SYBASE ASE 12.5
  - Microsoft SQL Server 2000
- Supported wrappers include the following:
  - DB2: DRDA®
  - Informix: Informix
  - ORACLE: NET8, (SQLNET)
  - SYBASE: CTLIB
  - MSSQLSERVER: MSSQLODBC3

Notes and restrictions

Ensure that the correct source table name is inserted into the registration table. Depending on the type of remote DBMS, the remote tablename or the local nickname must be used:

- DB2: remote table name (the tablename on the remote server)
- Non-DB2: local nickname (the corresponding nickname in the federated DB2 database)

A user mapping must exist for the local user to access the remote data source via nicknames and the remote user must have control privilege on the tables.

If the DB2 instance owner user ID is different from the local user ID, an additional user mapping for the DB2 instance owner user ID is needed.

The specified base table name must not be a view on a nickname. This is because a view can be over several nicknames and several CD and CCD tables can also be involved. As only one CD or CCD table can be specified in the replication capture clause, a view on nicknames can not be supported. In addition, nicknames on a remote views can not be supported because the primary key is missing.

The CD or CCD table must be a nickname and can not be a view or an alias.

For information on the *DB2 Replication Guide and Reference Version 8*, see "Related information" on page ix.

COMMITCOUNT FOR UPDATE count

For **incremental** update processing a commitcount can be specified, see "UPDATE INDEX" on page 147 for further information. If not specified, a default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='COMMITCOUNT'.

The COMMITCOUNT FOR UPDATE value for the index can be found in `DB2EXT.TEXTINDEXES.COMMITCOUNT`. This can be changed for each index using the ALTER INDEX command. It also applies to the scheduled update processing according to the UPDATE FREQUENCY specification. A value of 0 means that the update is completed in one transaction, with values >0 specifying the number of documents to process in one transaction.

The use of `commitcount` has implications on performance. For information, see "Performance considerations" on page 48.

RECREATE INDEX ON UPDATE

This does not allow incremental index updates, but recreates the index when an update operation is performed (by command or scheduled update). See the Usage Notes on "UPDATE INDEX" on page 147 for additional information.

Note

No triggers are created on the user table and no log table is created.

INDEX DIRECTORY directory

The directory path in which the text index is to be stored. As the directory will contain index data, ensure that the directory has read/write and execute permissions for the DB2 instance owner user ID.

The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME=INDEXDIRECTORY'. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server.

Note that in a distributed DB2 environment, this directory has to exist on every physical node.

WORK DIRECTORY directory

A separate work directory may be specified optionally, that will be used to store temporary files during index search and administration

operations. The directory must exist and have read/write and execute permissions for the DB2 instance owner user ID.

The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='WORKDIRECTORY'. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server.

Note that in a distributed DB2 environment, this directory has to exist on every physical node.

ADMINISTRATION TABLES IN tablespace-name

The name of the regular table space for administration tables created for the index. The table space must exist. If not specified, the tablespace of the user table is chosen, if the index is created on a base table.

In case of a nickname or a view, a default tablespace is chosen by DB2.

When creating text indexes on views, nicknames, or text indexes for stored procedure search on a distributed DB2 environment, the tablespace has to be single-noded.

CACHE TABLE (SQL-column-expression-list)

A cached table is built in addition to the index, which consists of the specified column expressions. This cache is used to return the result set via a stored procedure search without joining full-text search results with a DB2 table. Note that a regular DB2 search using the full-text index with the CONTAINS function is always possible.

Define the SQL-column expressions using unqualified column names of the table the index is created on. The allowed SQL-column expression types are all built-in and user-defined distinct types. The column names in the result set are determined using the following rules:

- If explicitly named by the SQL `AS` clause in the column expression, the specified name is used. For example: `CACHE TABLE (C1+C2 AS myname)`
- If a column of the specified table is used without the `AS` clause, the name of the column is used. For example: `CACHE TABLE(C1)`
- If an expression is used without `AS`, and which does not refer to a named column, CREATE INDEX reports an error.
- No duplicate column names are allowed.

`CLOB` data types are not supported as cache data types. You need to cast these to `VARCHARS`.

Note

Note that if the column names of the result set are not disjunct, the `CREATE INDEX` command returns an error. Also note that the cached table is not implicitly activated after creation, for example search by stored procedure is not possible until DB2TEXT ACTIVATE CACHE is performed.

This option may be used in a distributed DB2 environment only if the user table is stored in a single-noded tablespace.

PERSISTENT IN directory

Specifies that the cache is also created persistent and could be activated shortly after a deactivation or a system reboot. The persistent cache is stored in the specified directory.

Note that if the directory is not specified, the default is taken from the `db2ext.dbdefaults` view, where `DEFAULTNAME='CACHEDIRECTORY'`.

TEMPORARY

Specifies that the cache is not stored persistent. If neither PERSISTENT or TEMPORARY is specified, the default is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='USEPERSISTENTCACHE'.

MAXIMUM CACHE SIZE memsize

Specifies the maximum size of the cached table to be built during DB2TEXT ACTIVATE CACHE. The memsize parameter must be specified in megabytes as a positive integer. There is no default value for memsize. If the integer is too small, the ACTIVATE CACHE command will fail. The actual cache size is calculated during the ACTIVATE CACHE command.

The limit for the maximum cache size on the different platforms is:

- Windows: 1024 MB (1 GB = 1073741824 bytes)
- AIX: 1536 MB (1.5 GB = 1610612736 bytes)
- Solaris, Linux, HP-UX: 2048 MB (2 GB = 2147483647 bytes)

For more information, see Appendix B, "Using large amounts of memory", on page 205.

PCTFREE percentage

Specifies the percentage of the cache to be held free for additional documents. The percentage must be an integer value lower than 100 and greater or equal to 0. If not specified, the default is taken from the `db2ext.dbdefaults` view, where `DEFAULTNAME='PCTFREE'`.

See "ACTIVATE CACHE" on page 117 for details.

INITIAL SEARCH RESULT ORDER (SQL-order-by-list)
Specifies the order used for retrieving the user table contents during initial indexing. When using this option and skipping the dynamic ranking of full-text search results, the documents are returned in their indexing order, as stored in the cached result table.

For further information, see Chapter 16, "Stored procedure search function", on page 175.

> **Note**
> The index order can **not** be ensured for the new or changed documents after incremental update. For example: `INITIAL RESULT ORDER(length(column1) asc, column2+column3 desc)`

KEY COLUMNS FOR INDEX ON VIEW (SQL-columnname-list)
If indexes on views are created, the KEY COLUMNS FOR INDEX ON VIEW clause must be specified, otherwise it MUST NOT be specified. The list of column names specifies the columns that UNIQUELY identify a row in the view.

As this uniqueness cannot be checked by DB2 as in case of primary keys, the user is responsible to ensure the equivalent uniqueness. The specified columns build part of the log table for the index.

INDEX CONFIGURATION (option-value), ...
These are the index configuration values. The default values are underlined.

Option	Values	Description
TreatNumbersAsWords	0 or 1	Interprets sequences of digits as separate words, even if they are adjacent to characters, For example, the 0 default means that tea42at5 is considered as one word.
IndexStopWords	0 or 1	Considers or ignores stopwords during indexing. Currently, the stopword list is an UCS-2 file `<language>.tsw` in directory `<instance>/sqllib/db2ext/resources`. Changes to this file have **no effect** after index creation. Also note that `<language>` is the LANGUAGE value from the CREATE INDEX command.

UpdateDelay	seconds	Specifies the duration in seconds for incremental update without capture tables. Only entries older than this duration will be taken from the log table. This is to avoid lost updates. For example, document changes that are not reflected in the index in transaction scenarios where user transactions interfere with update commands. Therefore, the UpdateDelay parameter should be set to the maximum duration of a user write transaction on the table the index was created on.

CONNECT TO database-name

The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password

Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Changes to the database

- Change DB2 Net Search Extender catalog views.
- Create an index log table in specified table space. This is only if the RECREATE INDEX option is not specified and the capture table is not specified.
- Create an index event table in specified table space.
- Deferred to first update: Creation of triggers on the user text table (only if RECREATE INDEX is not specified and no capture table is used)
- If a replication capture table is used, the following change is made to the capture control tables:
  - an insert into the IBMSNAP_PRUNCTNL and IBMSNAP_PRUNE_SET tables

  The entries in these tables are uniquely identified by the columns:
  - APPLY_QUAL='NSE' || <DB2 instance running NSE>
  - SET_NAME= <internal index identifier>
  - TARGET_SERVER=<DB2 database name target to DB2TEXT operation>

  See page 37 for more information on the columns.

Changes to the shared memory

Deferred to ACTIVATE execution: If CACHE TABLE clause is used, a cache for the result table is built in *shared memory*.

Changes to the file system

- Subdirectories NODE<nr> are created under index, work and cache directories.
- The directory <internal index name> is created under <indexdirectory>/NODE<nr> where indexdirectory refers to the corresponding parameter of this command and NODE<nr> is related to the node number in a distributed DB2 environment.

Usage

Creation of a full-text index requires a primary key on the user table. In DB2 Net Search Extender Version 8.1, a multicolumn DB2 primary key can be used without type limitations. However, to use the table-valued search, no compound primary key is allowed.

The number of primary key columns is limited to 14, the total length of all primary key columns is limited to 1024 - 14 = 1010 bytes.

- The total size of the SQL expressions for ATTRIBUTES, CACHE TABLE and INITIAL SEARCH RESULT ORDER must not exceed 24K bytes.
- Initial index updates are always done as one logical transaction, there is no commitcount in this case.

Note

After creating the index, the length of primary key columns or the view key columns must not be changed by ALTER TABLE commands.

The synchronization between user table, full-text index and the cached result table is completed during the update index command. For further information, see the "UPDATE INDEX" on page 147.

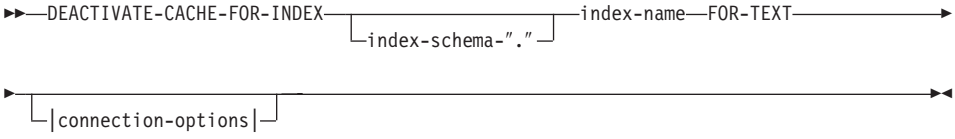
DEACTIVATE CACHE

This command releases a cached table. A persistent cache is kept to be reused on the next ACTIVATE command. Until the next activation, search operations via the stored procedure are no longer possible on the deactivated cache.

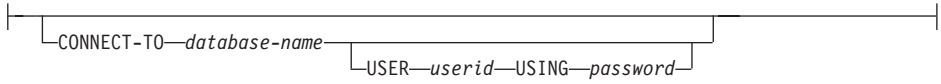
Authorization

According to the DB2 catalog views, the userid in this command must have the CONTROL privilege on the table the full-text index was created for.

Command syntax



connection-options:



Command parameters

index-schema
The schema of the text index as specified in the CREATE INDEX command. If no schema is specified, the userid of the DB2 connection is used as schema name.

index-name
The name of the text index as specified in the CREATE INDEX command.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

Note that this command could not be issued when one of the following commands is running on the index:

- ACTIVATE CACHE

- DEACTIVATE CACHE
- UPDATE INDEX
- ALTER INDEX
- DROP INDEX
- CLEAR EVENTS

Note

After deactivation of a persistent cache, the cache is made inaccessible for search by the stored procedure. However, this can be used for fast ACTIVATE, unless an update was performed in the meantime.

In this case, the persistent cache is automatically recreated from scratch using the ACTIVATE CACHE command.

DROP INDEX

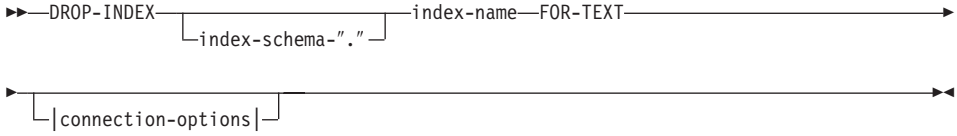
This command drops a full-text index for a text column. If the cache for the index is activated, it is deleted using this command.

Authorization

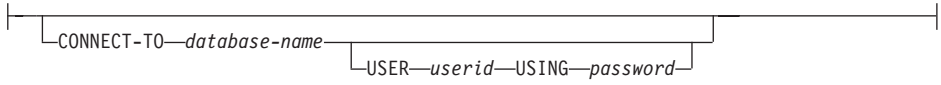
According to DB2 catalog views, the userid in this command must have the CONTROL privilege on the table the full-text index was created for. Alternatively, the user can be the database administrator (DBADM).

Alternatively, the database administrator (DBADM), can drop the index as they must be able to disable the database using the FORCE option.

Command syntax



connection-options:



Command parameters

index schema
The schema of the text index as specified in the CREATE INDEX command. If no schema is specified, the userid of the DB2 connection is used as the schema name.

index-name
The name of the index as specified in the CREATE INDEX command. With the index schema, it uniquely identifies the full-text index in a database.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

The index is deleted, irrespective of the activation status of its cached table. For additional information, see "ACTIVATE CACHE" on page 117 for more information.

Note that the command could not be issued when one of the following commands is running on the index:

- UPDATE INDEX
- CLEAR EVENTS
- ALTER INDEX
- ACTIVATE CACHE
- DEACTIVATE CACHE
- DROP INDEX

Note

Indexes must be manually dropped before or after the user table in DB2 is dropped. If not, the results are not correctly cleaned up.

Changes to the database

- Change DB2 Net Search Extender catalog views
- Drop the DB2 index
- Drop the index log/event tables
- Delete triggers on the user text table

When using the replication capture tables, entries in the IBMSNAP_PRUNE_SET and IBMSNAP_PRUNCTRNL tables are removed.

Changes to the shared memory

The cached table is deleted.

Changes to the file system

- The directory `<internal index name>` is deleted in the index and the work directories of the dropped index
- Delete a persistent cache for the index

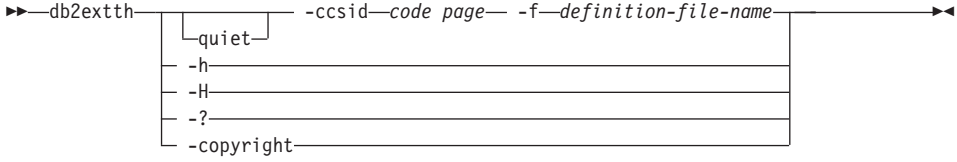
DB2EXTTH (Utility)

This independent utility compiles a thesaurus definition file. After running the thesaurus compiler, the THESAURUS-related features of the search argument syntax can be used.

Authorization

None. This command is not necessarily restricted for the table owner, but makes only sense in the context of querying.

Command syntax



Command parameters

-f definition-file-name
The name of the file containing the thesaurus definition. The file name must contain either the absolute path or the relative path to the file. The file name is restricted to 8+3 characters, the extension being optional.

The thesaurus dictionary is generated in the same directory as the definition file and has the same name. The only difference is that the dictionary has the following extensions: wdf, wdv, grf, grv, MEY, ROS, NEY, SOS, and Ikn, where n is a digit. Note that if existing thesaurus files have the same name, they are overwritten.

-ccsid code page
The code page in which the thesaurus definition file is written. See Appendix L, "Thesaurus supported CCSIDs", on page 275 for a list of the supported code pages for a thesaurus.

-quiet Output information is not displayed.

-copyright
Returns the internal build number of the product. Use this number when reporting problems.

-h, -H, or -?
Displays help information.

Usage

Use this command to compile a thesaurus definition file into a binary thesaurus definition format.

Note

The format is the same as in DB2 Text Information Extender Version 7.2 with the following changes:

- The new relationships BROADER and NARROWER are equivalent to the HIGHER_THAN and LOWER_THAN relationships previously used in Text Information Extender. To refer to these older search relationships, the new relationships must be used.
- Also note that the thesaurus dictionary files must be stored in `<os-dependent>/sqllib/db2ext/thes` to be usable during search, unless the thesaurus is fully qualified in the query.

For additional information, see Chapter 10, "Using a thesaurus to expand search terms", on page 93.

UPDATE INDEX

This command immediately starts the indexing process by bringing the index up to date to reflect the current contents of the text columns with which the index is associated.

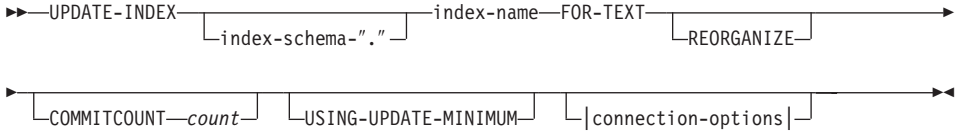
While the update is being performed, search using the CONTAINS predicate is possible. For an index with an activated cached result table, search by stored procedure is also possible during update. However, columns in the cached table may show new values, even though the changed text is not yet committed to the full-text index.

Using the RECREATE INDEX ON UPDATE option in the CREATE INDEX command will clear the index before recreation. Until completion of the update, empty results will be returned.

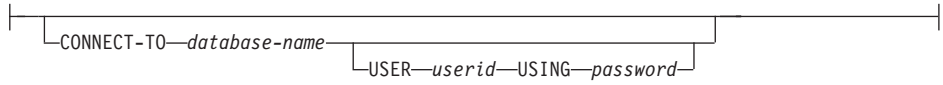
Authorization

According to the DB2 catalog views, the user ID in this command must have the CONTROL privilege on the table the full-text index was created for.

Command syntax



connection-options:



Command parameters

index-schema
The schema of the text index. This is specified in the `CREATE INDEX` command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
The name of the text index. This is specified in the `CREATE INDEX` command.

REORGANIZE
If a text column is frequently updated, then subsequent updates to the index can become inefficient. To make the update process efficient

again, reorganize the index. Use the DB2EXT.TEXTINDEXES view to determine if an index needs to be reorganized.

Use the REORGANIZE AUTOMATIC option of the `CREATE INDEX` command to avoid manually checking and reorganizing the index.

> **Note**
> The reorganization process takes place after a regular update.

USING UPDATE MINIMUM
Uses the UPDATE MINIMUM settings from the `CREATE INDEX` command and starts an incremental update only if the specified number of changes was reached. The default is to unconditionally start the update.

For distributed databases, the UPDATE MINIMUM is checked on every node.

See "CREATE INDEX" on page 125 for additional information.

COMMITCOUNT count
An INTEGER value >=0 displays the number of documents processed in one transaction by the search engine and by DB2 for incremental index updates.

However, for initial updates, such as the first update after the `CREATE INDEX` command, or any update with RECREATE INDEX ON UPDATE option, there is only one logical transaction which ignores COMMITCOUNT. This may be changed using the `ALTER INDEX` command.

CONNECT TO database-name
The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER userid USING password
Use a `password` and `userid` to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage

This command runs synchronously. It starts the update processing on all required DB2 logical/physical nodes in a distributed DB2 environment. The duration depends on the number of files to be indexed and the number of documents already indexed. The status of the update can be seen through a view that is created for each index. The name of this view can be retrieved

from DB2EXT.TEXTINDEXES in column EVENTVIEWNAME. For further information, refer to Appendix C, "Net Search Extender information catalogs", on page 209.

There are two options to view the number of commited documents that have been processed. To determine if an update is still running and how many documents have been committed to the index, use the DB2EXT.TEXTINDEXES (NUMBER_DOCS) view. Use the event view associated with the index for information on starting, committing changes, and finishing update processing.

To view the number of uncommited documents that are to be processed, use the CONTROL LIST ALL LOCKS FOR INDEX command.

Note

The views only display information from the connected node.

For incremental updates on a base table with physical nodes, the time on each node must be synchronized. If the times are not synchronized, updates maybe lost or may not occur at all.

You cannot issue the command if one of the following commands is running on the index:

- CLEAR EVENTS
- ALTER INDEX
- DROP INDEX
- ACTIVATE CACHE
- DEACTIVATE CACHE
- UPDATE INDEX

After updating an index with a deactivated persistent cached result table, the persistent cache is deleted, such that the next `ACTIVATE CACHE` command recreates it based on the database content.

If the user interrupts this command, all processes involved in the update function will stop. If a commitcount was used in an incremental update, some updates may be visible in the index, while others may require a new update command.

To stop the automatic updating of an index, look for the DB2 instance owner process running the update index command on the partition used for update services. Stop this process and the update processing on all the partitions.

Note

As the command works in two separate phases for index creation on all partitions and initial index updates, issue a `db2text drop index` command to ensure that the index is not partly available. If this command is not issued, the next update, which can be triggered by update command or the update frequency option, would perform a complete re-indexing to ensure a consistent state.

Changes to the database

- Inserts to the event table
- Delete from the index log table

When using the replication capture tables, the following changes are made to the database.

- A signal is added to the IBMSNAP_SIGNAL table before starting the initial update
- The synchpoint of IBMSNAP_PRUNE_SET is changed after the incremental update

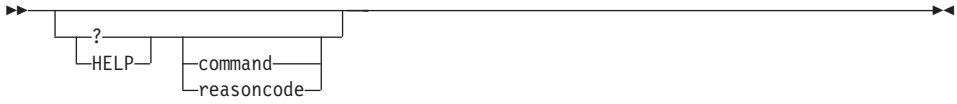
HELP

This displays the list of available DB2TEXT commands, or the syntax of an individual DB2TEXT command.

Authorization

None required.

Command syntax



Command parameters

HELP or ?
Provides help for the specified command or reason code.

command
The first keywords that identify a DB2TEXT command:

- ENABLE
- DISABLE
- CREATE
- DROP
- ALTER
- UPDATE
- CLEAR
- START
- STOP
- CONTROL
- ACTIVATE
- DEACTIVATE

reasoncode
Reason code from a DB2 Net Search Extender command.

Usage

If more than the first keyword is specified, the rest are ignored and the syntax of the identified command is displayed.

If no 'command' parameter is specified after '?' or 'HELP' (or no parameter at all), DB2TEXT lists all available DB2TEXT command parameters.

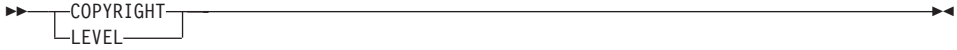
COPYRIGHT

Provides Net Search Extender product and copyright information.

Authorization

None required.

Command syntax



Command parameters

COPYRIGHT / LEVEL

Provides the version copyright statement, version number, and build information for the product.

Chapter 14. Syntax of search arguments

A search argument is the condition that you specify when searching for terms in text documents. It consists of search parameters and one or more search terms.

Examples of search arguments are given in "Specifying SQL search arguments" on page 79, and in a file called `search`. See the "Additional search syntax examples" on page 84.

The SQL scalar search functions that use search arguments are:

CONTAINS
This function uses a search argument to search for text in a particular text document. It returns the INTEGER value 1 if the document contains the text, or any relation specified in the search argument. Otherwise, it returns 0.

NUMBEROFMATCHES
This function uses a search argument to search in text documents and returns an INTEGER value indicating how many matches resulted per document.

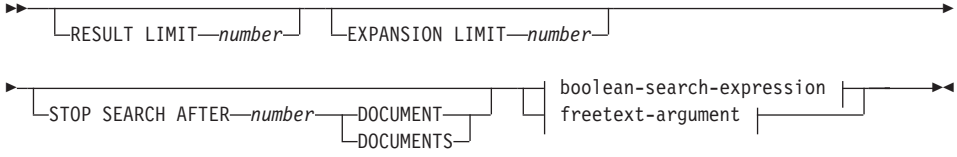
SCORE
This function uses a search argument to search in text documents. It returns a value for each document found, indicating how well the found document is described by the search argument.

Note
You use the same syntax in the search arguments of the stored procedure search and the SQL Table-Valued Function.

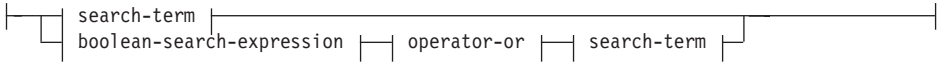
© Copyright IBM Corp. 1995, 2003

Search argument

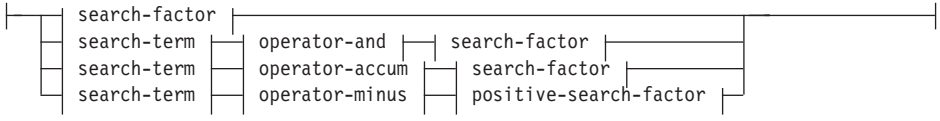
Search argument syntax



Boolean-search-expression:

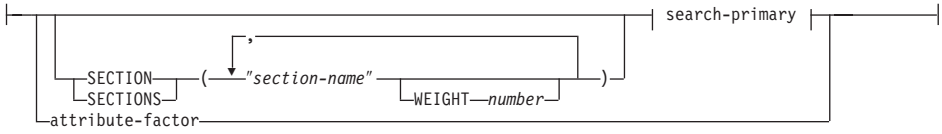


search-term:

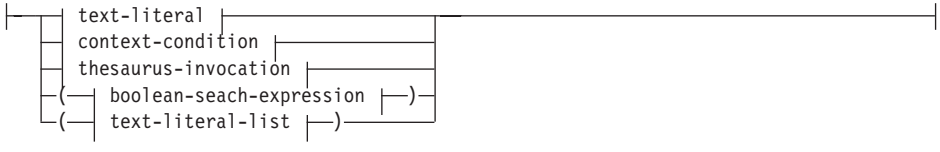


Search-factor:

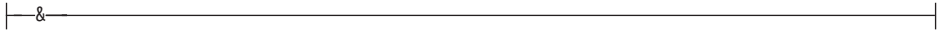
Positive-search-factor:



Search-primary:



Operator-and:

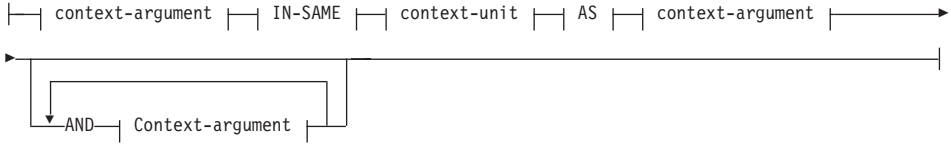


Operator-or:

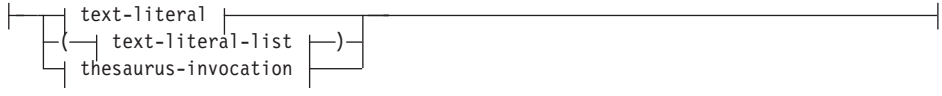
Operator-accum:

Operator-minus:

Context-condition:



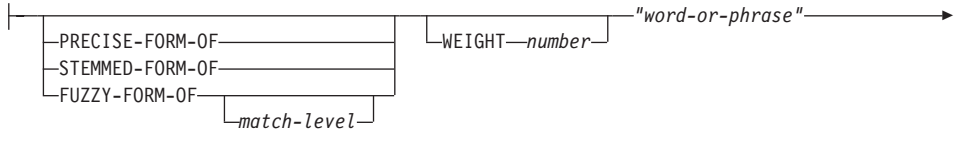
Context-argument:



Text-literal-list:

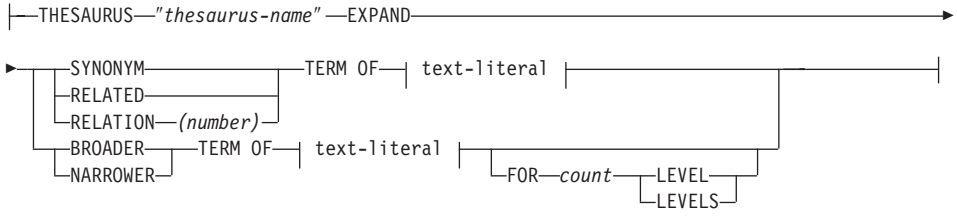
Context-unit:

Text-literal:

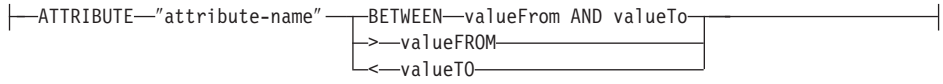


Syntax of search arguments

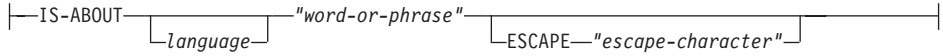
thesaurus-invocation:



Attribute-factor:



freetext-argument:



Examples

Examples are given in "Specifying SQL search arguments" on page 79.

Search parameters

RESULT LIMIT number
A keyword specifying the maximum number of results to be returned by the full-text search.

The RESULT LIMIT should be used together with the SCORE function to ensure that the returned results are scored and only the best results are processed.

EXPANSION LIMIT number
A keyword specifying the maximum number of times a term can be expanded for searching. For example, to determine how many times you can expand the search term 'a*'.

STOP SEARCH AFTER number DOCUMENTS(S)
A keyword specifying the search threshold. The search is stopped when the number of documents is reached during the search and an intermediate result is returned. A lower value will increase the search performance, but may lead to fewer results and omit documents with a potentially high rank.

Note that there is no default value and the *number* value must be a positive integer.

boolean-search-expression

The search-terms and search-factors can be combined using boolean operators NOT, AND, OR, ACCUM, and MINUS according to the syntax diagrams. The operators have the following precedence order (with the strongest first): NOT> MINUS = ACCUM = AND > OR. This can be seen in the following example:

```
"Pilot" MINUS "passenger" & "vehicle" | "transport" & "public"
```

is evaluated as:

```
(("Pilot" MINUS "passenger") & ("vehicle")) | ("transport" & "public")
```

The operator ACCUM evaluates to true, if one of the boolean arguments evaluates to true (which is comparable to the OR operator). The rank value is computed by accumulating rank values from both operands. The ACCUM operator has the same binding (precedence) as AND. The operator MINUS evaluates to true, if the left operand evaluates to true. The rank value is computed by taking the rank value for the left operand and subtracting a penalty, if the right operand evaluates to true.

search-primary

A search-primary consisting of a thesaurus-invocation evaluates to true, if any of the expanded text-literals is found in the (specified section of the document). A search-primary, consisting of a text-literal-list evaluates to true, if any of the text-literals is found in the (specified section of the document).

SECTION(S) *section-name*

A keyword specifying one or more sections in a structured document that the search is to be restricted to. The section name must be specified in a model file specified at index creation time, see "CREATE INDEX" on page 125.

Section names are case sensitive. Ensure that the section name in the model file and query is identical.

This model describes the structure of documents that contain identifiable sections, so the content of these sections can be individually searched. Section names cannot be masked using masking characters. The *positive-search-factor* using the SECTION clause evaluates to true, if the search primary is found in one of the sections.

context-argument IN SAME context-unit AS context-argument AND context-argument ...

This condition lets you search for a combination of text-literals occurring in the same paragraph or same sentence. Context arguments are always equivalent to text-literal-lists, and thesaurus expansion may be used to expand a text-literal to such a list.

The condition evaluates to true, if there is a context-unit (paragraph respectively sentence) in the document, which contains at least one of the text-literals of each expanded context-argument. This can be seen in the following example:

```
("a","b") IN SAME PARAGRAPH AS ("c","d")
          AND THESAURUS "t1" EXPAND SYNONYM TERM OF "e".
```

Assuming e1, e2 as synonyms of e, the following paragraphs would match:

```
".. a c e .." ,  ".. a c e1..",  "a c e2..",
".. a d e .." ,  ".. a d e1..",  "a d e2..",
".. b c e .." ,  ".. b c e1..",  "b c e2..",
".. b d e .." ,  ".. b d e1..",  "b d e2..".
```

PRECISE FORM OF

A keyword that causes the word (or each word in the phrase) following PRECISE FORM OF to be searched for exactly as typed. This form of search is case-sensitive; that is, the use of upper- and lowercase letters is significant. For example, if you search for `mouse`, you do not find “Mouse”.

STEMMED FORM OF

A keyword that causes the word (or each word in the phrase) following STEMMED FORM OF to be reduced to its word stem before the search is carried out. This form of search is not case-sensitive. For example, if you search for `mouse`, you find “Mouse”.

The way in which words are reduced to their stem form is language-dependent. Currently, only English is supported and the word must follow regular inflection endings.

FUZZY FORM OF

A keyword for making a “fuzzy” search, which is a search for terms that have a similar spelling to the search term. This is particularly useful when searching in documents that were created by an Optical Character Recognition (OCR) program. Such documents often include misspelled words. For example, the word `economy` could be recognized by an OCR program as `econony`. Note that the first three characters must match and that fuzzy search cannot be used if a word in the search atom contains a masking character.

match level
An integer from 1 to 100 specifying the degree of similarity, where 100 is more similar than 1. 100 specifies an ″exact match″, and 60 is already considered a very ″fuzzy value″. The fuzzier the match level is, the longer the lapsed search time, since more documents qualify for the search. The default match level is 70.

WEIGHT number
Associates a text-literal with a weight value to change the default score. The allowed weight values are integers between 0 (the lowest score weighting) and 1000 (the highest); the default value is 100.

word-or-phrase
A word or phrase to be searched for. The characters that can be used within a word are language-dependent. It is also language-dependent whether words need to be separated by separator characters. For English and most other languages, each word in a phrase must be separated by a blank character.

To search for a character string that contains double quotation marks, type the double quotation marks twice. For example, to search for the text ″wildcard″ character, use:

```
"""wildcard"" character"
```

Note that in the example, it is only possible to search for one set of quotation marks. You cannot search for two quotation marks in a sequence. There is also a maximum length of 128 bytes for each word or phrase.

Masking characters
A word can contain the following masking characters:

_ (underscore)
Represents any single character.

% (percent)
Represents any number of arbitrary characters. If a word consists of a single %, then it represents an optional word of any length. A word cannot be composed exclusively of masking characters, except when a single % is used to represent an optional word. If you use a masking character, you cannot use THESAURUS. Masking characters cannot follow a non-alphanumeric character.

ESCAPE escape-character
A character that identifies the next character as one to be searched for and not as one to be used as a masking character. For example, if an

escape-character is $, then $%, $_, and $$ represent %, _, and $ respectively. Any % and _ characters not preceded by $ represent masking characters.

THESAURUS *thesaurus-name*
A keyword used to specify the name of the thesaurus to be used to expand a text-literal. The thesaurus name is the file name (without its extension) of a thesaurus that has been compiled using the thesaurus compiler. It must be located in `<os-dependent>/sqllib/db2ext/thes`. Alternatively, the path can be specified preceding the file name.

EXPAND *relation*
Specifies which relation is used to expand the text-literal using the thesaurus. The thesaurus has predefined relations described in the `DB2EXTTH` command. These are referred to using the following keywords:

- SYNONYM, a symmetrical relationship expressing equivalence.
- RELATED, a symmetrical relationship expressing association.
- BROADER, a directed hierarchical relationship that can be followed by specified depth levels.
- NARROWER, a directed hierarchical relationship that can be followed by specified depth levels.

For user-defined relations, use `RELATION(number)`, that corresponds to the relation definition in `DB2TEXTTH`.

TERM OF *text-literal*

The text-literal, to which other search terms are to be added from the thesaurus.

count **LEVELS**

A keyword used to specify the number of levels (the depth) of terms in the thesaurus that are to be used to expand the search term for the given relation. If you do not specify this keyword, a count of 1 is assumed. The value of depth must be a positive integer value.

ATTRIBUTE *Attribute-name*
Searches for documents having attributes matching the specified condition. The attribute-name refers to the name of an attribute expression in the `CREATE INDEX` command, or to an attribute definition in the document model file.

The attribute-factor is allowed for attributes of type double only. The precision of the value is guaranteed for 15 digits. Numbers of 16 characters and above are rounded. Usage of masking characters is not allowed in attribute-name, valueFrom and, valueTo. For an explanation, see the following:

BETWEEN valueFrom AND valueTo
A BETWEEN attribute factor evaluates to true if the value of the attribute is greater than (not equal to) valueFrom and lower than (not equal to) valueTo.

>*valueFrom*
A ″>″ attribute factor evaluates to true if the value of the attribute is greater than (not equal to) valueFrom.

<*valueTo*
A ″<″ attribute factor evaluates to true if the value of the attribute is lower than (not equal to) valueTo.

If the attribute name in the `CREATE INDEX` command is specified with quotes, or is defined in a model file, the specified attribute name must match exactly. Whereas, if no quotes are specified in the `CREATE INDEX` command, the attribute name must be in uppercase.

IS ABOUT language word-or-phrase
An option that lets you specify a free-text search argument. It should be used to get a different kind of score algorithm as it checks the positioning of the terms within the documents. The closer together the terms used in the word-or-phrase are, the more terms are included in the document and the higher the score value returned.

The values allowed for language are described in Appendix E, “Supported languages”, on page 225, and are only relevant for the Thai language. If not specified, the language en_US is used as default. The language is used only for tokenization of the word-or-phrase.

Note that IS ABOUT is useful only if the score values are requested and the search results are ordered by score values.

Syntax of search arguments

Chapter 15. SQL scalar search function and the SQL table-valued function

DB2 Net Search Extender provides SQL scalar search functions and a SQL table-valued function for searching text documents stored in DB2 Universal Database.

This chapter describes the following SQL search functions.

A summary of the search functions

Search function	Purpose	Page
CONTAINS	Searches for text in a particular document.	164
NUMBEROFMATCHES	Searches and returns the number of matches found.	165
SCORE	Searches and returns the score value of a found text document.	166
DB2EXT.TEXTSEARCH	The SQL table-valued function returns a table of found primary keys, a number of matches, and/or score values.	167
DB2EXT.HIGHLIGHT	To get information about why a document qualified as a search result.	171

See Chapter 8, “Searching”, on page 77 for examples of using the SQL scalar search functions and the SQL table-valued function.

© Copyright IBM Corp. 1995, 2003

CONTAINS

The CONTAINS scalar function searches for text in a text document indexed by Net Search Extender. It returns the INTEGER value 1 if the document contains the text, or any relation specified in the search argument. Otherwise, it returns 0.

Function syntax

Function parameters

column name
The name of a column. The column must have an associated text index. You can create text indexes by using the administration command `DB2TEXT CREATE INDEX`.

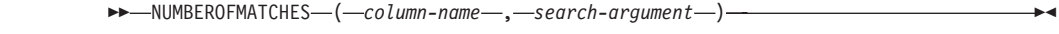
search-argument
A string of type VARCHAR containing the terms to be searched. See Chapter 14, “Syntax of search arguments”, on page 153.

Note
You cannot use the CONTAINS query on a text index created on a view.

NUMBEROFMATCHES

The NUMBEROFMATCHES scalar function searches in text documents and returns an INTEGER value indicating how many matches resulted per document.

Function syntax



Function parameters

column name
The name of a column. The column must have an associated text index. You can create text indexes by using the administration command `DB2TEXT CREATE INDEX`.

search-argument
A string of type VARCHAR containing the terms to be searched. See Chapter 14, "Syntax of search arguments", on page 153.

Note
You cannot use the NUMBEROFMATCHES query on a text index created on a view.

SCORE

The SCORE scalar function searches in text documents and returns a score value for each document found, indicating how well the found document is described by the search argument.

SCORE returns a DOUBLE value. As the search term appears more frequently in the document, the score of the document increases.

Function syntax

Function parameters

column name
The name of a column. The column must have an associated text index. You can create text indexes by using the administration command `DB2TEXT CREATE INDEX`.

search-argument
A string of type VARCHAR containing the terms to be searched. See Chapter 14, “Syntax of search arguments”, on page 153.

Note
You cannot use the SCORE query on a text index created on a view.

DB2EXT.TEXTSEARCH

In addition to the stored procedure search and the SQL scalar search functions, Net Search Extender provides two SQL table-valued functions which look very similar to the stored procedure.

Both table-valued functions are called `db2ext.textsearch`. The only difference between them is that one supports the HIGHLIGHT function and has two additional parameters, `numberOfHits` and `hitInformation`.

Note that you can not use the table-valued function on tables with a compound primary key.

For information on using the HIGHLIGHT function, see "DB2EXT.HIGHLIGHT" on page 171.

Note

The table-valued function may be used in a distributed DB2 environment only if the user table is stored in a single-node tablespace. You must also ensure that you connect to the correct node using the `DB2NODE` environment variable.

Function syntax

1. db2ext.textsearch without highlight support

```
db2ext.textSearch
                (
                query             VARCHAR(4096),
                indexSchema       VARCHAR(128),
                indexName         VARCHAR(128),
                resultFirstRow    INTEGER,
                resultNumberRows  INTEGER,
                primKeyBinding    <supported types>,// same type as primary key
                )

                return table
                (
                primKey           <supported types>,// same type as primary key
                numberOfMatches   INTEGER,
                score             DOUBLE,
                totalNbResults    INTEGER
                )
```

2. db2ext.textsearch with highlight support

```
db2ext.textSearch
(
query             VARCHAR(4096),
indexSchema       VARCHAR(128),
indexName         VARCHAR(128),
resultFirstRow    INTEGER,
resultNumberRows  INTEGER,
primKeyBinding    <supported types>,// same type as primary key
numberOfHits      INTEGER
)

return table
(
primKey           <supported types>,// same type as primary key
numberOfMatches   INTEGER,
score             DOUBLE,
totalNbResults    INTEGER
hitInformation    BLOB(20K)
)
```

Function parameters

The following are input parameters.

query See Chapter 14, "Syntax of search arguments", on page 153 for additional information.

indexSchema, indexName
Identifies the index to search. For more information, see "CREATE INDEX" on page 125.

resultFirstRow
The result list of the query is returned in parts. This parameter describes which row of the query result list is the first one to be entered into the result table of the table-valued function. The value should be >= 0.

Note that the number 0 identifies the first row in the query result list.

resultNumberRows
This parameter describes how many rows of the query result list are entered into the result table of the table-valued function, and where 0 means that all the results need to be returned.

Note that this is different from the result limit query parameter that determines the maximum size of the query result list.

primaryKeyBinding
The type of this parameter determines the type of the primaryKey Output parameter. If the text index has been created for a base table with a primary key of type `<type1>`, then `primaryKeyBinding` must also be of type `<type1>`.

Additionally, the parameter determines the scope of the text search. If primaryKeyBinding is set to NULL ("CAST(NULL as <type1>)", the scope of the search will be all the documents stored in the index. Alternatively, you can restrict the search to documents primaryKeyBinding is bound to.

For example, if primaryKeyBinding is set to CAST(5 as BIGINT), you restrict the search to the single document with the BIGINT primary key value of ″5″.

Note that only single column primary keys of the following types are supported: SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, VARCHAR FOR BIT DATA, DATE, TIME, and TIMESTAMP.

numberOfhits
This option specifies the maximum number for hit information returned by the db2ext.textsearch function. If 0 is specified, the information for up to a maximum of 1100 hits is provided. This process may be time-consuming.

Note that this parameter is only necessary for constructing the highlight information required by the db2ext.highlight function.

Function parameters

The following return values are stored in a temporary table which needs to be joined to your normal table if further results are requested. Note that the NUMBEROFMATCHES, SCORE, TOTALNUMBEROFRESULTS, and HITINFORMATION are only calculated if they are requested in your select statement.

primKey
The primary key of the found document.

numberofmatches
NUMBEROFMATCHES is an INTEGER value indicating how many matches resulted for each document.

score Score returns a DOUBLE value. As the search term increases in frequency in the document, the document score increases.

totalNumberOfResults
The query result list denotes how many results were found. Note that each row has the same value.

Also note that when you use the STOP SEARCH AFTER, or the RESULT LIMIT together with the SCORE syntax in a query, this number is no longer reliable.

hitInformation
The hit information returned by db2ext.textsearch is necessary for highlight processing. Currently, hit information for approximately 1100

hits can be contained in this output parameter. If the number of hits exceeds this threshold, hit information for these further hits is ignored.

Note that this value is only returned if you specify `numberOfHits`.

Usage

With the SQL table-valued function you are able to search on views in the same way you do with the stored procedure search. The exception being that no shared memory is needed, so the index does not need to be activated.

This function is primarily for those users who have used an SQL query within the stored procedure search. However, the restriction is that only a single column primary key on base tables is supported.

The following example shows how you can work on a multi-column primary key table:

```
select s.id from
db2ext.sample s, table (db2ext.textSearch(
        '"characteristics"',
         'DB2EXT',
         'COMMANDS',
         1,
         20,
         cast(NULL as INTEGER))) t
where s.id = t.primkey
```

In this example, you must first create a view on this table with a single unique key and then create the index on this view.

For an example of using the SQL table-valued function with the `db2ext.highlight` function, see page 172.

DB2EXT.HIGHLIGHT

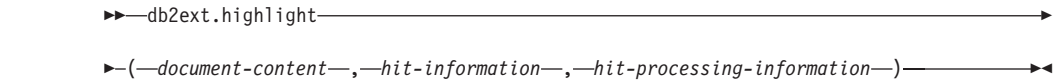
Use the `db2ext.highlight` function to get information about why a document qualified as a search result. More specifically, it can be used to:

- get hits
- get hits and surrounding text
- get the document with user-defined highlight tags surrounding the hits.

Note that the `db2ext.highlight` function can only be used with the `db2ext.textsearch` table-valued function. The table-valued function searches the index providing the results for the HIGHLIGHT function to use.

For information on using the `db2ext.textsearch` function, see "DB2EXT.TEXTSEARCH" on page 167.

Function syntax



Function parameters

The following are input parameters:

document content CLOB(100K)
Only UTF8 documents of a TEXT or XML format are supported. To increase this value, see "DB2EXTHL (utility)" on page 113.

hit information BLOB(20K)
A string containing hit information. This is returned by the `db2ext.textsearch` function, if the `numberOfHits` parameter is specified.

hit processing information VARCHAR(1024)
This parameter is a list of option value pairs separated by a comma ',' character with each string character surrounded by ″ ″ characters. It specifies how highlighting should be processed for the specified document. If none of the options are specified, the original document is returned.

TAGS = (″STRING″, ″STRING″)
This option enables the user to specify the tags to be inserted before and after a hit in the document. If this option is omitted, no tags are added before and after a hit in the document.

WINDOW_NUMBER = INTEGER
This option specifies how many parts (or windows) of the

document should be returned by the highlight function. Each window contains one or more hits and the first hit in each window determines the part of the document returned to the user. These hits may or may not have text surrounding the hit.

If this option is omitted, 0 is taken as the default and the entire document containing start and end tags (if specified) is returned. In this case, the `WINDOW_SIZE` option is ignored.

WINDOW_SIZE = INTEGER
This option specifies the recommended size of the window in bytes. This actual size may vary, depending on the number of hits, length of hits and the start and end tag sizes. If the option is omitted, 0 is the default and only hits without surrounding text will be returned.

WINDOW_SEPARATOR = "STRING"
This option specifies the tag used to separate one window from the next window. If the option is omitted, "..." is the default value.

FORMAT = "STRING"
This option specifies the format of the document. Valid values are `XML` or `TEXT`. If this option is omitted, then `TEXT` is taken as the default value. Ensure that the format value is the same as that specified during indexing.

MODEL_NAME = "STRING"
This option specifies the model name related to the specified XML document. Note that if the `FORMAT` is `TEXT`, this option results in an error condition.

SECTIONS = ("section-name1", ..., "section-nameN")
For XML documents, highlighting can be restricted to relevant sections. For example, they can be defined in the model file. To specify these sections, separate the one or more section names with a comma. If this option is omitted, highlighting is performed on the whole XML document. Note that if the `FORMAT` is `TEXT`, this option is ignored.

Function parameters

The following are return parameters.

CLOB(200K)
The HIGHLIGHT function returns a CLOB value containing the document parts modified by the HIGHLIGHT function.

Usage

The following example shows how you can use the HIGHLIGHT function:

```
select p.id,
       p.title,
       db2ext.highlight(p.content,
       t.hitinformation,
       'TAGS = ("<bf>", "</bf>"),
       WINDOWS_NUMBER = 5,
       WINDOWS_SIZE = 200,
       WINDOW_SEPARATOR = "...",
       FORMAT = "XML",
       SECTIONS = ("section1-name", "section2-name")')

FROM patent p, table (db2ext.textsearch(
       '"relational database systems"',
       'DB2EXT',
       'TI_FOR_CONTENT',
       0,
       20,
       CAST(NULL as BIGINT),
       15)) t

WHERE p.id = t.primkey
```

Using documents larger than 100 KB will cause the SQL query to terminate and produce an SQL error (SQL 1476N and sql error -433). To avoid this, use the `db2exthl` command to increase the document content size. For information, see "DB2EXTHL (utility)" on page 113.

Note

Special characters, such as ″newline″ will be returned as is.

Restrictions

- Only XML and flat text documents are supported.
- Only UTF8 databases are supported. For binary or datalink documents, you need to ensure that the documents are in UTF8.
- Thai documents are not supported.
- If there is a mismatch between the document format used during indexing and query time the HIGHLIGHT function will return unpredictable results.
- Only hits found in the text parts of a document will be highlighted.
- The highlight function can only be used with the `db2ext.textsearch` function.
- String values cannot contain the ″ character.

SQL highlight function

Chapter 16. Stored procedure search function

Net Search Extender provides a stored procedure search for returning predefined result tables. The result table is specified in the cache table section during create index. Use the stored procedure search when you need to return a small number of results in a specific order.

An example would be an Internet application where the first 20 rows are returned, but the rest of the results can also be returned in increments of 20 rows.

Note

The stored procedure function may be used in a distributed DB2 environment only if the user table is stored in a single-node tablespace.

You must also ensure that you connect to the correct node using the `DB2NODE` environment variable.

© Copyright IBM Corp. 1995, 2003

DB2EXT.TEXTSEARCH (for stored procedure search)

Function syntax

```
db2ext.TextSearch(

        IN      query                   VARCHAR(4096),
        IN      indexSchema             VARCHAR(128),
        IN      indexName               VARCHAR(128),
        IN      resultFirstRow          INTEGER,
        IN      resultNumberRows        INTEGER,
        IN      scoringFlag             INTEGER,
        IN      searchTermCountsFlag    INTEGER,
        OUT     searchTermCounts        VARCHAR(4096),
        OUT     totalNumberOfResults    INTEGER )
```

Function parameters

The following are input parameters.

Query See Chapter 14, "Syntax of search arguments", on page 153 for further information.

indexSchema, indexName
To identify the index to search. Refer to "CREATE INDEX" on page 125.

resultFirstrow
The query result list is returned in parts. The parameter describes which row of the query result list is the first one to be put into the result set of the stored procedure. The first row in the query result list is identified by the number 0.

resultNumberRows
This parameter describes how many rows of the query result list are put into the result set of the stored procedure.

This is not to be confused with the ″result limit″ expression in the query, which determines the maximum size of the query result list.

The value should be >= 0. Where 0 means that all the results need to be returned.

> **Note**
> If a larger result set is requested, ensure that a temporary user tablespace is available. If there is none available, then create a tablespace. The following example creates a tablespace on a UNIX platform:
>
> ```
> db2 "create user temporary tablespace tempts managed by system
>             using ('/work/tempts.ts')"
> ```

scoringFlag
0 means there is no scoring and 1 means there is scoring. If scoring is requested, an additional column with the score values is returned with the highest value first.

searchTermCountsFlag
This controls the searchTermCounts processing. If searchTermCountsFlag is 0, the searchTermCounts is not calculated.

Function parameters

The following are output parameters.

searchTermCounts
The number of occurrences of each search term query in the index. These counts are returned as a blank separated list in the order of search terms in the query.

See the **searchTermCountsFlag** for information.

totalNumberOfResults
The total number of results found in the query result list.

Also note that when you use the `STOP SEARCH AFTER`, or the `RESULT LIMIT` together with the `scoringFlag` syntax in a query, this number is no longer reliable.

Usage

The columns in the result set returned by the stored procedure are given by the `CACHE TABLE` option of the `DB2TEXT CREATE INDEX` command. If `scoringFlag=1`, then a column of type double is added. This column contains the `SCORE` value.

Use the following options to increase the performance of a second query with the same string as the first query. Note that this must be in a different cursor window with no `totalNumberOfResults` required:

- If you do not require scoring, add the following syntax: `STOP SEARCH AFTER x DOCUMENTS`, where `x` is the `resultFirstRow + resultNumberRows`.
- If you require scoring, add the following syntax: `STOP SEARCH AFTER y DOCUMENTS`, where `y` is equal to the `totalNumberOfResults` in the first query.

To ensure that you connect to the correct node for searching, it may be necessary to set the `DB2NODE` environment variable.

For UNIX, use the following command:

```
export DB2NODE=<no>
```

Note that it is important that all physical nodes have a synchronized time.

For Windows, use:

```
set DB2NODE= <no>
```

Note

A fenced user ID that is different from the instance owner ID does not work with partitioned databases.

Stored procedure with SQL query

This stored procedure allows the combing of results from a text query with an additional SQL query. Its use is restricted to indexes created on a table/view with a single key-column.

```
Textsearch(
    IN        query                    VARCHAR(32000),


TextSearchSql(
    IN      query                  VARCHAR(32000),
    IN      indexSchema            VARCHAR(128),
    IN      indexName              VARCHAR(18),
    IN      resultFirstRow         BIGINT,
    IN      resultNumberRows       BIGINT,
    IN       scoringFlag           INTEGER,
    IN       sqlQuery              VARCHAR(32000),
    INOUT    searchTermCounts      VARCHAR(32000),
    OUT      totalNumberOfResult   BIGINT)
```

Input parameters

See `TextSearch` stored procedure for information, as well as:

sqlQuery

An SQL statement that specifies the columns and rows of the result table returned by the stored procedure. The statement has following structure for non-ranked queries (ranking-Flag=0):

```
select ...from ... where ...
```

The where-clause (belonging to a from-clause referencing the table, or view the index was created on), must contain ″<keycolumn> in (%s)″ in a place where SQL allows an IN-predicate.

The statement has following structure for ranked queries (ranking-Flag=1):

```
select ...from ... where ...
```

The select-clause must contain a column-reference ″RSCORE″ in a place where SQL allows an expression. The from-clause must contain a ″%s″ where SQL allows a table-reference. The where-clause (belonging to a from-clause referencing the table or view the index was created on), must contain ″%s″ in a place where SQL allows an IN-predicate.

Input/Output parameters

See `TextSearch` stored procedure for information.

Output parameters

See `TextSearch` stored procedure for information.

Stored procedure search function

Output Result Set

See `TextSearch` stored procedure for information.

Error handling

See `TextSearch` stored procedure for information.

Chapter 17. Structured document support

Structured documents consist of document models and document file definitions.

Document models

A document model primarily controls what parts of a document's structure need to be indexed and how they are indexed. Its purpose is to:

- Identify text fields that should be distinguished in the source document
- Determine the type of such a text field
- Assign a field name to the text field

When the document model identifies text as belonging to a text field, the text is considered to be part of the textual content of the document, and terms are extracted and stored in the index.

The elements of a document model vary depending on the parser used for that document format:

- For HTML format, a document model uses the HTML tag names to define which tags should be indexed, and how to handle meta-tag information.
- For XML format, there is no predefined set of tags, so a document model must first define which tags are of interest. XML elements of the same name can also be distinguished based on what other elements they are embedded in.
- For GPP (general purpose parser) format, the document model interacts even more deeply with the parser, because it has to determine the boundaries of the text fields. Here the field definition must specify strings for detecting the boundaries of fields.
- For Outside-In formats, a document model uses tags similar to HTML tag names to define which tags should be indexed, and how to handle meta-tag information. Note that the Outside-In filtering format is also known as INSO.

See the relevant ″Defining a Document Model″ section for information.

For information on the document model syntax in the form of a Document Type Definition (DTD), and text field limitations, see Appendix G, “Document model reference”, on page 253.

© Copyright IBM Corp. 1995, 2003

Default document models

For HTML and XML documents, Net Search Extender provides default document models that are used if you do not define a document model. For structured plain text documents, you must provide and specify a document model.

If you use one of the default document models:

- All fields are indexed, and no special information, such as meta information, is extracted.
  - For HTML, each field is assigned the name of the corresponding tag.
  - For XML, the generated field name is the complete tag path, for example `/play/role/name`.
- No numeric attribute is indexed (as no numeric attribute is defined in the default document model).

Table 6. Behavior of the default document models for the supported document formats

Document type	Behavior of the default document model
HTML	Accepts these as text fields: <a> <address> <au> <author> <h1> <h2> <h3> <h4> <h5> <h6> <title>. Field name is the tag name, for example ″address″.
XML	Accepts all tags as text fields. Field name is the tag path name in Xpath notation, for example ″/play/title″.
Structured plain text (GPP)	No default document model.
Outside-In (INSO)	Accepts as text fields, the document properties shown in “Element parameters” on page 193 as returned by the Outside-In filters. The Field name is the name of the document property used by Outside-In, for example: ″SCCCA_TITLE″. There is no attribute support.

For each type of document, a document model is defined. As the models are all different, an example and explanation is provided for each one.

Note

Although the default document models do correctly process documents, for better indexing and search you should define your own document models.

With the default document model, the text of a document is fully indexed regardless of whether or not it is part of a text field. This means that unrestricted text searches include a search of that text.

Defining a document model for structured plain-text documents

Here is an example of a general-purpose (GPP) structured plain-text document:

```
[head]Handling structured documents
[/head]
[year]2002
[/year]
[abstract]This document describes the concept of structured documents
and the use of document models to...
[/abstract]
```

Here is an example of a GPP document model:

```
<?xml version="1.0"?>
<GPPModel>

  <GPPFieldDefinition
  name="Head"
  start="[head]"
  end="[/head]"
  exclude="YES" />

  <GPPFieldDefinition                  - This is the start of text field
  name="Abstract"
  start="[abstract]"
  end="[/abstract]"
  exclude="NO" />                     - This is the end of a text field

  <GPPAttributeDefinition              - This is the start of a document
  name="year"                               attribute
  start="[year]"
  end="[/year]"
  type="NUMBER" />                    - This is the end of a document
                                            attribute
</GPPModel>
```

The first line, `<?xml version="1.0"?>` specifies that the document model is written using XML tags. Note that this model is not written for XML format documents.

Each field is defined within a `GPPFieldDefinition` or `GPPAttributeDefinition` tag, which contain element parameters.

All the definitions must be contained within the `<GPPModel>` tag.

Element parameters

These are the parameters of the document model elements:

name You assign a name to the text field or document attribute for each definition. The names enable you to restrict a search query to the content of a specific text field or document attribute. Using the above examples, you could search for documents containing the word `structure` in the text field named `Abstract`.

start A boundary string in code page UTF-8 that marks the beginning of the text field or document attribute. There are no rules for specifying boundary strings; they can be any arbitrary UTF-8 string. Here are some examples: start="introduction:", start="note!", start="$$...".

Nonprintable characters and the special XML characters "<" and "&" must be specified using the standard XML escape character ("<" for "<", and "&" for "&").

end Optional. A boundary string in code page UTF-8 that marks the end of the text field or document attribute. If you do not specify an end tag, the next found start tag is assumed to be the end of the field. If no subsequent start tag is found, the field extends to the end of the document, and no further fields are identified.

type The type of document attribute must always be "NUMBER". The parameter does not apply to field definitions.

exclude
YES or NO. A parameter that determines whether the text in a field should be excluded and not indexed. This parameter does not apply to attribute definitions. In the example, the field definition "head" would be excluded, but definition "abstract" included.

Restrictions:

- There must not be two field definitions or attribute definitions having the same start tag. However, a field definition and attribute definition may have the same start tag and end tags.
- A start tag must not be a proper prefix of another. For example, you cannot have a start tag "author" and a start tag "authority".

- Start tags and end tags must not be empty strings.

For information on the Document Type Definitions, see "DTD for document models" on page 253.

For additional restrictions, see "Limitations for text fields and document attributes" on page 256.

What happens when a GPP document is indexed

The general-purpose parser scans the document looking for one of the start boundary strings. When it finds a start boundary string, it parses the subsequent field until it finds the corresponding end boundary string.

The content of the field is then indexed according to the definition term, that is, as a text field or document attribute. If the text field and document attribute have the same start and end boundary strings, the content of the field is indexed as both a text field and a document attribute.

No nesting of fields is allowed; if a new start boundary string is found in a field before the end boundary string has been reached, the new start boundary string is interpreted as normal text.

If no corresponding end boundary string is found, the field is assumed to extend to the end of the document, and an appropriate reason code is reported.

If no end boundary string is specified in the document model, the new start boundary string signals the end of the previous field.

Defining a document model for HTML documents

The HTML parser converts the text to code page UTF-8. It performs HTML tag recognition and classifies them into tag classes:

- Tagged information to be ignored, such as font information
- Tags that provide positional information, such as <p>; for new paragraph
- Tags that provide structural information, such as <Title>

It recognizes all character entity references defined in HTML 4, like "ä" (ä) and resolves them to the corresponding code points in UTF-8.

It recognizes meta tags and parses the meta tag text.

Here is an example of an HTML document:

```
<HTML>
<HEAD>
<META NAME="year" CONTENT="2002">
```

```
<TITLE> The Firm </TITLE>
</HEAD>
<BODY>
<H1>Synopsis</H1>;


<H1>Prologue</H1>;:
:
</BODY>
```

Here is an example of an HTML document model:

```
<?xml version="1.0"?>
<HTMLModel>

 <HTMLFieldDefinition
 name="subtitle"
 tag="title"
 exclude="YES" />

 <HTMLFieldDefinition                - This is the start of text field
 name="header1"
 tag="h1"
 exclude="YES" />                    - This is the end of the text field

 <HTMLAttributeDefinition            - This is the start of the document
 name="year"                                 attribute
 tag="meta"
 meta-qualifier="year"
 type="NUMBER" />                    - This is the end of the document
                                           attribute
 </HTMLModel>
```

The first line, `<?xml version="1.0"?>`, specifies that the document model is written using XML tags. Note that this model is not written for XML format documents.

Each field is defined within a `HTMLFieldDefinition` or `HTMLAttributeDefinition` tag, which contain element parameters.

All the text field definitions must be contained within the `<HTMLModel>` tag.

Element parameters

These are the parameters of the document model elements:

name You assign a name to the text field or document attribute for each definition. The names enable you to restrict a search query to the content of a specific text field or document attribute. Using the above examples, you could search for documents containing the word `firm` in the text field named `subtitle`.

tag Identifies an element whose start and (implied) end tags mark

the text field or document attribute. The text that is inside an element of that name makes up the content of the defined field.

The case of the tag is disregarded.

Using the above examples, the text following any `H1` tag is indexed as being part of the field ″header1″. In which case, ″synopsis″ and ″prologue″ would be indexed.

meta-qualifier

This tag has to be used with the **tag** element. By specifying tag=″meta″, the value of the content that matches the meta-qualifier is extracted.

In the HTML document example, the meta tag has the following elements:

```
<META NAME="Author" CONTENT="J. Grisham">
```

In the document model example, meta-qualifier=″author″. Therefore, the content ″J. Grisham″ is indexed as the value of the string attribute ″author″.

type The type of document attribute must be ″NUMBER″. The parameter does not apply to field definitions.

exclude YES or NO. A parameter that determines whether the text in field definitions should be excluded and not indexed. This parameter does not apply to attribute definitions. In the example, the field definition ″header1″ would be excluded, but definition ″subtitle″ included.

All other text of a document is indexed, but not as part of any field.

For information on the Document Type Definitions, see “DTD for document models” on page 253.

For restrictions, see “Limitations for text fields and document attributes” on page 256.

Defining a document model for XML documents

Net Search Extender does not attempt to detect the code page of an XML document. The CCSID specified during `CREATE INDEX`, or if not specified, the DB2 code page.

Here is an example of an XML document:

Document model

```
<?xml version="1.0"?>
<purchaseOrder orderDate="2001-01-20">
    <shipAddress countryCode="US">                [1]
        <name>Alice Smith</name>                  [2]
        <street>123 Maple Street</street>
        <city>Mill Hill</city>
        <state>CA</state>
        <zip>90999</zip>
    </shipAddress>
    <item partNo="123" quantity="1">
        <name>S&B Lawnmower Type ABC-x</name>     [3]
        <price>239.90</price>
        <shipDate>2001-01-25</shipDate>
    </item>
    <item partNo="987" quantity="1">     [3]
        <name>Multifunction Rake ZYX</name>
        <price>69.90</price>
        <shipDate>2001-01-24</shipDate>
    </item>
</purchaseOrder>
```

Here is an example of an XML document model:

```
<?xml version="1.0"?>
<XMLModel>

<XMLFieldDefinition                    [1]
name="addresses"
locator="/purchaseOrder/shipAddress"
exclude="no" />

<XMLFieldDefinition                   [2]
name="customerName"
locator="//shipAddress/name"
exclude="yes"/>

<XMLAttributeDefinition                   [3]
name="partNumber"
type="NUMBER"
locator="/purchaseOrder//item/@partNo" />

</XMLModel>
```

The first line, `<?xml version="1.0"?>`, specifies that the model is written using XML. Each field is defined within a `XMLFieldDefinition` or `XMLAttributeDefinition` tag, which contains element parameters.

Note that all the text field definitions must be contained within the `<XMLModel>` tag. For restrictions, see "Limitations for text fields and document attributes" on page 256.

Element parameters

These are the parameters of the document model elements:

name You assign a name to the text field or document attribute for each definition. These names enable you to restrict a search query to the content of a specific text field or document attribute.

You can use one of the following variables in a name. The variable is replaced by a string generated from the matching element in the source document.

Variable	Value
$(NAME)	The actual qualified name (QName) of the XML element that matched the XPath.
$(LOCALNAME)	The actual local name (without prefix) of the XML element that matched the XPath.
$(PATH)	The actual absolute path as a sequence of slashes and tags of the XML element that matched the XPath.

type The type of document attribute must be ″NUMBER″. The parameter does not apply to field definitions.

locator

Expressions in the XPath language that select the parts of the source documents to be used as search fields.

When writing a XML Document Model file, the `locator` must be identical to the tags in the XML document, otherwise the queries will not return a result.

These locators are from the example. For further information, see the syntax in “The semantics of locator (XPath) expressions” on page 254.

**`purchaseOrder	salesOrder`**	All `purchaseOrder` elements and `salesOrder` elements
`shipAddress`	All `shipAddress` elements
*	All elements (this is the abbreviation of `child::*` – see the syntax for further information)
name/item	All `item` elements that have a `name` parent
purchaseOrder//item	All `item` elements that have a `purchaseOrder` ancestor
/	The root node
comment()	All comment nodes
processing-instruction()	All processing instructions

attribute::* (or @*)
All attribute nodes

NCName
An XML name not containing colons

QName
An NCName that can be preceded by NCName: (an NCName followed by a colon), like this: `NCName:NCName`

A literal is a string enclosed either in single or double quotes. For an exact definition of terminal tokens see the XML recommendations.

The XPath locators are similar to XML Stylesheet Language Transformation (XSLT) patterns. They comprise exactly the subset of XSLT patterns that do not contain any predicates nor the functions 'id' and 'key' nor the node tests 'text()' and 'node()'.

ignore YES or NO. Use the parameter to make exceptions to the locator.

Sometimes you may want to specify a general locator, such as *, to match the nodes you want to index. But you may also specify that some nodes matching a more specific locator should not be indexed.

To formulate this, include a field definition with the more specific locator for the nodes to be ignored during indexing. You then give this locator a higher priority than the one with the general locator, and specify `ignore="yes"`. This indicates to the indexer that it must not generate field information for the matching nodes.

Note that when such an ignored node is embedded in a field-generating node, the content of the ignored node gets indexed, because it also belongs to the contents of the field-generating node.

priority
A real number between -1 and +1 that specifies the priority to be given to a definition found by a particular locator.

If you do not specify a priority, the default priorities are used:

- Multiple alternatives separated by | are treated as a set of definitions, one for each alternative.
- Locators that match by a single name; that is, locators of either of the following forms have default priority 0:
  - `ChildOrAttributeAxisSpecifier QName`
  - `ChildOrAttributeAxisSpecifier processing-instruction(Literal))`
- Locators of the form `ChildOrAttributeAxisSpecifier NCName:*` have default priority -0.25.

- Other locators of the form `ChildOrAttributeAxisSpecifier NodeTest` have default priority -0.5.
- Any other locator has default priority 0.5.

Note that the more specific the locator is, the higher the default priority. For example, the unspecific locator * gives a low priority to the found definition, whereas a name is a more specific locator and gives a higher priority.

Also note that when a node is matched by more than one locator, you can determine which of the definitions are chosen by assigning priorities to them. The definition with the highest priority is chosen. If two definitions have the same priority, the latest is chosen.

This conflict resolution is the same as that used in XML Stylesheet Language Transformation (XSLT).

exclude

YES or NO. A parameter that determines whether the text in field definitions should be excluded and are not indexed. This parameter does not apply to attribute definitions.

In the example, the field definition ″customerName″ would be excluded, but definition ″addresses″ included.

What happens when an XML document is indexed

The following table shows what is put into the index.

Table 7. Entries in the text index

Field name	Indexed text	
addresses	123 Maple Street Mill Hill CA 90999	[1]
customerName	Alice Smith	[2]
partNumber	123 987	[3]

Note that in [1], where the text of the `shipAddress` element is indexed under field name `addresses`, the `name` element (`Alice Smith`) is not indexed. This is because the `name` element is itself in the document model, and indexed under field name `customerName` [2]. This means that, although embedded elements are allowed in the document model, they are not indexed as part of the embedding text field; they are indexed separately.

The content of fields is determined by the following rules:

- For a field whose locator matches a comment or a processing instruction, the field content is the actual comment text or processing instruction text.
- For a field that matches an XML element or the root node, the field content consists of any text from any embedded element except for elements that are matched by other fields.

The document must contain well-formed XML, but it is not necessary for a DTD to be specified in the XML document. No DTD validation or entity resolution is carried out; Net Search Extender only matches the XML document against the document model.

For information on the Document Type Definitions, see "DTD for document models" on page 253.

For restrictions, see "Limitations for text fields and document attributes" on page 256.

Defining a document model for Outside-In filtered documents

Document models for the Outside-In format are very similar to HTML document models in that they allow you to map structural elements identified by a given set of tags to NSE text fields and document attributes. Assume you have a set of Microsoft Word documents and want to index the document properties ″title″, ″subject″, and ″keyword″ as fields, and the document properties ″author″ and ″category″ as document attributes. The following example for an Outside-In document model will achieve this mapping:

```
<?xml version="1.0"?>
<INSOModel>

<INSOFieldDefinition
name="title"
tag="SCCCA_TITLE"/>

<INSOFieldDefinition
name="title"
tag="SCCCA_SUBJECT"/>

<INSOFieldDefinition
name="title"
tag="SCCCA_KEYWORDS"/>

<INSOAttributeDefinition
name="author"
tag="SCCCA_AUTHOR"
type="STRING"/>

<INSOAttributeDefinition
name="category"
```

```
tag="SCCCA_CATEGORY"
type="STRING"/>

</INSOModel>
```

Element parameters

These are the parameters of the document model elements:

name A name that you assign to the text field or document attribute. You assign a field name to each field definition, and an attribute name to each attribute definition. These names are the means by which a query can restrict search to the content of a certain text field and can search for documents having a certain attribute.

tag Identifies a tag whose begin and end or implied-end elements mark the text field or the document attribute. The text that is inside an element of that name makes up the content of the defined field or attribute. The case of the tag is disregarded. Possible values are described below.

type The type of document attribute can be either ″NUMBER″, ″DATE″, or ″STRING″. This parameter does not apply to field definitions.

exclude
″YES″or ″NO″. A parameter that determines whether the text in field definitions should be excluded. If a field definition has the parameter exclude=″YES″, then those text field are not indexed. This parameter does not apply to attribute definitions.

Outside-In document models consist of field and/or attribute definitions that each define a name and a tag. For attribute definitions a type is also required, whereas field definitions have an optional ″exclude″ flag. As with HTML models, the name attribute of such a definition defines the name of the Net Search Extender field or attribute that the document part is to be mapped on. This can be an arbitrary UTF-8 text string. For additional information, see the Outside-In Content Access Specification, Version 7.5.

For a list of possible values for the tag attribute relating to the Outside-In begin, end and document property tags, see “Outside-In tag attribute values” on page 257 for further information.

What happens when an Outside-In document is indexed

By default, all the text is indexed as not belonging to any field. Whenever a begin tag that appears in the stream of text matches a definition item in the document model that is currently active, the text between the begin tag and its corresponding end tag is treated according to that definition term. For example, as an indexed field, an excluded field, or as an attribute or both.

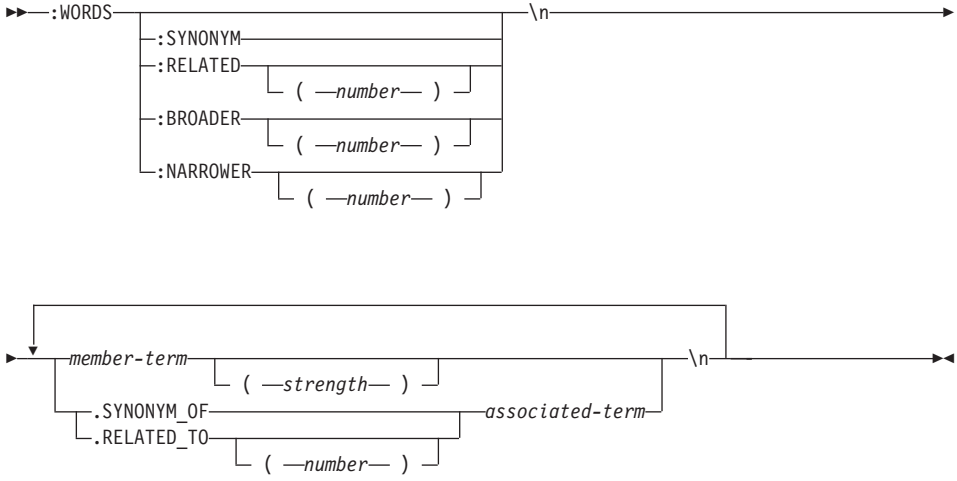
If no matching definition exists, the begin tag and its corresponding end tag are ignored.

As Outside-In filters automatically recognize the format and code page of the document, the CCSID specification has no effect. If the Outside-In filters are unable to determine the correct format and code page, the document is treated as a unicode (UTF-16) file. This corresponds to the Outside-In specific document type FI_UNICODE.

Chapter 18. Thesaurus support

Here is the syntax of each definition group:

Syntax of a thesaurus definition



Note that \n is not part of the syntax, but represents the end of a line in the thesaurus definition file.

You can insert comment lines in a thesaurus definition file like this:

```
# my comment text
```

:WORDS
A keyword that begins a group of related words.

:SYNONYM,
:RELATED [(*number*)],
:BROADER [(*number*)],
:NARROWER [(*number*)]
A relation name.

Relation names consist of a relation type and a number. If the number is omitted, zero is assumed, which is the system-provided relation name. :SYNONYM is always the system-provided relation name.

Relation names that begin with a colon, such as :SYNONYM, precede a list of words that are related to each other by the same relation. For example:

© Copyright IBM Corp. 1995, 2003

```
:WORDS
 :SYNONYM
   air steward
   cabin staff member
   flight attendant
```

member-term

A term to be included in the thesaurus dictionary.

- Maximum length is 64 bytes (42 bytes for code page UTF-8).
- Single-byte characters and double-byte characters of the same letter are regarded as the same.
- Uppercase and lowercase characters are not distinguished.
- A term can contain a blank character.
- The single-byte character period ″.″ or colon ″:″ cannot be used.

This parameter can be useful if you do not want a thesaurus lookup to include words that have a weak relation to the looked up term. Strength is a numerical value from 1 to 100. The default value is 100.

.SYNONYM_OF,
.RELATED_TO [(*number*)]

A relation name. Relation names consist of a relation type and a number. If the number is omitted, zero is assumed, which is the system-provided relation name. The relation name .SYNONYM is always the system-provided relation name.

Relation names that begin with a period, such as .SYNONYM_OF, define the relation between one word and another. For example:

```
:WORDS
   air steward
 .SYNONYM_OF cabin staff member
 .SYNONYM_OF flight attendant
```

The optional *number* identifies a user-defined relation. This must be a unique number from the whole thesaurus definition file (currently 1 to 128). For example: `RELATED_TO(42)`.

If you want to use symbolic names for thesaurus relations in your application instead of the relation name and number, your application must handle the name-to-number mapping. For example, if you define the relation `opposite_of` as RELATED_TO(1), your application must map this name to the internal relation name RELATED_TO(1).

associated-term

Each associated term must be preceded by the relation name. The associated term is related to each member term with respect to the specified relation. If all member terms are related to each other, this can be specified using a member relation.

- Maximum length is 64 bytes (42 bytes for code page UTF-8).

- Single-byte characters and double-byte characters of the same letter are regarded as the same.
- Uppercase and lowercase characters are not distinguished.
- A term can contain a blank character.
- The single-byte character period ″.″ or colon ″:″ cannot be used.

Here is an example of an associated term:

```
:WORDS:SYNONYM
  reject
  decline
    RELATED_TO(1) accept
```

Part 3. Appendixes

© Copyright IBM Corp. 1995, 2003

Appendix A. Migration

DB2 Net Search Extender Version 8.1.x has been extensively altered to include the search interface and functions of DB2 Text Information Extender Version 7.2. Accordingly, there are three migration options:

- Moving from Text Information Extender Version 7.2 to Net Search Extender Version 8.1.x
- Moving from Net Search Extender Version 7.2 to Net Search Extender Version 8.1.x
- Moving from Net Search Extender Version 8.1 to Net Search Extender Version 8.1.x

Note

For the latest migration information, check the `release.txt` file on the CD-ROM and DB2 Net Search Extender Web site.

Moving from Net Search Extender Version 8.1 to Net Search Extender Version 8.1.x

There are no prior steps necessary when moving from Net Search Extender Version 8.1 to Net Search Extender Version 8.1.x. Therefore, you can remove your old installation and install the latest version of Net Search Extender.

Migrate your DB2 instance from Version 8.1 to Version 8.1.x using `db2iupdt` for both UNIX and Windows. Then migrate all your enabled databases of this instance using the new `db2extmdb` tool. The caller must be the instance owner and the syntax is:

```
db2extmdb <database name>
```

While the migration is running, no changes to user tables with text indexes should be done.

Notes and recommendations

The migration steps are logged in the following file:

```
<os-dependent>/sqllib/db2ext/db2extm <database-name>.log
```

Before calling the `db2extmdb` program, the user should take a backup of all index directories and the database.

© Copyright IBM Corp. 1995, 2003

Moving from Net Search Extender Version 7.2 to Net Search Extender Version 8.1.x

If you have been using Net Search Extender Version 7.2 and do not require all the new features, simply continue using the old interfaces by installing the Net Search Extender Version 7.2 compatibility interfaces available on the CD-ROM, or by download. See the `release.txt` for further information. To use the new Net Search Extender functionality, you need to change your administration scripts to correspond to the altered administration syntax and change your search syntax to the new search interfaces.

Note that there is no automatic way to migrate from Net Search Extender Version 7.2 to Net Search Extender Version 8.1.x.

Also note that these compatibility interfaces are deprecated and will not be available in future releases.

Moving from Text Information Extender Version 7.2 to Net Search Extender Version 8.1.x

There are no prior steps necessary when moving from Text Information Extender Version 7.2 to Net Search Extender Version 8.1.x. Therefore, you can remove your old installation and install the latest version of Net Search Extender.

Migrate your DB2 instance from Version 7.2 to Version 8.1.x using `db2iupdt` for both UNIX and Windows. Then migrate all your enabled databases of this instance using the new `db2extmdb` tool. The caller must be the instance owner and the syntax is:

```
db2extmdb <database name>
```

During the first step, the program gathers all db2ext administration information relevant for migration into a new table, called DB2EXT.TMIGRATION. In the table, each text index is represented as a single one row. The migration information table will persist until the database has been migrated successfully and should not be dropped by the user.

If an error occurs, fix the error and call `db2extmdb` again.

During migration of the text index, different processing occurs. This depends on the state of the 'log-table'. If the log table is empty, the index will be migrated, which should be the fastest way. If the log table is not empty, a consistent state with the database can not be ensured and the index needs to be re-created. This process can take a considerable amount of time.

While the migration is running, no changes to user tables with text indexes should be done.

Recommendation

Before calling the `db2extmdb` program, the user should take a backup of all index directories, the database, and has verified that all document model files used for creation of the version 7.2 text indexes still exist and are accessible for reading.

After DB2 instance migration, the previous `sqllib` directory is renamed to `sqllib_v71`. If you have text indexes stored on the default index directory, move the `sqllib_v71/db2ext/indexes` directory to the new sqllib directory, `sqllib/db2ext/indexes`.

Notes and recommendations

The migration steps are logged in the following file:

`<os-dependent>/sqllib/db2ext/db2extm <database-name>.log`

For HP-UX or Linux, this step is not necessary as Text Information Extender is not available on these platforms.

Also note that DB2 Text Information Extender Version 7.2 is no longer available.

Migration

Appendix B. Using large amounts of memory

Using the cache for a stored procedure search requires a large amount of memory and different memory requirements for the following platforms:

- AIX
- Windows
- The Solaris Operating Environment
- Linux
- HP-UX

AIX (32-bit and 64-bit)

Configuring the system limits:

- Check the system limits using the command `ulimit -a`
- If there are values other than "unlimited", use the following steps:
  - Log on as root.
  - Back up the file `/etc/security/limits` and then edit the file to raise the hard limits.
  - Set all values to the "unlimited" (value -1) for the DB2 instance owner used.

Configuring the shared memory limits:

- On AIX, there is no need to configure the shared memory limits.

Configuring the swap space:

- Obtain the system RAM size using the command `lsattr -E -l sys0`
- Obtain the size of the swap space using the `lsps -a` command.
- Set the swap space size to at least 1.5 - 2 times of either the RAM amount of your system, or use the `MAXIMUM CACHE SIZE` parameter you provide in the `CREATE INDEX` command. Use the SMIT utility to select the larger number.

  Note that the maximum cache size limit on AIX is about 1536 MB (1.5 GB = 1610612736 bytes).

Windows (32-bit)

Adjusting the size of the paging file:

© Copyright IBM Corp. 1995, 2003

- Set the Windows virtual-memory paging file size to at least 1.5 - 2 times of either the RAM amount of your system, or use the `MAXIMUM CACHE SIZE` parameter you provide in the `CREATE INDEX` command. Select the larger number. See the Windows documentation for information on changing the size of the paging file.

  Note that the maximum cache size limit on Windows is about 1000 MB (1 GB = 1073741824 bytes).

The Solaris Operating Environment (32-bit and 64-bit)

Configuring the system limits:

- Check the system limits using the command: `ulimit -a`
- Then use the following steps:
  - Log on as root.
  - Back up the file `/etc/system` and then edit the file to raise the hard limits.
  - Add or check that the following lines are set to at least the minimum values shown:

    ```
    rlim_fd_cur -> Default 64, recommended >= 1024
    rlim_fd_cur_max -> Default 1024, recommended >= 4096
    ```

Configuring the shared memory limits:

- Check current settings using command `sysdef -i`
- Edit the file /etc/system to set the shared memory size limit using: `set shmsys:shminfo_shmmax=0xffffffff`

  You may also have to increase the following parameter values:

  `set shmsys:shminfo_shmmni=512`

  `set shmsys:shminfo_shmseg=128` and then reboot the system.

Configuring the swap space:

- Obtain the system RAM size using the command `/usr/sbin/prtconf`
- Obtain the size of the swap space using the `swap -l` command.
- Set the swap space size to at least 1.5 - 2 times of either the RAM amount in your system, or use the `MAXIMUM CACHE SIZE` parameter you provide in the `CREATE INDEX` command. Select the larger number.

  Refer to the Solaris system documentation for information on how to add swap space.

  Note that the maximum cache size limit on Solaris is about 2000 MB (2 GB = 2147483647 bytes).

Linux (32-bit)

Check DB2 documentation about specific kernel parameters.

To see your current shared resouce limits use `ipcs -l`. To check the system limits, use the `ulimit -a` command.

HP-UX (32-bit and 64-bit)

Set the `shmmax` parameter to 134217728 or 90% of the physical memory (in bytes), whichever is higher.

For example, if you have 196 MB of physical memory in your system, set `shmmax` to 184968806 (196*1024*1024*0.9).

Appendix C. Net Search Extender information catalogs

DB2 Net Search Extender stores important information about defaults, configurations, text indexes, and formats in catalog tables. To view this information, you can query some views on tables.

The following views and tables reflect the current configuration of your system:

- Database level information views:
  - `db2ext.dbdefaults`
  - `db2ext.proxyinformation` table
- Index level information views:
  - `db2ext.textindexes`
  - `db2ext.textindexformats`
  - `db2ext.indexconfiguration`

  Note that for compatibility reasons, DB2 Text Information Extender views are still available. These are `db2ext.textcolumns`, `db2ext.models`, and `db2ext.formats`.
- Table views for a text index:
  - Event view
  - Log table view

Views for database-level information

The view `db2ext.dbdefaults` displays all the default values for the database.

The defaults on the database level cannot be changed and are available as attribute-value pairs in this view:

```
db2ext.dbdefaults

db2 select DEFAULTNAME, DEFAULTVALUE from DB2EXT.DBDEFAULTS
```

Table 8. db2ext.dbdefaults view

Attribute	Default Value	Notes
CCSID	CCSID of database	Default CCSID for documents. This is applied if no CCSID is specified in the `CREATE INDEX` command.

© Copyright IBM Corp. 1995, 2003

Table 8. db2ext.dbdefaults view (continued)

Attribute	Default Value	Notes
FORMAT	TEXT	Document default format. This is applied if no format is specified in the `CREATE INDEX` command.
INDEXDIRECTORY	See the path name under **Notes**	Directory for full-text index files. This is applied if no index directory is specified in the `CREATE INDEX` command. The path name is: `$DB2EXT_INSTOWNERHOMEDIR/sqllib/db2ext/indexes`
LANGUAGE	EN_US	The document language.
MODELCCSID	CCSID of database	CCSID of document model files.
UPDATECOMMITCOUNT	0	The number of changes processed in one transaction during an update.
CLEARCOMMITCOUNT	0	The number of changes processed in one transaction during an CLEAR INDEX command.
UPDATEFREQUENCY	NONE	When to check for updates in new indexes.
UPDATEMINIMUM	1	The minimum number of changes before update is executed.
WORKDIRECTORY	See the path name under **Notes**	Directory for index temporary files. The path name is: `<os_dependent>/sqllib/db2ext/indexes`
CACHEDIRECTORY	See the path name under **Notes**	Default directory for `PERSISTENT CACHE` option of the `CREATE INDEX` command. The path name is: `<os_dependent>/sqllib/ db2ext/memory`
PCTFREE	50	The percentage of the cache left free for future inserts.
USERPERSISTENTCACHE	1	Use the persistent cache.
AUTOMATICREORG	1	The `REORGANIZE` option in the `CREATE INDEX` command. This means automatic reorganization.
TREATNUMBERSASWORDS	0	Do not interpret sequences as separate words, even if they are adjacent characters. For example, the 0 default means that tea42at5 is considered as one word.

Table 8. db2ext.dbdefaults view (continued)

Attribute	Default Value	Notes
INDEXSTOPWORDS	1	Ignore stopwords during indexing.
VERSION		NSE V8.1.2 Current version number of Net Search Extender.
UPDATEDELAY	0	Only log entries are used for incremental updates with a timestamp. If the log entry is older than the current timestamp, this is an updatedelay. This should only be used for long running transactions on the user table during an update command to avoid lost updates. Note that with log entries and incremental updates, no capture table is used.

db2ext.proxyinformation table

If you are using datalinks and want to access files using a proxy server, you must specify proxy information in the `db2ext.proxyinformation` table.

Table 9. db2ext.proxyinformation view

Attribute	Type	Notes
PROXYHOST	VARCHAR(254)	The host name of the proxy server.
PROXYPORT	VARCHAR(6)	The used port for the proxy server.
PROXYTYPE	VARCHAR(10)	The type of the proxy server (either PROXY or SOCKS).
PROXYTIMEOUT	INTEGER	The timeout in seconds.

You can insert a maximum of one row. If you have a proxy server on port 123, you can insert a row with the following SQL statement:

```
db2 insert into db2ext.proxyinformation values ('proxy1', '123','PROXY', 10)
```

Views for index-level information

You can query information at an index-level using the following DB2 Net Search Extender views:

- `db2ext.textindexes`
- `db2ext.textindexformats`
- `db2ext.indexconfiguration`
- `<index eventview name schema>.<index eventview name>`

For backward compatibility reasons, DB2 Text Information Extender views db2ext.textcolumns, db2ext.formats, and db2ext.models are still supported, but deprecated.

Note that in the db2ext.textcolumns view the OPERATION, OPERATIONBEGIN, and OPERATIONEND columns are no longer supported.

db2ext.textindexes view

Each database enabled for DB2 Net Search Extender contains a db2ext.textindexes view. This contains information on settings, statistics, and defaults for the created text indexes in this database.

When you create a text index, new entries are created in db2ext.textindexes. When you drop the text indexes, these entries are deleted.

You can query the view to obtain information about the indexes. This is an example using the index schema:

```
db2 "select COLNAME from DB2EXT.TEXTINDEXES where INDSCHEMA='myschema'
     and INDNAME='myindex'"
```

Note, however, that you cannot modify the view using normal SQL data manipulation commands, or explicitly create or drop the catalog view. Additional contents of the view are found in the following table.

Also note that the replication parameters are not included in this view.

Table 10. db2ext.textindexes view

Attribute	Type	Notes
INDSCHEMA	VARCHAR(128)	Schema name of the text index.
INDNAME	VARCHAR(128)	Name of the text index.
TABSCHEMA	VARCHAR(128)	The table name of the schema for base tables, nicknames, and views.
TABNAME	VARCHAR(128)	Alias name the index was created on.
COLNAME	VARCHAR(128)	Column the index was created on.
CCSID	INTEGER	Document CCSID for this index.
LANGUAGE	VARCHAR(5)	Document language for this index.
FUNCTIONSCHEMA	VARCHAR(128)	Schema of the column mapping function.
FUNCTIONNAME	VARCHAR(18)	Name of the column mapping function.
INDEXDIRECTORY	VARCHAR(256)	Directory for full-text index files.
WORKDIRECTORY	VARCHAR(256)	Directory for index temporary files.

Table 10. db2ext.textindexes view (continued)

Attribute	Type	Notes
CACHEDIRECTORY	VARCHAR(256)	Directory for persistent cache (if persistentcache=1).
UPDATEFREQUENCY	VARCHAR(300)	Trigger criterion for applying automatic updates to this index.
UPDATEMINIMUM	INTEGER	Minimum number of documents that must be changed before an update is performed.
EVENTVIEWSCHEMA	VARCHAR(128)	Schema of the event view created for this index.
EVENTVIEWNAME	VARCHAR(128)	Name of the event view created for this index.
LOGVIEWSCHEMA	VARCHAR(128)	Schema of the log view created for an index.
LOGVIEWNAME	VARCHAR(128)	Name of the log view created for an index (important for incremental update on views).
COMMITCOUNT	INTEGER	Default for commitcount update.
NUMBER_DOCS	INTEGER	Total number of documents currently in the index. Note that during an index update, this value is only updated if the `commitcount` is set.
REORG_SUGGESTED	INTEGER	Indicates if performance can be improved by running UPDATE INDEX REORGANIZE. This parameter is only true (1) if at least one of the nodes has an index reorganization suggested.
REORGAUTOMATIC	INTEGER	1, if the index gets automatically reorganized during the update operation.
RECREATEONUPDATE	INTEGER	1, if the index gets automatically reorganized during the update operation.
CREATIONTIME	TIMESTAMP	Time of index creation.
UPDATETIME	TIMESTAMP	Time of last update. If UPDATE TIME is equal to CREATION TIME, then no update has been processed.
PERSISTENTCACHE	INTEGER	1, if persistent cache is used.
MAXIMUMCACHESIZE	INTEGER	Maximum size of cache.

Table 10. db2ext.textindexes view (continued)

Attribute	Type	Notes
PCTFREE	INTEGER	Percentage of cache left free for future inserts.
CACHETABLE	VARCHAR(32000)	Column expression list for the CACHE TABLE.
RESULTORDER	VARCHAR(32000)	SQL-order-by for INITIAL RESULT ORDER.
ATTRIBUTES	VARCHAR(32000)	Column expression list for ATTRIBUTES.
VIEWKEYCOLUMNS	VARCHAR(32000)	Key columns for index on view.

db2ext.indexconfiguration view

Index configuration parameters are available in the `db2ext.indexconfiguration` view. The view is available through normal SQL query facilities. This is an example using the index name:

```
db2 "select VALUE from DB2EXT.INDEXCONFIGURATION where INDSCHEMA='myschema'
     and INDNAME='myindex' and PARAMETER ='INDEXSTOPWORDS'"
```

Additional contents of the view are found in the following tables.

Table 11. db2ext.indexconfiguration view

Attribute	Type	Notes
INDSCHEMA	VARCHAR(128)	Schema name of the index.
INDNAME	VARCHAR(128)	Name of the index.
PARAMETER	VARCHAR(30)	Type of parameter.
VALUE	VARCHAR(512	Value of the parameter.

For the `PARAMETER` and `VALUE` attributes, there are several values available.

Table 12. db2ext.indexconfiguration view

Attribute and values	Attribute and values
PARAMETER	VALUE
- TREATNUMBERASWORDS	- 0 or 1
- INDEXSTOPWORDS	- 0 or 1
- UPDATEDELAY	- seconds >= 0

For further information, see the `CONFIGURATION` option of the `CREATE INDEX` command.

db2ext.textindexformats view

Format and model information for indexes is available in the db2ext.textindexformats view. This is an example using the index name:

```
db2 "select FORMAT from DB2EXT.TEXTINDEXFORMATS where INDSCHEMA='myschema'
     and INDNAME='myindex'"
```

Additional contents of the view are found in the following table.

Table 13. db2ext.textindexformats view

Attribute	Type	Notes
INDSCHEMA	VARCHAR(128)	Schema name for the index (used as prefix for indexname and schemaname in the log table).
INDNAME	VARCHAR(128)	Index name specified in CREATE INDEX command.
FORMAT	VARCHAR(30)	The model is bound to this format.
MODELNAME	VARCHAR(30)	The name of a document model.
MODELFILE	VARCHAR(256)	File containing the model definition.
MODELCCSID	INTEGER	CCSID of MODELFILE.
DEFAULT	INTEGER	Currently 1, as multiple formats in an index are not currently supported.

Table views for a text index

You can query information at an index level using these DB2 Net Search Extender views:

- Event view
- Log table view

Event view

This view allows you to get information on indexing status and error events, and when problems occur during indexing, for example, a document not being found. These index update events are then written to the index's event table.

The schema and name are stored in the db2ext.textindexes view. To get the name of the event view, use the following example:

```
db2 "select EVENTVIEWSCHEMA, EVENTVIEWNAME from DB2EXT.TEXTINDEXES
       where INDSCHEMA = 'myschema' and INDNAME = 'myindex'
```

Views for index-level information

The event view for an index consists of the following columns.

Table 14. The event view

Attribute	Type	Notes
OPERATION	INTEGER	The operation on the user table is be reflected in full-text index (insert = 0/ update = 1/ delete = 2). When using a replication capture table, update operations are split into a delete and an insert operation. In this case, an insert operation in the event table can be either from an insert, or an update operation on the source table the index was created on.
TIME	TIMESTAMP	Timestamp of event entry creation.
REASON	INTEGER	Reason code. For a list of reason codes, see Appendix I, "Text Search Engine reason codes", on page 263.
SEVERITY	INTEGER	The severity of the table entry. For example, 1 is for information purposes, 4 indicates a warning, and 8 means a table entry error.
MESSAGE	VARCHAR(1024)	Additional text information.
KEY1, ... KEY14	Depends on the user table	First primary key column of user table to the last primary key column (a maximum of 14).
PARTITION	INTEGER	The database partition number on which this error occurs. In a non-distributed environment, this is 0.

The events can be cleared by the `DB2TEXT CLEAR EVENTS` command, see "CLEAR EVENTS" on page 123 for further information.

Note

Informational events, such as starting, committing, and finishing update processing are also available in this view.

In this case, `Key1`, `... Key14` and `OPERATION` all have NULL values.

In the case of indexes on views, the PK01, ..., PK14 columns relate to the columns specified in the KEY COLUMNS clause of the `CREATE INDEX` command.

Log tables, views, and nicknames

The purpose of the log table is to store the change operations on the user table or view that require synchronization with the external full-text index.

For indexes created on regular tables or nickname tables, there are triggers created on the user table to feed change information into the log table. However, if replication capture tables are used, no log table is created and the replication capture table is used instead.

For log tables, the update command reads the entries and deletes them after successful synchronization.

However, in the case of indexes on views, triggers cannot fill the log table. As you can update the view, the user is responsible for this task.

Table 15. The log table view

Attribute	Type	Notes
OPERATION	INTEGER	The type of change on the user table that requires index synchronization: (0 = insert, 1 = update, 2 = delete).
TIME	TIMESTAMP	The timestamp for the creation of a row in this table.
PK01 ... PKnm	Same as user table	In case of errors, the column where the problem occurred. They are a copy of the primary key columns of the user table or the equivalent key columns in case of an index on a view.

The user who creates the table is able to select, update, insert, and delete this view.

If you specify a replication capture table in the create index command, no log table is created and the replication capture table is used instead. The replication capture table must contain the following columns:

Table 16. The replication capture table

Attribute	Type	Notes
IBMSNAP_OPERATION	INTEGER	The type of change on the CD or CCD table that requires index synchronization: (I = insert, U= update, D= delete). When using a replication capture table, update operations are split into a delete and an insert operation. In this case, an insert operation in the event table can be either from an insert, or an update operation on the source table the index was created on.
IBMSNAP_COMMITSEQ	CHAR	Maps to the corresponding column of the CD or CCD table.
IBMSNAP_INTENTSEQ	CHAR	Maps to the corresponding column of the CD or CCD table.
PK01 ... PKnm	Same as user table	In case of errors, the column where the problem occurred. They are the primary key columns of the user table.

The user who defines the table is able to select, update, insert, and delete with the grant option.

Appendix D. Supported CCSIDs

The following CCSIDs are supported on DB2 Net Search Extender.

CCSIDs

37	USA/Canada - CECP
273	Germany F.R./Austria - CECP
274	Old Belgium Code Page
277	Denmark, Norway - CECP
278	Finland, Sweden - CECP
280	Italy - CECP
284	Spain/Latin America - CECP
285	United Kingdom - CECP
290	Japanese (Katakana) Extended
297	France - CECP
301	Japan DB PC
367	ASCII
420	Arabic Bilingual
423	Greece - 183
424	Israel (Hebrew)
437	US English
500	International #5
737	MS DOS Greek
806	Hindi
813	Greek
819	Latin-1
833	Korean Extended
836	Simplified Chinese Extended
838	Thai with Low Tone Marks & Ancient Characters

© Copyright IBM Corp. 1995, 2003

848	PC, Cyrillic, Ukrainian with Euro
850	Latin-1
852	Latin-2
855	Bulgarian
857	Turkish
858	Personal Computer - Multilingual with Euro
860	Portuguese
862	Hebrew
863	Canadian
864	Arabic
866	Russian
867	Israel - Personal Computer
869	Greek
870	Latin 2 - EBCDIC Multilingual
871	Iceland - CECP
872	Cyrillic - PC with Euro
874	Thai
875	Greece
891	Korea - Personal Computer
895	Japan 7-Bit Latin
901	PC Baltic Multi with Euro
902	8-bit Estonia with Euro
912	Latin-2
915	Russian
916	Hebrew
920	Turkish
921	Latvian, Estonian
922	Estonian
923	Latin 9
924	Latin 9 EBCDIC
927	Taiwan PC

930	Japan EBCDIC
932	Japanese, combined SBCS/DBCS
933	Korean
935	Chinese (simplified)
937	Chinese (traditional)
938	Taiwan PC
939	Japanese
941	Japan OPEN
942	Japanese, combined SBCS/DBCS
943	Japanese, combined SBCS/DBCS
948	Chinese (traditional), combined SBCS/DBCS
949	Korean
950	Chinese (traditional), combined SBCS/DBCS
954	Japanese
964	Chinese (traditional), combined SBCS/DBCS
970	Korean
1025	Cyrillic, Multilingual
1026	Latin #5 - Turkey
1027	Japanese (Latin) Extended
1040	Korean Extended - Personal Computer
1041	Japanese Extended - Personal Computer
1043	Traditional Chinese Extended - PC
1046	Arabic
1047	Latin 1/Open Systems
1051	H-P Emulation, Roman 8
1088	Revised Korean - Personal Computer
1089	Arabic
1112	Baltic - Multilingual, EBCDIC
1115	People’s Republic of China (PRC)-PC
1122	Estonia, EBCDIC
1123	Cyrillic, Ukraine

1124	Ukrainian
1125	Ukrainian
1131	Vietnamese
1137	Devanagari EBCDIC
1140	USA, Canada, etc. ECECP
1141	Austria, Germany ECECP
1142	Denmark, Norway ECECP
1143	Finland, Sweden ECECP
1144	Italy ECECP
1145	Spain, Latin America (Spanish) ECECP
1146	UK ECECP
1147	France ECECP
1148	International ECECP
1149	Iceland ECECP
1153	EBCDIC Latin 2 Multilingual with Euro
1154	EBCDIC Cyrillic, Multilingual with Euro
1155	EBCDIC Turkey with Euro
1156	EBCDIC Baltic Multi with Euro
1157	EBCDIC Estonia with Euro
1158	EBCDIC Cyrillic, Ukraine with Euro
1160	Thai with Low Tone Marks & Ancient Characters
1161	Thai with Low Tone Marks & Ancient Chars - PC
1162	Thai MS Windows
1163	Vietnamese ISO-8 with Euro
1164	Vietnamese EBCDIC with Euro
1200	UCS2
1208	UTF8
1250	Latin-2, Belorussian
1251	Russian
1252	Latin-1
1253	Czech

1254	Turkish
1255	Hebrew
1256	Arabic
1258	Vietnamese
1275	Apple, Latin 1
1280	Apple Greek
1281	Apple Turkish
1282	Apple Central European
1283	Apple Cyrillic
1351	Japan OPEN
1363	Korean
1364	Korean
1381	Chinese (simplified), combined SBCS/DBCS
1383	Chinese (simplified), combined SBCS/DBCS
1386	Chinese (simplified), combined SBCS/DBCS
1388	Chinese (simplified), combined SBCS/DBCS
1390	Japanese
1392	China GB18030
1394	Shift JIS X0213
1399	Japan EBCDIC
4909	Greece/Latin ASCII
4930	Korea DB EBCDIC
4933	China EBCDIC
4971	Greece EBCDIC
5026	Japanese Katakana
5035	Japanese Latin
5039	Japanese, combined SBCS/DBCS
5210	China SB PC
5346	Windows Latin-2
5347	Windows Cyrillic
5348	Windows Latin-1

5349	Windows Greece
5350	Windows Turkey
5351	Windows Hebrew with Euro
5352	Windows Arabic
5353	Windows Baltic
5354	Vietnamese
9044	Latin-2 PC
9048	Hebrew PC
9049	Turkey PC
9061	Greece PC
9238	Arabic - PC
12712	Hebrew EBCDIC
13121	Korea SB EBCDIC
13488	UCS2
17248	Arabic PC
17584	UCS-2
18030	Chinese character standard
21427	Taiwan BIG-5
33722	IBMeucJP
61955	

Appendix E. Supported languages

These are the language parameters that you can specify in DB2 Net Search Extender. In Net Search Extender, the only significant language-specific processing is done on documents written in the Thai language, or if requested, during stopword processing. See “Stopwords” on page 261 for more information.

AR_AA	Arabic/Arabic Speaking
BE_BY	Belorussian/Belarus
BG_BG	Bulgarian/Bulgaria
CA_ES	Catalan/Spain
CS_CZ	Czech/Czech Republic
DA_DK	Danish/Denmark
DE_CH	German/Switzerland
DE_DE	German/Germany
EL_GR	Greek/Greece
EN_AU	English/Australia
EN_BE	English/Belgium
EN_GB	English/U.K.
EN_US	English/U.S.
EN_ZA	English/South Africa
ES_ES	Spanish/Spain
ET_EE	Estonian/Estonia
FI_FI	Finnish/Finland
FR_BE	French/Belgium
FR_CA	French/Canada
FR_CH	French/Switzerland
FR_FR	French/France
HE_IL	Hebrew/Israel
HI_IN	Hindi/India
HR_HR	Croatian/Croatia

© Copyright IBM Corp. 1995, 2003

HU_HU	Hungarian/Hungary
ID_ID	Indonesian/Indonesia
IT_CH	Italian/Switzerland
IW_IL	Hebrew/Israel
IT_IT	Italian/Italy
JA_JP	Japanese/Japan
KO_KR	Korean/Korea
LT_LT	Lithuanian/Lithuania
LV_LV	Latvian/Latvia
MK_MK	Macedonian/FYR Macedonia
MS_MY	Malay/Malaysia
NB_NO	Norwegian Bokmal/Norway
NL_BE	Dutch/Belgium
NL_NL	Dutch/Netherlands
NN_NO	Norwegian Nynorsk/Norway
NO_NO	Norwegian/Norway
PT_BR	Portuguese/Brazil
PL_PL	Polish/Poland
PT_PT	Portuguese/Portugal
RO_RO	Romanian/Romania
RU_RU	Russian/Russia
SH_SP	Serbian (Latin)/Serbia
SK_SK	Slovak/Slovakia
SL_SI	Slovenian/Slovenia
SQ_AL	Albanian/Albania
SR_SP	Serbian (Cyrillic)/Serbia
SV_SE	Swedish/Sweden
TA_IN	Tamil/India
TE_IN	Telugu/India
TH_TH	Thai/Thailand
TR_TR	Turkish/Turkey

UK_UA	Ukrainian/Ukraine
VI_VN	Vietnamese/Vietnam
ZH_CN	Chinese/PRC
ZH_TW	Chinese/Taiwan

Languages

Appendix F. Net Search Extender messages

DB2 Net Search Extender provides the following message types:

- Information and warning messages
- Error messages

Note that the SQL states returned from the search function are 38600 plus the CTE error number.

Information and warning messages

CTE0001 Operation completed successfully.

CTE0002 The update and locking services are up and running.

CTE0003 Index update started.

CTE0004 Index update ended.

CTE0005 Index update commit: ″%1″,″%2″,″%3″ documents inserted, updated, and/or deleted successfully.

CTE0006 Problem accessing text index. Check `db2diag.log` for details.

CTE0007 The section ″%1″ does not occur in any of the documents or is not a valid document model section name.

CTE0008 Index reorganization started.

CTE0009 Index reorganization ended.

CTE0010 The attribute ″%1″ is not valid.

CTE0011 Cache activation started.

CTE0012 Cache activation ended.

CTE0013 Persistent cache was removed.

CTE0014 Cache deactivated.

Error messages

CTE0100 A DB2 operation failed. DB2 information: ″%2″ ″%4″.

Explanation: A DB2 error occurred that does not allow further processing.

What to do: For more detailed information on this DB2 error, use `db2 ? SQLxxx`.

© Copyright IBM Corp. 1995, 2003

CTE0101 A search engine operation failed. Reason code: ″%2″, ″%3″, ″%4″, ″%5″, ″%6″ .

Explanation: A Search Engine error occurred that does not allow further processing.

What to do: For more detailed information, see the Search Engine reason code descriptions.

CTE0102 A general system function failed. Error: ″%2″.

Explanation: A system error occurred that does not allow further processing.

What to do: Additional information can be found on UNIX in the `errno.h` header file.

CTE0103 An internal error occurred. Location: ″%1″, ″%2″.

Explanation: An internal processing error that does not allow further processing. Try to start and stop the update and locking services, as well as DB2.

What to do: If the error persists, start a trace and also check the `db2diag.log`.

CTE0104 Memory allocation error (search engine).

Explanation: The system has run out of memory.

What to do: Increase the available memory size for the instance owner, or stop other processes running parallel.

CTE0105 Memory allocation error.

Explanation: The system has run out of memory.

What to do: Increase the available memory size for the user, or stop other processes running parallel.

CTE0106 Table ″%1″.″%2″ has no primary key.

Explanation: You tried to create an index on a table that does not have a primary key.

What to do: Call the `db2 alter table` to ensure the existence of a primary key. Then try to create the index again.

CTE0107 Directory ″%1″ does not exist.

Explanation: You specified a directory which does not exist.

What to do: Create the directory, ensure accessibility to the instance owner. Then try to specify the directory again. Note that in a distributed DB2 envionment, this directory has to exist on every physical node.

CTE0108 The internal size ″%4″ of the key columns on object ″%1″.″%2″ is larger than maximum allowed size of ″%3″.

Explanation: The internal representation of the key columns exceeds the maximum size.

What to do: Change the layout of the table before creating the index again. Use smaller key columns, which also benefit performance.

CTE0109 The number of key columns ″%3″ on object ″%1″.″%2″ is larger than the allowed maximum of ″%4″.

Explanation: A maximum number of 14 key columns is supported.

What to do: Change the layout of the table before creating the index again.

CTE0111 The file ″%1″ is not readable.

Explanation: The file specified cannot be read.

What to do: Check the access rights for the file. Take into account that the Stored Procedure runs as a fenced user ID, which may also require rights to work on this file.

CTE0112 The file ″%1″ cannot be opened.

Explanation: The file specified could not be opened.

What to do: Verify that the file is correctly specified.

CTE0113 Error converting model file ″%1″ to UTF8 encoding.

Explanation: The specified CCSID or the default database CCSID does not match the model file CCSID.

What to do: Ensure correct specification of the model file CCSID.

CTE0114 Unable to register document model ″%1″ in file ″%2″.

Explanation: The model file could not be used.

What to do: Check that the model file syntax is correctly specified.

CTE0115 A locking problem occurred. Lock Manager information: ″%1″ ″%2″.

Explanation: An internal locking problem occurred.

What to do: Check the current locks using the `db2text control` command. Using the same command, clean up the pending locks. If this does not help, stop and restart the locking and update services.

CTE0116 Operation conflicts with existing lock.

Explanation: You have tried to use a command that is currently not allowed when other commands are running on the index.

What to do: Check the locks held on this index to see which commands are currently running. Wait until the other commands have finished. If the operation is no longer running but the lock is still active, clean up the locks for the index and try again.

CTE0117 All available lock space for databases is used. Change the configuration.

Explanation: You tried to work on more databases than are configured in your lock file.

What to do: Change the number of databases you want to work in parallel with in your lock configuration `db2extlm.cfg`. Restart the update and locking services using the `db2text stop` and `db2text start` commands.

CTE0118 All available lock space for indexes on a databases is used. Change the configuration.

Explanation: You tried to work on more indexes for one database than are configured in your lock file.

What to do: Change the number of indexes you want to work in parallel with in your lock configuration file `db2extlm.cfg`. Restart the update and locking services using the `db2text stop` and `db2text start` commands.

CTE0119 All available space for locks on an index is used.

Explanation: The operations you are running require more locks for one index than are configured in your lock configuration file.

What to do: Change the number of locks you want to work in parallel with in your lock configuration `db2extlm.cfg`. Restart the update and locking services using the `db2text stop` and `db2text start` commands.

CTE0120 Update and locking services configuration file error.

Explanation: The configuration file `db2extlm.cfg` is in error.

What to do: Check the `db2extlm.cfg` file and correct the error. Restart the update and locking services using the command `db2text start`.

CTE0121 The update and locking services configuration file cannot be opened.

Explanation: The file `db2extlm.cfg` could not be opened.

What to do: Check if the file exists and that it can be accessed. If the file cannot be accessed, try to update your db2 instance using `db2iupdt`.

CTE0122 A syntax error was found in the update and locking services configuration file.

Explanation: A syntax error was found in the update and locking services configuration file.

What to do: Check the update and locking services configuration file for errors.

CTE0126 The update and locking service input file ″%1″ is corrupted.

Explanation: A required file for update and locking services is corrupted.

What to do: Check if the file exists and if it can be accessed. If you can access the file, rename the file and restart the update and locking services. The file should be created again. However, this action removes all of the specified frequency updates for create index.

CTE0127 An update and locking service error has occurred. Reason code: ″%1″.

Explanation: An internal error has occurred in the update and locking service area.

What to do: Stop DB2 and Net Search Extender and then clean up your shared resources. Try to start both again. If this does not work, report the problem to your IBM representative.

CTE0129 NULL values are not allowed to be passed as parameters.

Explanation: DB2 has passed a NULL value to an internal user-defined function.

What to do: First make sure the specified base table has a primary key. Change your select statement to avoid this problem. Switch on the trace function and pass the returned information on to IBM Services.

CTE0130 The specified search argument exceeds the maximum length. The current search argument length is ″%1″ and the maximum supported length is ″%2″.

Explanation: The length of the specified search argument is ″%1″. The maximum length must not exceed ″%2″.

What to do: Reduce the length of your search argument to ″%2″.

CTE0131 The user-defined function ″%1″. ″%2″ does not exist.

Explanation: The specified user-defined function does not exist in this database.

What to do: Check the name specified for this user-defined function, or register the user-defined function in the database you are using.

CTE0132 The text index ″%1″. ″%2″ does not exist.

Explanation: The specified text index does not exist in this database.

What to do: Check the name specified and the database you are using. Use the `db2ext.textcolumns` view to see the existing text indexes.

CTE0133 The text index ″%1″. ″%2″ already exists.

Explanation: The text index that you specified already exists in this database.

What to do: Check the name specified and the database you are using. Use the `db2ext.textcolumns` view to see the existing text indexes.

CTE0135 The object "%1". "%2" does not exist.

Explanation: The specified object name does not exist in this database.

What to do: Check the object name specified and the database you are using.

CTE0136 The column "%1" does not exist in "%2"."%3".

Explanation: The specified column does not exist.

What to do: Check the column name that you specified. Check the table, view, or database you are using.

CTE0137 The table space "%1" does not exist.

Explanation: The specified tablespace does not exist in this database.

What to do: Check the name specified and the database you are using.

CTE0138 The table space "%1" is not regular.

Explanation: The specified table space is not regular. The event table can only be created in a regular table space.

What to do: Use this command again with a regular table space.

CTE0139 The environment variable "%1" is not set.

Explanation: A required environment variable is not set.

What to do: Check your environment, specify the required variable, and use the command again.

CTE0140 The database "%1" is already enabled for text.

Explanation: The database you specified is already enabled for text.

What to do: Check the name that you specified. Also check the DB2DBDFT variable that implies an implicit connection.

CTE0141 The database "%1" is not enabled for text.

Explanation: The database you specified is not enabled for text.

What to do: Check the database name you specified and the DB2DBDFT variable. If the database name is correct, use the command `db2text enable database for text`.

CTE0142 The command requires control authority on "%1". "%2" granted to user "%3".

Explanation: You do not have the authority to use this command.

What to do: Only the owner of this table can use this command or provide you with the required authorization.

CTE0143 The command requires database administration authority for user "%1".

Explanation: You do not have the required authority to use this command.

What to do: Only the owner of the database can use this command or provide you with the required authorization.

CTE0144 There is at least one text index active in database "%1".

Explanation: You cannot disable your database until all text indexes are dropped.

What to do: See the `db2ext.textcolumns` view for the existing indexes. Drop the existing indexes using the DROP INDEX command or

specify the FORCE option with the DISABLE DATABASE command.

CTE0145 **The CCSID ″%1″ is not supported.**

Explanation: The CCSID that you specified is not supported.

What to do: Specify a valid CCSID.

CTE0146 **The language ″%1″ is not supported.**

Explanation: The specified language is not supported.

What to do: Specify a valid language.

CTE0147 **The format ″%1″ is not supported.**

Explanation: The specified format is not supported.

What to do: Specify a valid format.

CTE0148 **The specified format ″%1″ does not accept a model file.**

Explanation: The format ″%1″ does not support model files.

What to do: Use a format that accepts a model file, or remove the model file from your command.

CTE0149 **Too many terms (beginning with ″%1″) are specified for the index update frequency.**

Explanation: The syntax for the update frequency is not correct.

What to do: Ensure that the DAY, HOUR, and MINUTE parameters are only specified once.

CTE0150 **Unexpected end of command. Check the command syntax.**

Explanation: The command syntax is not correct.

What to do: Check the command syntax. Verify that you specified the required parameters.

CTE0151 **Token ″%1″ is unexpected. Check the command syntax.**

Explanation: The syntax of the command is not correct.

What to do: Check the command syntax and verify that the token you are using is allowed in the specific command.

CTE0152 **Token ″%1″is too long.**

Explanation: The token is too long.

What to do: Check the command syntax and verify that the token is reduced to the maximum size allowed.

CTE0153 **Token ″%1″ occurs twice in the update frequency.**

Explanation: You specified an incorrect syntax for the update frequency.

What to do: Ensure that the DAY, HOUR, and MINUTE parameters are only specified once.

CTE0154 **The value ″%1″ for ″%2″ is out of range. The valid range is ″%3″ - ″%4″.**

Explanation: You specified an incorrect value. The value should be in the allowed range.

What to do: Update your command. Change the value to match those in the allowed range.

CTE0155 **The search string is empty.**

Explanation: You specified an empty search string.

What to do: Check that the search string includes valid alphanumeric characters.

CTE0157 **Syntax error near ″%1″.**

Explanation: You specified an incorrect search syntax.

What to do: Check the syntax near %1. Correct and try again.

CTE0158 The freetext search string is missing.

Explanation: Specify a freetext string.

What to do: Check that the search string after ″is about″ includes valid alphanumeric characters.

CTE0159 Search string exceeds the allowed length of ″%1″.

Explanation: The search string is too long.

What to do: Reduce the size of the search string and try again.

CTE0160 No section name has been specified in the search string.

Explanation: You need to specify a valid section name.

What to do: Add a valid section name and try again.

CTE0162 The escape command could not be processed.

Explanation: Your search string includes too many special characters that can be used as masking characters.

What to do: Reduce the number of special characters in your search term, or avoid the escape command. The following special characters can be used: ! * + , _ . : ; { } ~ | ? [ ] ` = \

CTE0163 No thesaurus name specified in thesaurus clause.

Explanation: A thesaurus search is requested without a thesaurus name.

What to do: Specify a thesaurus name in your search argument.

CTE0164 Syntax error in thesaurus relation ″%1″.

Explanation: The specified syntax for the thesaurus relation is not correct.

What to do: Update the thesaurus relation according to the syntax specification.

CTE0166 Freetext must be the last statement in search query.

Explanation: It is not allowed to have further operators after the ″is about″ token.

What to do: Rewrite the query string. The last operator must be ″is about″.

CTE0167 Syntax error in free text query ″%1″.

Explanation: The syntax for the free text string is not correct.

What to do: Update the free text string according to the syntax specification.

CTE0168 A left parenthesis in a section statement is missing.

Explanation: The syntax for the section statement is not correct.

What to do: Update the section statement according to the syntax specification.

CTE0169 A comma or right parenthesis is missing in a section statement.

Explanation: The syntax for the section statement is not correct.

What to do: Update the section statement according to the syntax specification.

CTE0170 A closing double quote is missing.

Explanation: The specified syntax for the search term is not correct.

What to do: Update the search term according

to the syntax specification.

CTE0171 **An open double quote for a section name is missing.**

Explanation: The syntax for the section statement is not correct.

What to do: Update the section statement according to the syntax specification.

CTE0172 **The closing double quote for the section name is missing.**

Explanation: The syntax for the section statement is not correct.

What to do: Update the section statement according to the syntax specification.

CTE0173 **One escape character must be defined in an escape clause.**

Explanation: There can be no more than one character in an escape clause.

What to do: Remove the additional characters in the escape clause.

CTE0174 **A blank character is not allowed as an escape character.**

Explanation: It is not allowed to have a blank character in an escape clause.

What to do: Change the escape clause to a clause with a valid character.

CTE0175 **An escape clause is defined but no mask character is found in the search phrase.**

Explanation: An escape clause is specified without using a mask character.

What to do: Remove the escape clause.

CTE0176 **The succeeding character of an escape character in the phrase is neither the same character nor a mask character.**

Explanation: The character after the escape character must be either a masking character or the escape character itself.

What to do: Change the search string to correctly use the escape character.

CTE0177 **The number value ″%1″ is invalid.**

Explanation: The specified number in the search argument is not valid.

What to do: Check the documentation about the valid range. Update the value in the search argument.

CTE0178 **Mask characters in fuzzy phrase must be preceded by an escape character.**

Explanation: Masking together with fuzzy search is not allowed.

What to do: Update the search string with an escape character.

CTE0179 **Thesaurus name ″%1″ exceeds allowed length of ″%2″.**

Explanation: Primary keys longer than 60 bytes are not supported.

What to do: Change the layout of the table before creating the index again.

CTE0180 **Thesaurus ″%1″ can not be found.**

Explanation: The thesaurus specified cannot be found.

What to do: Check that the thesaurus files are located in the thesaurus directory or fully qualified.

CTE0181 Library ″%1″ cannot be loaded.

Explanation: A library cannot be found.

What to do: Check that the library is located in the library path and available. Start and stop DB2 to ensure that the current settings are used.

CTE0182 Function ″%1″ cannot be loaded from library ″%2″.

Explanation: A library entry point cannot be loaded.

What to do: The library accessed seems to be invalid. Check that the library is specified only once.

CTE0183 Error occurred using shared system resources.

Explanation: A request to shared system resources like shared memory or semaphores cannot be fulfilled.

What to do: Check the current system status and configuration. On UNIX use the `ipcs` command to check the resources. Stop all applications, such as DB2 and DB2 Net Search Extender. If further resources are listed, clean them up using `ipcrm`.

CTE0184 No db2text start command was issued.

Explanation: A command was called which requires the locking and update services.

What to do: Start the update and locking services with `db2text start`.

CTE0185 The update and locking services are already active.

Explanation: A `db2text start` is issued but the update and locking services are already running.

What to do: No further action.

CTE0186 Update and locking service error occurred, check db2diag.log for details.

Explanation: An update and locking service error occurred.

What to do: Check the `db2diag.log` for further information, or clean up your shared resources. See also CTE0183.

CTE0187 Update and locking services are still active, use FORCE option to stop the services.

Explanation: The `db2text stop` command has not stopped the locking services, there are still processes running.

What to do: Check with `db2text control` which processes are running and wait for those to finish. If you need to stop them, use the FORCE option.

CTE0188 There is a temporary problem using update and locking services. Please try again.

Explanation: The `db2text stop` command has not stopped the locking services. Programs are still running or an inconsistent situation is found.

What to do: Check with `db2text control` which processes are running and wait for those to finish. To stop them, use the FORCE option.

CTE0189 The executable program ″%1″ cannot be found.

Explanation: The program file cannot be located or accessed.

What to do: Check if the program file is located in the `bin` or `adm` directory of the DB2 server. The installation is corrupt if the file cannot be found.

CTE0190 The executable program ″%1″ cannot be started.

Explanation: The program cannot be started.

What to do: Check if the program is located in

the bin or adm directory of the DB2 server and that the appropriated libraries are installed. For further information, call the program manually on the server.

CTE0191 The drop index operation is incomplete. Check db2diag.log for details.

Explanation: The drop index operation is incomplete, possibly caused by the FORCE option.

What to do: Using the FORCE option drops everything regardless of any errors. Check the index directory for pending files and remove these manually.

CTE0192 Errors occurred in an update index operation. Check event table "%1". "%2" and db2diag.log for details.

Explanation: During the index update process, any document errors are written to the event table.

What to do: Check the event table for more information about the document errors. Clean up the event log after the problems have been fixed.

CTE0194 The type "%1" of column "%2" is not supported.

Explanation: You used a column that is not in the list of the supported ones.

What to do: Check create index for a list of valid columns for Keys and Indexing. Make the appropriate changes to the command and try again.

CTE0195 "%1" is not an absolute path.

Explanation: An absolute path on the server is required.

What to do: Check the path and write an absolute path in the command.

CTE0198 No corresponding text index.

Explanation: There is no text index on the column.

What to do: Check if the text index still exists.

CTE0199 There is no text index corresponding to column "%1" of table "%2".

Explanation: You tried to search on a column without a text index.

What to do: Check the column you are searching on, or create a text index on the column.

CTE0200 At least one command option must be specified.

Explanation: The ALTER INDEX command changes the characteristics of an index, such as the update and storage options. None of the characteristics to be changed was specified.

What to do: Specify at least one command option. Refer to the command syntax for all possible options.

CTE0201 There is a conflict with an existing text index on the same column.

Explanation: A text index defined on the same column was created with different parameters from this create index command.

What to do: Correct the parameter values in the create index command. Make sure that following parameters have the same value for the existing index and the index to be created: ccsid, language, format, document model, index configuration, column function, and attributes.

CTE0202 The object "%1"."%2" must be a view when key columns are specified.

Explanation: The specified object is not a view. The KEY COLUMNS FOR INDEX ON VIEW

clause is only allowed when indexing a column of a view.

What to do: Remove the KEY COLUMNS FOR INDEX ON VIEW(SQL-columname-list) clause.

CTE0203 The text index ″%1″.″%2″ was not created with the CACHE TABLE option. This is required for command execution.

Explanation: This command can only be executed if the specified index was created with the CACHE TABLE option.

What to do: Create an index with CACHE TABLE option. Refer to the documentation for the command syntax.

CTE0204 An attribute name is missing. Add ″AS <attribute name>″ to the attribute expression.

Explanation: Whenever a column expression is used in the attribute expression, an attribute name must be supplied. For example: (C1+C2 AS myname).

What to do: Add ″AS <attribute name>″ to the attribute expression.

CTE0205 CACHE TABLE expressions are not valid.

Explanation: The column list in the cache table expression is not valid.

What to do: Correct the cache table column list in the create index command. Make sure the columns exist in the specified table. If a function is applied on a column, verify that it is used correctly.

CTE0206 ATTRIBUTE expressions are not valid.

Explanation: The column list in the attribute expression is not valid.

What to do: Correct the attribute column list in the create index command. Make sure the columns exist in the specified table. If a function is applied on a column, verify that it is used correctly.

CTE0207 KEY COLUMNS FOR INDEX ON VIEW not specified for index on view ″%1″.″%2″.

Explanation: If indexes on views are created, the KEY COLUMNS FOR INDEX ON VIEW(SQL-columname-list) clause must be specified. The list of column names specifies the columns that UNIQUELY identify a row in the view.

What to do: Include the KEY COLUMNS FOR INDEX ON VIEW(SQL-columname-list) clause in the create index command.

CTE0208 INITIAL SEARCH RESULT ORDER columns are not valid.

Explanation: The column list in the INITIAL SEARCH RESULT ORDER(SQL-order-by list) expression is not valid.

What to do: Correct the order by column list in the create index command. Check if the syntax is correct and the columns exist in the specified table. If a function is applied on a column, verify that it is used correctly.

CTE0209 The type ″%1″ of attribute column ″%2″ is not supported, type DOUBLE is required.

Explanation: For attribute columns, the only supported data type is DOUBLE.

What to do: Make sure the attribute columns of the table with the text column to be indexed are of type DOUBLE. It may be possible to use cast operators in attribute column expressions. Refer to the SQL Reference for data types which can be casted to double.

CTE0210 The value ″%1″ for index configuration parameter ″%2″ is not valid. A valid value is ″%3″.

Explanation: The specified value for the configuration parameter is incorrect. For valid

values of the parameters refer to the command syntax.

What to do: Correct the index configuration parameter value in the create index command.

CTE0211 **"%1" is not a valid index configuration parameter.**

Explanation: The index configuration option is not known.

What to do: Check the create index command syntax. Valid index configuration options are TreatNumbersAsWords and IndexStopWords. These have to be comma separated: index configuration(treatnumberaswords 1, indexstopwords 1).

CTE0212 **Internal index configuration file "%1" could not be saved.**

Explanation: The internal configuration file for the index could not be saved.

What to do: Make sure the instance owner has write permissions to the directory the file should be saved in. If a file with the same name already exists, make sure that it is writable for the instance owner.

CTE0213 **Internal index configuration file template "%1" could not be loaded.**

Explanation: The internal index configuration file template could not be read.

What to do: Make sure the file exists in the correct location and is readable.

CTE0214 **Internal error when setting new entry "[%1],%2=%3" for index configuration file.**

Explanation: Internal error while writing an internal configuration file for the index.

What to do: If the file exists, check if it is readable and writable for the instance owner. Check that there is enough space on the device where the file is located.

CTE0215 **Index creation on alias "%1"."%2" is not supported. Use base table "%3"."%4" instead.**

Explanation: The index cannot be created on the alias.

What to do: Type in the create index command with the base table.

CTE0217 **The schedule service is already active.**

Explanation: The service is already active, you do not need to start it.

What to do: No action required.

CTE0218 **Function "%1" failed with error code "%2".**

Explanation: A Windows function failed with the specified error code which does not allow further processing.

What to do: Use the specified Windows system error code to get detailed error information.

CTE0219 **The service "%1" could not be opened. Error code "%2".**

Explanation: The specified service cannot be found on the Windows system.

What to do: Check if the specified service is installed on the Windows system. Use the specified Windows system error code to get detailed error information.

CTE0220 **The DB2 instance profile path could not be found.**

Explanation: Internal DB2 function to obtain the DB2 instance profile path failed.

What to do: Create a DB2 instance without specifying the instance profile path information and retry the command.

CTE0221 UpdateFrequency ″%1″ is incorrectly specified.

Explanation: The syntax for the update frequency statement is not correct.

What to do: Correct the update frequency statement according to the syntax specification.

CTE0222 The schedule service input file ″%1″ is corrupted.

Explanation: The scheduler file containing index update information is corrupted.

What to do: Use your system editor and try to correct the problem. Maybe an entry has been truncated, or the ending line character has been deleted. If this does not restore the file content, try the following:

- Call command `db2text stop` to stop the scheduler.
- Delete the scheduler service file.
- Call command `db2text start` to start the scheduler.
- Use command `db2text alter index ...` to recreate the update frequency entries for all concerned indexes.

CTE0223 File ″%1″ could not be closed.

Explanation: The file specified cannot be closed.

What to do: Verify that the file is correctly specified.

CTE0224 File ″%1″ could not be copied to ″%2″.

Explanation: The first file cannot be copied to the second file.

What to do: Verify that the files are correctly specified. Check if the second file already exists and is read only. Also check if there is enough free space on the system.

CTE0225 File ″%1″ could not be removed.

Explanation: The file specified cannot be removed from the system.

What to do: Verify that the file is specified correctly and check the file access rights.

CTE0225 File ″%1″ could not be removed.

Explanation: The file specified cannot be removed from the system.

What to do: Verify that the file is specified correctly and check the file access rights.

CTE0227 A write operation on file ″%1″ failed.

Explanation: The file specified is not writable.

What to do: Verify that the file is correctly specified and check the file access rights. Also check if there is enough free space on the system.

CTE0228 The user has insufficient access rights at the operating system level.

Explanation: The command requires administrator rights at the operating system level.

What to do: Ensure that you have operating system administrator rights. Check if you are a member of the administrator group.

CTE0231 ″%1″ is not defined in same nodegroup (″%4″) as the tablespace of ″%2″.″%3″.

Explanation: The tablespace of the administration tables is required to be distributed over different nodes in exactly the same way as the table containing the text column to be indexed. To enforce this, it is checked whether the specified tablespace is defined in the same nodegroup.

What to do: Specify a tablespace that is defined in the same nodegroup as the table containing the text column to be indexed.

CTE0232 **The specified or default tablespace ″%1″ is not single-noded. This is necessary for an index on a view, or when the CACHE TABLE option is specified.**

Explanation: An index on a view or with the CACHE TABLE option enabled is only supported for tables on a single node.

What to do: Put the table in a single-noded tablespace if the default tablespace caused this error. Alternatively, specify another single-noded tablespace, if you specified a multi-noded tablespace.

CTE0233 **There is a conflicting administration command running. Please retry this command later.**

Explanation: Another administration command is still running or terminated abnormally without releasing the command lock.

What to do: Check with CONTROL LIST which locks are still active. If there is an active lock but no command running, clear the lock manually using the CONTROL CLEAR command. Be aware that someone else may be running the administration command holding the lock.

CTE0234 **There is a conflicting administration command running on a text index. Please retry this command later, or specify the FORCE option of a DISABLE DATABASE command.**

Explanation: Another administration command is still running or terminated abnormally without releasing the command lock.

What to do: Check with CONTROL LIST which locks are still active. If there is an active lock but no command running, clear the lock manually using the CONTROL CLEAR command. Be aware that someone else may be running the administration command holding the lock. For a DISABLE DATABASE command you may specify the FORCE option which stops all other commands on that database.

CTE0235 **No valid license found for DB2 Net Search Extender.**

Explanation: There was no valid license found for DB2 Net Search Extender.

What to do: Check whether the license was correctly installed with `db2lic`. Make sure existing instances are updated after the product install.

CTE0236 **Only Node0 is supported on MPP instances.**

Explanation: Text Indexes can only be created on MPP instances, if the table with the text column to be indexed resides on Node0.

What to do: Check the node group of the tablespace in which the table is defined.

CTE0237 **Internal error: log table ″%1″.″%2″ contains an invalid operation ″%3″.**

Explanation: The log table keeps track of operations executed on the table containing the indexed text column. This table might be corrupted, as it contains an entry not written by DB2 Net Search Extender.

What to do: Check the log table and delete the corrupted entry.

CTE0238 **Internal error: table ″%1″.″%2″ contains an incorrect syntax expression in column ″%3″.**

Explanation: There is an error in the expression list in the specified text column.

What to do: Check the delimiter Begin and End pairs.

CTE0239 **Internal error: total length of index properties ″%1″ exceeds maximum ″%2″.**

Explanation: The maximum size of the index properties (1016 bytes) is exceeded. The

properties contain the instance, index, and work directory as well as other information.

What to do: Make sure these path names are not too long.

CTE0240 Internal error: setting environment variable ″%1″ failed.

Explanation: Setting the specified environment variable failed. There may be a problem with the environment setup.

What to do: Check your OS specific guidelines.

CTE0241 Internal error: datalink UDF ″%1″.″%2″ returns type ″%3″. The expected type is: ″%4″.

Explanation: The datalink UDF used to fetch the content of a datalink reference is defined with an unexpected datatype.

What to do: Extract the definition of the datalink UDF from `SYSCAT.FUNCTIONS` and report the error to IBM services.

CTE0242 Value ″%1″ for parameter ″%2″ is invalid.

Explanation: The search stored procedure or the table valued function `DB2EXT.TEXTSEARCH` was called with invalid parameters.

What to do: Correct the parameter values of the search stored procedure or table valued function. For valid parameters refer to the documentation.

CTE0243 The cache for text index ″%1″.″%2″ has not been activated.

Explanation: A Net Search Extender operation requires an activated cache. The cache is currently not activated. These are the possible reasons:

- The cache has never been activated after the last DB2TEXT START command.
- The cache has been explicitly deactivated with the DB2TEXT DEACTIVATE CACHE command.

What to do: Perform a DB2TEXT ACTIVATE CACHE command for the index and rerun the Net Search Extender operation.

CTE0244 Internal error: call to ″%1″ returns rc=″%2″, SQLCODE=″%3″.

Explanation: An internal processing error occurred when calling an internal function.

What to do: If the error persists, start a trace and check the `db2diag.log`. Report the error.

CTE0245 The requested cache size exceeds the available cache size. Increase the maximum cache size to a value > ″%1″ or decrease the pctfree value.

Explanation: The cache size necessary to load all data exceeds the MAXIMUM CACHE SIZE value for an index. This can be detected during activation of the cache (the DB2TEXT ACTIVATE command), or by an index update operation while the cache is activated.

What to do: If the error was reported in a DB2EXT ACTIVATE command, recalculate the maximum cache size using the DB2EXT.MAXIMUM_CACHE_SIZE function and alter the MAXIMUM CACHE SIZE setting for the index. Eventually decrease the PCTFREE value. If the maximum number of documents is exceeded during incremental update, rebuild the cache with the commands `db2 deactivate cache` and `db2text activate cache recreate`.

CTE0246 File ″%1″ is empty.

Explanation: A DB2TEXT CREATE INDEX command failed because the document model file specified in the command is empty.

What to do: Specify a valid document model file in the command.

CTE0247 A DB2 Net Search Extender stored procedure could not be created.

Explanation: A DB2TEXT ENABLE DATABASE command failed to create the internal stored procedure DB2EXT.CTESRVSP.

What to do: Check the additional DB2 error message associated with a CREATE PROCEDURE statement for details. If the error cannot be corrected by removing an existing stored procedure with an identical name, start a trace and report the error.

CTE0248 The generated search string is too long. Reduce the complexity of search query.

Explanation: A Net Search Extender query is too long or too complex to be processed by the base search engine. The complexity is affected by thesaurus expansions, FUZZY FORM OF expressions, and masking characters.

What to do: Reduce complexity or length of the query.

CTE0249 Executable program ″%1″ terminated abnormally.

Explanation: When executing a Net Search Extender command, the executable ″%1″ was called, but terminated abnormally.

What to do: Verify, that the executable was not terminated explicitly by user interaction, for example, a signal. If not, start a trace, rerun the command, and report the error.

CTE0250 The return type ″%1″ of column type transformation function ″%2″.″%3″ is not supported.

Explanation: In a DB2TEXT CREATE INDEX command a column type transformation was specified that returns an unsupported datatype. Supported datatypes are: CHARACTER, VARCHAR, LONG VARCHAR, CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, BLOB, and DATALINK.

What to do: Choose a different column type transformation function.

CTE0251 Internal error: the column type ″%1″ is not supported.

Explanation: A column type is used that is not in the list of supported types.

What to do: Check create index for a list of valid columns for Keys and Indexing. Make the appropriate changes to the command and try again. If the error persists, start a trace and also check the `db2diag.log`. Report the error to IBM Services.

CTE0252 The parameter ″%1″ is missing.

Explanation: Internal error - when executing a Net Search Extender command, an administration executable program was called with a missing parameter ″%1″.

What to do: Try to change Net Search Extender parameter commands to avoid the problem. If the error persists, switch on the trace function and report the error to IBM Services.

CTE0253 The document listed in the log view was not found.

Explanation: The contents of a text document that is listed in the log view has changed and could not be accessed.

What to do: Check that the document exists and the read/access permissions of the text documents to be included in the index.

CTE0254 The cache for index ″%1″ is already activated.

Explanation: The index has already been activated with the ACTIVATE CACHE command.

What to do: Check the specified index name and the database that you are using.

CTE0255 **A column name for a cache result column expression is missing. Add ″AS <cache column name>″ to the expression.**

Explanation: A cache result column expression must be named. For example: 'C1+C2 AS myresult'.

What to do: Add ″AS <cache column name>″ to the expression.

CTE0256 **The query necessary to select data for indexing failed. Reduce the complexity of the attribute, cache table, or the initial search result order expressions.**

Explanation: Net Search Extender creates a query from the expressions in your command to select data for indexing from the database. The query failed because it was too complex.

What to do: Reduce the complexity of attribute, cache table, or initial search result order expressions.

CTE0257 **Error creating shared memory.**

Explanation: The shared memory resource could not be created due to a previous error or permission problem.

What to do: Check `db2diag.log` for further information, or clean up your shared resources. See also error CTE0183.

CTE0258 **Shared memory version error.**

Explanation: The shared memory resource could not be accessed because it is corrupted or there is a version conflict.

What to do: Check `db2diag.log` for further information. Disable and re-enable the database and then try again.

CTE0259 **Cannot insert entry in global shared memory. Entry already exists.**

Explanation: An entry to be inserted in global shared memory already exists because of a previous error.

What to do: Check `db2diag.log` for further information. Restart the update and locking services using the commands `db2text stop` and `db2text start`.

CTE0260 **Cannot access entry in global shared memory. Entry not found.**

Explanation: An entry to be removed from global shared memory does not exist because of a previous error.

What to do: Check `db2diag.log` for further information. Try to restart the update and locking services using the commands `db2text stop` and `db2text start`.

CTE0261 **There is at least one cache activated for a text index in this instance. Deactivate the cache for any activated index using the DEACTIVATE CACHE command, or use the FORCE option to stop.**

Explanation: The db2text stop command can only be used if you run a DEACTIVATE CACHE command for all text indexes that have been activated with the ACTIVATE CACHE command.

What to do: Deactivate the cache for any activated index using the DEACTIVATE CACHE command or use the FORCE option to stop.

CTE0262 **The value for parameter ″%1″ is too long.**

Explanation: The value exceeds the maximum allowable size.

What to do: Check the maximum size.

CTE0263 **The text index ″%1″.″%2″ was created with the RECREATE INDEX ON UPDATE option. In this context, the UPDATE MINIMUM or COMMITCOUNT FOR UPDATE may not be specified.**

Explanation: Update minimum and commitcount for update are only effective if the index is updated incrementally.

What to do: If you want to recreate the index each time an update is performed, remove the UPDATE MINIMUM and COMMITCOUNT FOR UPDATE settings. If you want to use UPDATE MINIMUM and COMMITCOUNT FOR UPDATE, do not specify RECREATE INDEX ON UPDATE.

CTE0264 **Errors occurred in an activate index operation. Check event view ″%1″.″%2″ and the db2diag.log for details.**

Explanation: During the index activate process, errors are written to the event table and the `db2diag.log` file.

What to do: Check the event table for more information about the document errors. Clean up the event log after the problems have been fixed.

CTE0265 **The tablespace of a user table or administration tablespace (″%1″) is not only defined on node 0.**

Explanation: If text indexes are created on MPP instances, the tablespace of the user table must only reside on Node0.

What to do: Use a table where the tablespace resides on Node0.

CTE0266 **ValueFrom ″%1″ must be smaller than ValueTo ″%2″.**

Explanation: The values specified in the attribute search are not valid. If the search syntax is 'BETWEEN ValueFrom AND ValueTo', the lower boundary (ValueFrom) must be smaller than upper boundary(ValueTo).

What to do: Change the boundaries in the 'BETWEEN ValueFrom AND ValueTo' clause.

CTE0267 **The Net Search Extender database objects in the database ″%1″ are in an inconsistent state.**

Explanation: At least one DB2 Net Search Extender object is missing or corrupted. Either the database has not been migrated after installation of a new DB2 Net Search Extender product version, or a database user has changed or dropped Net Search Extender internal object(s). In this case, all text indexes are lost and the database has to be disabled for text.

What to do: For a database migration to the current version please follow the migration description found in the DB2 Net Search Extender documentation. Alternatively, issue a DB2TEXT DISABLE DATABASE command using the FORCE option. You can then enable the database for text again by using the DB2TEXT ENABLE DATABASE command.

CTE0270 **Logtable ″%1″.″%2″ could not be modified after incremental update. Entries are to be processed during the next UPDATE.**

Explanation: When starting an incremental index update, a timestamp is created. This serves as a threshold for change records to be processed. Changes occurring concurrently to the incremental update are then processed later, during the next update. In certain situations, there can be changes in transactions that are uncommitted at the time the update starts, but are committed while the index update is being performed. This may potentially lead to inconsistencies.

To avoid such an inconsistent situation, the change records prior to the threshold timestamp are not deleted from the logtable, although they have been partially processed. On the next incremental update the changes will be re-applied to the index.

What to do: On the next index update the changes are re-applied to the index. In case of delete operations, this can lead to the following error: CTE0101: ItlEnReasonCode_Docmap_ docid_not_found.

Note that this error can be ignored, as the document was already deleted. If CTE0270 errors frequently occur, consider dropping and re-creating the index with a modified timestamp threshold for incremental index update. For example: db2text ″CREATE INDEX ... INDEX CONFIGURATION(UPDATEDELAY 30)″

This means that processing during an incremental update run only changes records older than 30 seconds and avoids interference with concurrent change transactions of less than 30 seconds.

CTE0273 The cache for index ″%1″,″%2″ is already activated.

Explanation: The index has already been activated with the ACTIVATE CACHE command.

What to do: Check the specified index name and the database that you are using.

CTE0274 The target database system ″%1″ for the connection is not supported.

Explanation: You tried to execute a DB2TEXT command with a connection to a database system that is not supported by DB2 Net Search Extender.

CTE0275 The type and version information for server ″%1″ could not be found.

Explanation: The type and version information for the server could not be found in the DB2 catalog view 'SERVERS'.

What to do: Make sure that the DB2 federated environment is set up correctly.

CTE0276 The datalink UDF could not be found.

Explanation: The datalink UDF is not registered as a user defined function in the database.

What to do: Execute the command 'db2 -tvf ctedlcud.ddl' in the directory db2ext/ddl.

CTE0277 A cache memory segment could not be attached.

Explanation: The system cannot allocate enough memory to load a large cache segment, or the cache segment cannot be opened because it has been previously deleted.

What to do: Check your system settings and increase the amount of paging space and free memory. For large cache sizes you may need to prepare your system. Refer to the DB2 Net Search Extender documentation. Use the DEACTIVATE and ACTIVATE [RECREATE] commands to recreate the cache. If the problem persists, check db2diag.log for additional information.

CTE0278 On an AIX 32-bit system, change the MAXDATA setting before activating a large cache.

Explanation: When you use the search stored procedure on an AIX 32-bit system, you may need to change the MAXDATA setting for the db2fmp executable.

What to do: Refer to the DB2 Net Search Extender documentation for details about changing the MAXDATA setting.

CTE0279 The size of the cached data has reached a system limit.

Explanation: By decreasing the PCTFREE value, you can increase the maximum data size during cache activation. This enables the system to reserve less freespace in the cache.

What to do: Use a lower PCTFREE value or reduce your amount of text data to be cached. Use the DEACTIVATE and ACTIVATE [RECREATE] commands to recreate the cache.

CTE0280 There is not enough disk space to write persistent cache files.

Explanation: The system can not write a large enough file for persistent cache in the cache directory.

What to do: Change the persistent cache directory to an empty file system by using the ALTER INDEX command. Alternatively, reduce the cache size by decreasing the PCTFREE or MAXIMUM CACHE SIZE values or by using a temporary cache.

CTE0281 Deletion of persistent cache file ″%1″ has failed.

Explanation: The file does not exist or cannot be accessed.

What to do: Check if this file still exists and delete it manually. ″;

CTE0282 The number of documents in the cache has reached a system limit.

Explanation: By decreasing the PCTFREE value, you can increase the maximum number of document entries to be cached during cache activation. This enables the system to reserve less freespace in the cache.

What to do: Use a lower PCTFREE value or reduce the amount of document entries in the cache. Use the DEACTIVATE and ACTIVATE [RECREATE] commands to recreate the cache.

CTE0283 A cache memory segment could not be created.

Explanation: The system cannot allocate enough memory for loading a large cache segment into memory. By decreasing the PCTFREE value, you achieve a smaller cache segment size.

What to do: Check your system settings and increase the amount of paging space and free memory. You can also decrease the cache size by using a lower PCTREE value. For large cache sizes, you may need to prepare your system. Refer to the DB2 Net Search Extender documentation. Use the DEACTIVATE and ACTIVATE [RECREATE] commands to recreate the cache. If the problem persists, check db2diag.log for additional information.

CTE0284 The text index is located on node ″%1″, but the search function was called on node ″%2″.

Explanation: The search stored procedure or table valued function DB2EXT.TEXTSEARCH was not called on the node where the index is located. The search function will not automatically be distributed to the correct node.

What to do: Set the DB2NODE environment variable to the node where the index is connected before connecting to the database.

CTE0285 Search function is not allowed for a text index which is distributed to multiple nodes.

Explanation: The table valued function DB2EXT.TEXTSEARCH must not be called with indexes that are distributed to multiple nodes, since it will not be automatically distributed to the correct nodes, but executed on the coordinator node.

What to do: Use the CONTAINS, SCORE or NUMBEROFMATCHES function in a multiple node environment.

CTE0286 No row found in ″%1″.″IBMSNAP_REGISTER″ for source table ″%2″.″%3″ and capture change table ″%4″.″%5″.

Explanation: No valid entry was found in the IBMSNAP_REGISTER table for the replication capture table characteristics specified in the DB2TEXT CREATE INDEX command. A valid entry must contain the specified source table for the index incolumns SOURCE_OWNER and SOURCE_NAME, with SOURCE_VIEW_QUAL=0 and the specified replication capture table in columns PHYS_CHANGE_OWNER and PHYS_CHANGE_TABLE.

Possible causes: The specified source table was

not registered as a replication source for the replication capture table.

What to do: Register the source table correctly for DB2 Replication, or specify a correct replication capture table for the source table.

CTE0287 **Invalid value ″%1″ for ″%2″ in ″%3″.″IBMSNAP_REGISTER″ for source table ″%4″.″%5″ and capture change table ″%6″.″%7″.**

Explanation: A replication setting found in the IBMSNAP_REGISTER table is not allowed. Possible causes: 1.The column CHG_UPD_TO_DEL_INS does not contain the value 'Y'. 2.The column CCD_CONDENSED contains the value 'Y'.

What to do: When registering the source table for DB2 Replication, ensure that update operations are transformed into pairs of delete and insert operations. In addition, ensure that no condensed replication capture tables are used.

CTE0288 **Source table ″%1″.″%2″ and capture change table ″%3″.″%4″ are on different servers (″%5″ and ″%6″).**

Explanation: The specified source table and replication capture table must reside on the same server.

CTE0289 **The wrapper ″%1″ is not supported.**

Explanation: The wrapper is not supported. Refer to the DB2 Net Search Extender documentation for a list of supported wrappers. See page 134 for information.

CTE0290 **The alias ″%1″.″%2″ is not allowed in the replication clause.**

Explanation: You are not allowed to specify an alias for a nickname in a replication clause.

What to do: Specify the nickname instead of the alias, or create a new nickname for the remote table.

CTE0451 **The specified document format ″%1″ is not supported by the highlighting UDF.**

Explanation: The document format ″%1″ does not support highlighting.

What to do: Use a document format that is supported by the highlighting UDF.

CTE0452 **Syntax error near option ″%1″ in the highlighting UDF.**

Explanation: You specified an incorrect syntax near the specified option.

What to do: Check the syntax near option %1. Correct and try again.

CTE0453 **The return size of the highlighting UDF is too small.**

Explanation: The requested parts of the highlighted document does not fit into the return parameter of the highlighting UDF.

What to do: Decrease the window number, the window size and/or the number of sections from which hits should be displayed. This will reduce the document parts returned to the user.

CTE0454 **Error converting the parameters of the highlighting UDF from codepage ″%1″ to codepage UTF8.**

Explanation: The parameters of the highlighting UDF in the specified CCSID (which may be the default database CCSID), cannot be converted to UTF8.

What to do: Ensure correct specification of the CCSID.

CTE0455 **The database codepage ″%1″ is not supported in the highlighting UDF.**

Explanation: The database has a codepage which is not supported by the highlighting UDF.

CTE0456 The highlighting UDF only supports documents in codepage UTF8.

Explanation: Only documents in codepage UTF8 support the highlighting UDF.

CTE0457 The value ″%1″ for parameter ″%2″ is not valid in the highlighting UDF.

Explanation: A value for a highlighting parameter is not valid.

What to do: Check the parameter value and ensure that the value is allowed in the data range.

CTE0841 Missing command option ″%1″.

Explanation: A required command option was not specified.

What to do: Check the specified parameters and add the missing parameter.

CTE0842 No value is specified for the command option ″%1″.

Explanation: A required value for a command option was not specified.

What to do: Check the specified parameters and add the missing option.

CTE0843 No numeric value is specified for the command option ″%1″.

Explanation: A string instead of a number has been specified.

What to do: Check the specified parameters and change the string to the correct number.

CTE0844 The definition file path ″%1″ is too long.

Explanation: The specified path is too long and could not be processed.

What to do: Use a shorter path and try again.

CTE0845 No definition file is specified.

Explanation: The definition file needs to be specified.

What to do: Add a valid definition file and try the call again.

CTE0846 The definition file name ″%1″ is too long.

Explanation: The specified definition file name is too long.

What to do: Reduce the length of the definition file name to the size allowed.

CTE0847 The definition file ″%1″ does not exist.

Explanation: The specified definition file could not be found.

What to do: Check that the definition file is in the correct path and can be accessed by the current user.

CTE0849 The dictionary file ″%1″ could not be locked.

Explanation: The process was not able to lock the dictionary file. Either you do not have write access, or another process has opened the file for writing.

What to do: Check the running processes to ensure that no process is locking the dictionary file and check your access rights.

CTE0850 Output file ″%1″ already exists.

Explanation: The specified output file could not be overwritten.

What to do: Check that you are able to create the thesaurus in the specified directory.

CTE0851 The integrity of the dictionary file ″%1″ is lost.

Explanation: The thesaurus dictionary files are corrupted.

What to do: Clean up the directory and compile your definition file again.

CTE0852 Dictionary file ″%1″ version error.

Explanation: Your dictionary file was generated with an older version of the thesaurus compiler.

What to do: Compile your definition file again with the current version of the thesaurus compiler.

CTE0853 The existing dictionary ″%1″ cannot be overwritten.

Explanation: An existing dictionary cannot be overwritten.

What to do: Check your write access right on the dictionary file, its directory location and subdirectory location.

CTE0855 A thesaurus term is incorrectly specified.

Explanation: There is a syntax error in your definition file.

What to do: Check your DB2 Net Search Extender documentation for information on creating a thesaurus definition file and thesaurus support.

CTE0856 The definition file ″%1″ is empty.

Explanation: An empty definition file is not allowed.

What to do: Check your DB2 Net Search Extender documentation for information on creating a thesaurus definition file and thesaurus support.

CTE0857 No block starting line found in file ″%1″ at line ″%2″.

Explanation: There is a syntax error in your definition error.

What to do: A block has to start with ':WORDS'. Check your DB2 Net Search Extender documentation for information on thesaurus concepts.

CTE0858 An invalid relationship is specified in file ″%1″ at line ″%2″.

Explanation: There is a syntax error in your definition error.

What to do: You have to examine your 'associated-term-definition'. Check your DB2 Net Search Extender documentation for information on creating a thesaurus definition file.

CTE0859 The relationship number is out of range in file ″%1″ at line ″%2″.

Explanation: The user-defined relations are all based on the associative type. They are identified by unique numbers between 1 and 128.

What to do: Verify your relationship numbers.

CTE0861 No terms are defined in file ″%1″ at line ″%2″.

Explanation: Required terms are not specified.

What to do: Check your DB2 Net Search Extender documentation for information on creating a thesaurus definition file.

CTE0861 The thesaurus term in file ″%1″ at line ″%2″ is too long.

Explanation: The length of the thesaurus term is restricted to 64 bytes.

What to do: Alter the size of your thesaurus term and try again.

CTE0862 Strength is incorrectly specified in file ″%1″ at line ″%2″.

Explanation: There is a syntax error in your definition file.

What to do: Check your DB2 Net Search Extender documentation for information on creating a thesaurus definition file and thesaurus support.

CTE0863 Strength is out of range in file ″%1″ at line ″%2″.

Explanation: The strength value should be specified between 1 and 100.

What to do: Change the strength value so that it is a numerical value from 1 to 100.

CTE0864 Internal error: Thesaurus compiler failed with reason code ″%1″.

Explanation: An internal processing error occurred that does not allow further processing. Try to start and stop the update and locking services, as well as DB2.

What to do: If the error persists, start a trace and also check the db2diag.log.

CTE0865 The directory ″%1″ could not be created.

Explanation: The specified directory could not be created.

What to do: Check if the directory already exists and the permissions on the directory.

CTE0866 The directory ″%1″ could not be removed.

Explanation: The directory could not be removed.

What to do: Check that you have write permissions on the specified directory.

Appendix G. Document model reference

DB2 Net Search Extender provides the following reference information for document models:

- The DTD for document models
- The semantics of locator (XPath) expressions
- Limitation for text fields and document attributes
- Outside-In tag attribute values

DTD for document models

Here is a formal description of the syntax of document models in the form of a document type definition (DTD):

```
<!ELEMENT GPPModel (GPPFieldDefinition|GPPAttributeDefinition)+>
<!ELEMENT HTMLModel (HTMLFieldDefinition|HTMLAttributeDefinition)+>
<!ELEMENT XMLModel (XMLFieldDefinition|XMLAttributeDefinition)+>

<!ELEMENT GPPFieldDefinition EMPTY>
<!ATTLIST GPPFieldDefinition name CDATA #REQUIRED>
<!ATTLIST GPPFieldDefinition start CDATA #REQUIRED>
<!ATTLIST GPPFieldDefinition end CDATA #IMPLIED>
<!ATTLIST GPPFieldDefinition exclude (YES|NO) NO>

<!ELEMENT GPPAttributeDefinition EMPTY>
<!ATTLIST GPPAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition start CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition end CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition type NUMBER #REQUIRED>

<!ELEMENT HTMLFieldDefinition EMPTY>
<!ATTLIST HTMLFieldDefinition name CDATA #REQUIRED>
<!ATTLIST HTMLFieldDefinition tag CDATA #REQUIRED>
<!ATTLIST HTMLFieldDefinition meta-qualifier CDATA #IMPLIED>
<!ATTLIST HTMLFieldDefinition exclude (YES|NO) NO>

<!ELEMENT HTMLAttributeDefinition EMPTY>
<!ATTLIST HTMLAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST HTMLAttributeDefinition tag CDATA #REQUIRED>
<!ATTLIST HTMLAttributeDefinition meta-qualifier CDATA #IMPLIED>
<!ATTLIST HTMLAttributeDefinition type NUMBER #REQUIRED>

<!ELEMENT XMLFieldDefinition EMPTY>
<!ATTLIST XMLFieldDefinition name CDATA #REQUIRED>
<!ATTLIST XMLFieldDefinition locator CDATA #REQUIRED>
<!ATTLIST XMLFieldDefinition ignore (YES|NO) NO>
<!ATTLIST XMLFieldDefinition priority CDATA #IMPLIED>
<!ATTLIST XMLFieldDefinition exclude (YES|NO) NO>
```

© Copyright IBM Corp. 1995, 2003

```
<!ELEMENT XMLAttributeDefinition EMPTY>
<!ATTLIST XMLAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST XMLAttributeDefinition locator CDATA #REQUIRED>
<!ATTLIST XMLAttributeDefinition ignore (YES|NO) NO>
<!ATTLIST XMLAttributeDefinition priority CDATA #IMPLIED>
<!ATTLIST XMLAttributeDefinition type NUMBER #REQUIRED>
```

The semantics of locator (XPath) expressions

According to the XML data model, XML documents are viewed as trees containing these kinds of nodes:

- The root node
- Element nodes
- Text nodes
- Attribute nodes
- Namespace nodes
- Processing instruction nodes
- Comment nodes

The links between those nodes, in other words the tree-forming relationship, reflect the immediate containment relationship in the XML document.

The **root node** can appear only at the root and nowhere else in the tree. It contains, as its children, the document element and optional comments and processing instructions.

Element nodes can contain any kinds of nodes except for the root node. The other kinds of nodes are only allowed at terminal nodes of the tree.

There are three kinds of **containment links**: 'child', 'attribute', and 'namespace'. The 'attribute' and 'namespace' containment links must lead to attribute and namespace nodes, respectively. In other words, to access the children of an element node (in terms of graph theory) you need to follow 'attribute' links to find all contained attributes, follow 'namespace' links to find all contained namespace declarations, and follow 'child' links to find contained elements, text nodes, processing instructions, and comments.

An XPath expression needs to be interpreted with respect to a context node, and denotes a set of nodes. When used as Net Search Extender selector patterns, the context node is free, that is, a relative path pattern `p` is interpreted as `//p`.

These are the Net Search Extender XPath selector patterns:

- `Pattern '|' LocationPathPattern` in context N denotes the union of the nodes matched by Pattern and LocationPathPattern, both in context N.

- `'/'RelativePathPattern` in context N denotes whatever this RelativePathPattern denotes in the context of the root.
- `'//'RelativePathPattern` in context N denotes the union of the denotations of this RelativePathPattern interpreted in any context that is a descendant (on the child axis) of the root.
- `RelativePathPattern '/' StepPattern` matches a node in context N, if and only if that node is matched by StepPattern in the context of its parent, and its parent node is matched by RelativePathPattern in context N.
- `RelativePathPattern '//' StepPattern` matches a node in context N, if and only if that node is matched by StepPattern in the context of its parent, and it has an ancestor node that is matched by RelativePathPattern in context N.
- `'child'::NodeTest` (abbreviated syntax: NodeTest) in context N matches a node that is a child of N (on the child axis) and that satisfies NodeTest.
- `'attribute'::NodeTest` (abbreviated syntax: @NodeTest) in context N matches a node that is an attribute of N and that satisfies NodeTest.
- `NodeType '(' ')'` is satisfied for a node if and only if it is of the specified type.
- `'processing-instruction' '(' Literal ')'` is satisfied for any processing-instruction-type node that has Literal as its name.
- `'*'` is satisfied for any element or attribute node (name mask for element name).
- `NCName ':' '*'` is satisfied for any element node that has NCName as its name prefix.
- `QName` is satisfied for any node with the specified name.

Note

A NodeTest of the form NameTest assumes the node to be of the principal type on the selected axis, which is attribute type on the attribute axis and child type on the child axis. Consequently, NameTest cannot be used to choose comments or processing instruction nodes, but only child and attribute nodes. Moreover, the patterns allow for the selection of any kind of node, except for namespace nodes, because the axis specifier 'namespace' is not allowed.

Examples of patterns:

- `chapter | appendix` denotes all chapter elements and appendix elements
- `table` denotes all table elements
- `*` denotes all elements (note that this is the abbreviation of child::*)
- `ulist/item` denotes all item elements that have a ulist parent

- `appendix//subsection` denotes all subsection elements with an appendix ancestor
- `/` denotes the singleton set containing just the root node
- `comment()` denotes all comment nodes
- `processing-instruction()` denotes all processing instructions
- `attribute::*` (or `@*`) denotes all attribute nodes

This is the syntax of the locator element:

```
Locator     ::= LocationPathPattern
           | Locator '|'  LocationPathPattern
 LocationPathPattern  ::= '/' RelativePathPattern ?
           | '//'? RelativePathPattern
 RelativePathPattern  ::= StepPattern
           | RelativePathPattern '/' StepPattern
           | RelativePathPattern '//' StepPattern
 StepPattern    ::= ChildOrAttributeAxisSpecifier NodeTest
 ChildOrAttributeAxisSpecifier ::=
           ('child' | 'attribute') '::'
           | '@'?
 NodeTest    ::= NameTest
           | NodeType '(' ')'
           | 'processing-instruction' '(' Literal ')'
 NameTest    ::= '*' | NCName ':' '*' | QName
 NodeType    ::= 'comment' | 'processing-instruction'
```

NCName and QName are as defined in the XML Names Recommendation:

Limitations for text fields and document attributes

Here is a list of the limitations for text fields and document attributes:

- Maximum number of fields in an index: 32767
- Maximum number of values for one attribute of type STRING in one document: 1024
- Maximum number of attributes of type STRING: 253
- Number of characters in a STRING attribute value is truncated to 128
- Maximum number of attributes of types DATE and NUMBER: 32766
- Number of characters in a DATE or NUMBER attribute value is truncated to 128
- For NUMBER attributes, a double precision floating point number is accepted as a value.
- Maximum number of values that can be specified for one attribute of type DATE or NUMBER in one document: unlimited

These are the tags that can be included in an HTML document model:

- `<A>`

- <ADDRESS>
- <AU>
- <AUTHOR>
- <H1>
- <H2>, <H3>, <H4>, <H5>
- <H6>
- <TITLE>

Tags like <HEAD> and <BODY> that can contain other tags, cannot be specified in an HTML document model as a text field.

Outside-In tag attribute values

Possible values for the tag attribute relating to Outside-In document property tag types:

```
SCCCA_ABSTRACT
SCCCA_ACCOUNT
SCCCA_ADDRESS
SCCCA_ATTACHMENTS
SCCCA_AUTHORIZATION
SCCCA_BACKUPDATE
SCCCA_BASEFILELOCATION
SCCCA_BILLTO
SCCCA_BLINDCOPY
SCCCA_CARBONCOPY
SCCCA_CATEGORY
SCCCA_CHECKEDBY
SCCCA_CLIENT
SCCCA_COMPANY
SCCCA_COMPLETEDDATE
SCCCA_COUNTCHARS
SCCCA_COUNTPAGES
SCCCA_COUNTWORDS
SCCCA_CREATIONDATE
SCCCA_DEPARTMENT
SCCCA_DESTINATION
SCCCA_DISPOSITION
SCCCA_DIVISION
SCCCA_DOCCOMMENT
SCCCA_DOCTYPE
SCCCA_EDITMINUTES
SCCCA_EDITOR
SCCCA_FORWARDTO
SCCCA_GROUP
SCCCA_KEYWORD
SCCCA_LANGUAGE
SCCCA_LASTPRINTDATE
SCCCA_LASTSAVEDBY
SCCCA_MAILSTOP
SCCCA_MANAGERSCCCA_MATTER
SCCCA_OFFICE
```

```
SCCCA_OPERATOR
SCCCA_OWNER
SCCCA_PRIMARYAUTHOR
SCCCA_PROJECT
SCCCA_PUBLISHER
SCCCA_PURPOSE
SCCCA_RECEIVEDFROM
SCCCA_RECORDEDBY
SCCCA_RECORDEDDATE
SCCCA_REFERENCE
SCCCA_REVISIONDATE
SCCCA_REVISIONNOTES
SCCCA_REVISIONNUMBER
SCCCA_SECONDARYAUTHOR
SCCCA_SECTION
SCCCA_SECURITY
SCCCA_SOURCE
SCCCA_STATUS
SCCCA_SUBJECT
SCCCA_TITLE
SCCCA_TYPIST
SCCCA_USERDEFINEDPROP
SCCCA_VERSIONDATE
SCCCA_VERSIONNOTES
SCCCA_VERSIONNUMBER
```

Possible values for the tag attribute relating to Outside-In begin and end tag subtypes:

```
SCCCA_ALTFONTDATA
SCCCA_ANNOTATIONREFERENCE
SCCCA_CAPTIONTEXT
SCCCA_CHARACTER
SCCCA_COMPILEDFIELD
SCCCA_COUNTERFORMAT
SCCCA_CUSTOMDATAFORMAT
SCCCA_DATEDEFINITION
SCCCA_DOCUMENTPROPERTYNAME
SCCCA_ENDNOTEREFERENCE
SCCCA_FONTANDGLYPHDATA
SCCCA_FOOTNOTEREFERENCE
SCCCA_FRAME
SCCCA_GENERATEDFIELD
SCCCA_GENERATOR
SCCCA_HYPERLINK
SCCCA_INDEX
SCCCA_INDEXENTRY
SCCCA_INLINEDATAFORMAT
SCCCA_LISTENTRY
SCCCA_MERGEENTRY
SCCCA_NAMEDCELLRANGE
SCCCA_REFERENCEDTEXT
SCCCA_STYLE
SCCCA_SUBDOCTEXT
SCCCA_TOA
```

```
SCCCA_TOAENTRY
SCCCA_TOC
SCCCA_TOCENTRY
SCCCA_TOF
SCCCA_VECTORSAVETAG
SCCCA_XREF
```

Note that the tables include any document property, as well as all the tag subtypes recognized by the INSO filters. There are two subtype exceptions: SCCCA_DOCUMENTPROPERTY and SCCCA_BOOKMARK.

Appendix H. Text Search Engine

DB2 Net Search Extender provides the following Text Search Engine information:

- Tokenization
- Stopwords

Tokenization

During indexing, Net Search Extender processes document text in the following way, breaking the text up into tokens.

Words

All alphanumeric characters (″a″..″z,″A″..″Z″, ″0″..″9″) are used to create the full-text index. Separation characters are blank characters and the characters described in the sentence recognition section below. Control characters, such as line feed (also known as a new line character) and blank characters, are interpreted as follows: Control characters (less than 0x20) in the middle of the line are regarded as blank characters. Blank characters and control characters before and after a line feed (0x0A) are ignored. Line feed before and after a 1-byte character are regarded as blank characters and 2-byte characters for the same character are always regarded as the same characters. Capital letters and small letters for the same character, for example, ″A″ and ″a″, are regarded as the same characters if nothing is specified during search, or as different characters if exact matching is required during search.

Sentences

Net Search Extender recognizes ″.″, ″!″, ″?″ followed by blank characters, and the Japanese and Chinese full-stop at the end of a line as the end of a sentence.

Paragraphs

Paragraph recognition is dependent on the document format. In Plain Text format, any two consecutive new line characters (possibly with an intervening carriage return) are recognized as a paragraph boundary. In HTML, the paragraph tag <p> is interpreted as paragraph boundary. The other document formats do not support paragraph recognition.

Stopwords

Stopwords are words with a high frequency and no relevant content for the text retrieval process. Usually, all function words (in a linguistic sense) are considered stopwords, for example ″and″, ″or″, and ″in″.

© Copyright IBM Corp. 1995, 2003

Net Search Extender provides stopword processing for a list of languages, where the stopwords are not indexed and therefore, cannot be searched on. However, the result of stopword processing is a smaller and faster text index.

Note that stopwords that are not indexed are processed the same way as normal words during search. However, if a stopword has been indexed, the stopword is ignored during the search process.

Languages supporting stopwords

The following languages provide stopword processing.

AR_AA	Arabic as spoken in Arabic countries
CA_ES	Catalan as spoken in Spain
DA_DK	Danish as spoken in Denmark
DE_CH	German as spoken in Switzerland
DE_DE	German as spoken in Germany
EL_GR	Greek as spoken in Greece
EN_GB	English as spoken in the U.K.
EN_US	English as spoken in the U.S
ES_ES	Spanish as spoken in Spain
FI_FI	Finnish as spoken in Finland
FR_CA	French as spoken in Canada
FR_FR	French as spoken in France
HE_IL	Hebrew as spoken in Israel
IS_IS	Icelandic as spoken in Iceland
IT_IT	Italian as spoken in Italy
IW_IL	Hebrew as spoken in Israel
NB_NO	Norwegian Bokmal as spoken in Norway
NL_BE	Dutch as spoken in Belgium
NN_NO	Norwegian Nynorsk as spoken in Norway
PT_BR	Portuguese as spoken in Brazil
PT_PT	Portugese as spoken in Portugal
RU_RU	Russian as spoken in Russia
SV_SE	Swedish as spoken in Sweden

Appendix I. Text Search Engine reason codes

0	Operation performed successfully - no error occurred.
1	An invalid handle was passed to a function.
2	Function could not allocate enough memory.
3	Function could not perform due to access limitations or security restrictions.
4	The operation is not supported for this version of the Text Search Engine run time.
5	The operation is currently not enabled.
6	The application violated the Text Search Engine protocol by calling Text Search Engine functions in illegal order.
7	An unexpected error occurred. Please report this to your service representative.
8	An invalid language was specified.
9	The specified language is valid but not supported by the Text Search Engine run time.
10	An invalid CCSID was specified.
11	The specified CCSID is valid but not supported by the Text Search Engine run time.
12	An invalid document ID was specified.
13	The specified document format is valid but not supported by the Text Search Engine run time.
14	An invalid document format was specified.
15	The operation could not succeed due to access limitation during file input/output.
16	The operation could not succeed due to read errors during file input/output.
17	The operation could not succeed due to read errors during file input.
18	The operation could not succeed due to write errors during file output.
19	The operation could not succeed due to seek errors during file input/output.

© Copyright IBM Corp. 1995, 2003

20	The operation could not succeed due to tell errors during file input/output.
21	The operation could not succeed due to close errors during file input/output.
22	The operation could not succeed due to errors during rename operations.
23	The operation could not succeed due to errors during remove operations.
24	The operation could not succeed due to errors during mkdir operations.
25	One or more function arguments did have an invalid value (for example, an unexpected null pointer or an invalid enumeration type value).
26	The specified directory does not exist.
27	An unexpected Text Search Engine error occurred. Please examine the Text Search Engine error code in the error info object for further details.
28	An unexpected COS error occurred. Please report this error.
29	An attempt was made to update an empty document.
30	The specified argument is not supported for this operation.
31	The date attribute parser found an invalid value when trying to parse a date attribute.
32	The number attribute parser found an invalid value when trying to parse a number attribute.
33	Attribute name invalid, probably too long.
35	Reserve number for future use.
36	The input document contains an attribute (DATE, NUMBER, or STRING) that exceeds the length limit for attributes. The attribute text has been truncated to that limit.
38	The warning threshold as set by the user has been exceeded. As a consequence, this error has been generated.
39	The input document could not be indexed. It contains too many nested fields.
40	The limit of different attributes for one of the attribute types has been exceeded for this index.

46	The iterator is not (or no longer) valid, because its list is empty or deleted.
47	The function is not supported for the passed kind of handle. This error occurs, for example, when trying to use itlQueryResultEntryObtainData on a list iterator that does not represent a query result iterator.
48	This warning is issued if a stop word file cannot be found for the specified language and resource path.
49	This warning is issued if a stop word file does not contain any stop words.
50	This warning is issued if a stop word file does contain invalid data.
100	The index could not be opened because it does not exist with the specified name and/or directory.
101	The specified index name is not a valid index name.
102	The specified index directory is not a valid directory name.
103	The operation cannot be performed because the Text Search Engine detected a corruption of the index structure and/or index file sets.
104	The specified index cannot be created because it already exists with the given name and directory.
109	Before any other operation can be performed on this index, a rollback operation must be performed.
110	The index configuration file does not contain the mandatory section as specified in the error context.
111	The index configuration file does not contain the mandatory option as specified in the error context.
112	The index configuration file contains invalid data in the option as specified in the error context.
113	The index configuration file does not match the Text Search Engine version.
200	The specified document model name is not a valid model name.
201	The specified document model field name is not a valid field name.
202	The specified document model is not known.

203	The specified document model already exists and cannot be redefined.
204	Too many or too large document models have been added to the index.
205	The document model contains too many elements.
206	The document model element contains a parameter (XML attribute) that is not allowed for this type of element.
207	The document model element contains a parameter value that is not allowed for this type of parameter (XML attribute).
208	The document model element does not contain a required parameter (XML attribute), like ″name″.
209	The document model does not seem to be XML, or starts with an unexpected XML element.
210	The given XPath (locator value) contains an unexpected token.
211	The given XPath (locator value) contains an unexpected axis specifier (name followed by two colons).
212	The given XPath (locator value) contains an unexpected node test.
213	The document model directory file (extension .mdx) is corrupted.
214	The document model index file (extension .mox) is corrupted.
215	The document contains an XML element which is mapped to a document attribute and which contains another document attribute. The inner attribute is ignored.
216	The given parameter value is too long as a GPP or HTML tag.
217	The document model contains a duplicate field definition.
218	The document model contains a duplicate attribute definition.
300	The operation cannot be performed because the Text Search Engine detected a corruption in the index files used for document name mapping.
301	The operation cannot be performed because the Text Search Engine detected an invalid document number.
302	The operation cannot be performed because the Text Search Engine detected an invalid document identifer.
303	The operation cannot be performed because the Text Search Engine found no index entry for the document identifier.

304	The operation cannot be performed because the Text Search Engine found no index entry for the document number.
305	The operation cannot be performed because the Text Search Engine detected an overflow in used document numbers.
306	The document identifier that the application tried to index has appeared already in the list of documents. The Text Search Engine does not support duplicate document identifiers appearing in one indexing sequence, that is, before the update has been committed.
340	The term strength is not valid.
341	The relation number is not valid, must be in.
342	The relation type is invalid, use one of the defines described in API.
343	The phrase (term) is too long.
344	Unexpected end of file encountered while reading.
345	Version conflict detected when reading index/thesaurus files.
346	Overflow in thesaurus buffers.
347	Invalid name, probably too long a name, for file or directory.
348	Lookup did not find term (phrase) in dictionary or entry in definition file does not contain mandatory term.
349	Definition file is empty.
350	Thesaurus dictionary or definition file as specified via input parameter does not exist.
351	Syntax errors in definition file.
352	The Relationship was specified incorrectly.
352	The Relationship number was out of range.
360	An invalid single character masking was used.
361	An invalid multiple character masking was used.
362	Operator arity is smaller than number of operands given in query.
363	Operator value out of range defined by ItlEnOperator enumeration.
364	Value for rank formula out of enumeration range.
365	Number identifying proximity segment is out of range.

366	Query is under construction and cannot be redefined or reset.
367	Scope given as previous search result denotes empty result.
368	Invalid call requesting to add field names before setting the first one.
369	Invalid search flag requesting an invalid comparison with index content is ignored. If, for example, a case-sensitive comparison was requested for an index that was build in a case insensitive manner, this reason code is shown in the error information.
370	Masking of strings is not supported for Thai or DBCS languages.
371	No valid query input. For example, the search terms is available.
372	Invalid comparison operations requested.
373	Invalid comparison operations requested.
374	Search index handle was requested for an empty index.
375	The combination of operator and requested operator mode is not supported.
380	Search result is incomplete, search was discontinued due to threshold.
381	Index lookup revealed that query contained stopwords.
401	The operation cannot be performed because the Text Search Engine detected a corruption in the index files used for field/attribute name mapping.
402	The operation cannot be performed because the Text Search Engine detected an invalid field or attribute name.
403	The operation cannot be performed because the given field or attribute name is unknown.
404	The limit of different attributes for one of the attribute types or of different fields has been exceeded for this index.
500	The document/data contains an invalid character sequence (in a UTF8, UTF16, or DBCS source).
501	The code page converter was in error.
502	The document/data contains an incomplete character sequence (in a UTF8, UTF16, or DBCS source).
503	The code page converter has an invalid descriptor.

600	The XML document contains an asynchronous entity. For example, an unquoted XML attribute value.
602	Invalid character reference, (for example, or).
603	Invalid binary entity reference.
604	XML Parser Expat could not be created.
605	An attribute name in tag must be unique.
607	XML Parser found an invalid external entity reference.
608	Documents includes an incorrect token, like missing a < or >.
609	XML documents must have an enclosing tag, and after this enclosing end tag no text is allowed.
610	A processing instruction is not allowed at its position. For example, the first processing instruction is not the prolog <?xml .. ?>.
611	An element is a sequence of start tag, content, and end tag. This error occurred, for example, from the sequence ″<s> text /s>″, because the end tag is not correct.
612	Memory allocation failed in XML parser.
614	Invalid parameter entity reference.
615	A non-complete character, maybe only the first byte of a 2-byte UTF8 character.
616	Recursive entity reference.
617	XML Syntax error; for example text outside the enclosing start and end tag.
618	Every start tag needs a matching end tag.
619	Unclosed cdata section.
620	Unclosed token; for example text after the last token in a document.
621	There is an entity in the document that could not be resolved.
622	Unexpected error.
631	Could not parse field or attribute information in meta-tag. Tag must have the format <meta name=″abc″ content=″xyz″>; maybe attributes name or content of the meta-tag not correct.
632	The entity could not be transformed to a character.
650	Different field definitions begin with the same start tag.

651	One start tag contains another, so the tags are ambiguous.
652	If a field and an attribute use the same start tag, then they must use the same end tag or both no end tag.
653	A field not yet closed if the document ends.
654	No document model is specified for the structured format. The document will be parsed as plain text document without field or attribute informations.
670	The operation could not be performed, because it requires the ″Outside In″ (TM) libraries, which could not be found.
671	The operation could not be performed, because a required procedure from the ″Outside In″ (TM) libraries could not be loaded. Probably the libraries are outdated or corrupted.
672	An error occurred while the document was processed with ″Outside In″.

Appendix J. Troubleshooting

DB2 Net Search Extender provides the following information for tracing faults.

Tracing faults

If you need to report an error to an IBM representative, you may be asked to switch on tracing so that information can be written to a file that can be used for locating the error.

As system performance is affected when tracing is switched on, only use the trace facility when directed by an IBM Support Center representative, or by your technical support representative.

To turn tracing on, use the DB2 facility:

```
db2trc on
```

See the *DB2 UDB Command Reference* documentation for further information.

To receive information specific to Net Search Extender, a mask with component `in 96` can be used:

```
db2trc on -m *.*.96.*.*
```

In the case of severe errors, it may also help to look in `db2diag.log`.

© Copyright IBM Corp. 1995, 2003

Appendix K. Data Link messages

Errors from the Data Link will not lead to Net Search Extender errors and, therefore, a termination of the indexing process. Ensure to check the event log and take care of not indexed documents. You need to manually take care that those are reindexed if required.

Table 17. Data Link warning messages

Number	Data Link Message
01H90=	CTEDL - Error setting the return Blob value.
01H91=	CTEDL - DataLink I/O Operation timed out.
01H92=	CTEDL - The character encoding is not supported.
01H93=	CTEDL - Unsupported DataLink scheme.
01H94=	CTEDL - Error creating instance of datatype BLOB.
01H95=	CTEDL - Error setting proxy information.
01H96=	CTEDL - UNC scheme only valid on Windows but the OS could not be determined.
01H97=	CTEDL - UNC scheme only valid on Windows OS.
01H98=	CTEDL - DFS scheme only valid on AIX.
01H99=	CTEDL - DFS scheme only valid on AIX but the OS could not be determined.
01H01=	CTEDL - Error determines the port number in the url string.
01H02=	CTEDL - Unknown Datalink scheme detected.
01H03=	CTEDL - Could not establish connection.
01H00=	CTEDL - Error during execution of DataLink UDF.
01H80=	CTEDL - The DataLink file ″{0}″ could not be found.
01H81=	CTEDL - Unauthorized DataLink file access to ″{0}″.
01H82=	CTEDL - Unexpected end of file or end of stream reached for ″{0}″.
01H83=	CTEDL - DataLink file ″{0}″ is not readable.
01H85=	CTEDL - DataLink URL scheme ″{0}″ requires file name.
01H86=	CTEDL - No connection to DataLink file server ″{0}″ established.
01H60=	CTEDL - Bad HTTP Request - malformed DataLink URL syntax.
01H61=	CTEDL - Unauthorized DataLink request - user authentication required.
01H62=	CTEDL - DataLink access requires payment.
01H63=	CTEDL - Forbidden access for DataLink URL.

© Copyright IBM Corp. 1995, 2003

Table 17. Data Link warning messages (continued)

01H64=	CTEDL - File not found on the DataLink server.
01H65=	CTEDL - The requested method is not allowed for the DataLink resource.
01H66=	CTEDL - Request not acceptable.
01H67=	CTEDL - Proxy Authentication Required.
01H68=	CTEDL - Client request timeout.
01H69=	CTEDL - Conflict with the current state of the DataLink resource.
01H10=	CTEDL - The DataLink resource is no longer available at the server.
01H11=	CTEDL - A content length must be specified to accept the request.
01H12=	CTEDL - The precondition given in the header field evaluated to false.
01H13=	CTEDL - The requested DataLink entity is too large.
01H14=	CTEDL - The requested DataLink URL is too long.
01H15=	CTEDL - Unsupported MIME type.
01H16=	CTEDL - Range Request not satisfiable.
01H17=	CTEDL - Expectation failed.
01H18=	CTEDL - Request to DataLink URL gets no content length information.
01H20=	CTEDL - HTTP response not valid.
01H70=	CTEDL - Internal DataLink server error.
01H71=	CTEDL - Functionality not supported by the DataLink server.
01H72=	CTEDL - Bad Gateway.
01H73=	CTEDL - Service unavailable - DL server temporarily overloaded or maintained.
01H74=	CTEDL - Gateway timeout.
01H75=	CTEDL - HTTP version not supported.
01H30=	CTEDL - Try to establish a socket connection - error in underlying protocol.
01H31=	CTEDL - Could not establish a route to the DataLink Server {0}.
01H32=	CTEDL - Could not connect the socket to the remote address {0}.
01H33=	CTEDL - Could not bind the socket to the local address.
01H34=	CTEDL - IP address of DataLink server {0} could not be determined.
01H35=	CTEDL - Unknown service exception - no MIME type support.
01H36=	CTEDL - Malformed URL '{0}' - no supported protocol or DL URL could not be parsed.

Appendix L. Thesaurus supported CCSIDs

The following CCSIDs are supported by the thesaurus:

CCSIDs

819	Latin 1
850	PC Data Latin 1
874	Thai
932	Combined Japanese
943	Combined Japanese
950	Combined Traditional Chinese
954	Japanese
970	Combined Korean
1208	UTF 8
1250	Latin 2
1252	Latin 1
1253	Czech
1254	Turkish
1255	Hebrew
1256	Arabic
1258	Vietnamese
1363	Combined Korean
1381	Combined Simplified Chinese
1383	Chinese (simplified), combined SBCS/DBCS
1386	Chinese (simplified), combined SBCS/DBCS
5039	Japanese (combined SBCS/DBCS)

To compile the thesaurus definition file, see "DB2EXTTH (Utility)" on page 145.

© Copyright IBM Corp. 1995, 2003

Appendix M. Messages returned by the thesaurus tools

ADM_MSG_INVALID_CCSID

Explanation: Invalid CCSID specified.

The requested code page is not supported.

ITL_THES_MSG_DEFFILE_MISSING

Explanation: Parameter error *file name*. The thesaurus definition file does not exist.

ITL_THES_MSG_NONAME_ERROR

Explanation: Parameter error. No thesaurus definition file name specified.

ITL_THES_MSG_PATHLEN_ERROR

Explanation: Parameter error *file name*. The thesaurus definition file path is too long. The path length must not exceed the maximum length supported for directory names in the operating system.

ITL_THES_MSG_NAMELEN_ERROR

Explanation: Parameter error *file name*. The thesaurus definition file name is too long.

ITL_THES_MSG_NO_TARGET_DIR_ERROR

Explanation: Parameter error. No target directory specified.

ITL_THES_MSG_UNEXPECTED_ERROR

Explanation: Internal unexpected error.

ITL_THES_MSG_PARAMETER_ERROR

Explanation: Internal parameter error.

ITL_THES_MSG_FILE_OPEN_ERROR

Explanation: Could not open file *file name*.

ITL_THES_MSG_FILE_REACHED_END

Explanation: Unexpected end of file in *thesaurus definition file*.

There is an error in the definition file.

ITL_THES_MSG_FILE_READ_ERROR

Explanation: Could not read file *file name*.

ITL_THES_MSG_FILE_WRITE_ERROR

Explanation: Could not write file *file name*.

ITL_THES_MSG_FILE_ACCESS_ERROR

Explanation: Could not access file *file name*.

ITL_THES_MSG_FILE_REMOVE_ERROR

Explanation: Could not remove file *file name*.

ITL_THES_MSG_FILE_RENAME_ERROR

Explanation: Could not rename file *file name 1* to *file name 2*.

ITL_THES_MSG_FILE_CLOSE_ERROR

Explanation: Could not close file *file name*.

ITL_THES_MSG_FILE_EOF_ERROR

Explanation: Unexpected end of file in *file name*.

Error in definition file.

ITL_THES_MSG_MEMORY_ERROR

Explanation: Memory error.

ITL_THES_MSG_BUFFER_OVERFLOW

Explanation: Buffer overflow.

© Copyright IBM Corp. 1995, 2003

ITL_THES_MSG_LOCKING_ERROR

Explanation: Could not lock dictionary *file name*.

ITL_THES_MSG_LOCKED

Explanation: Thesaurus dictionary *dictionary name* is in use.

ITL_THES_MSG_OUTFILE_EXIST

Explanation: Output file *file name* already exists.

ITL_THES_MSG_DICT_INTEGRITY_ERROR

Explanation: Integrity of dictionary *dictionary name* is lost.

The thesaurus dictionary file is corrupted.

ITL_THES_MSG_DICT_VERSION_ERROR

Explanation: Dictionary *dictionary name* version error.

The thesaurus dictionary was created with an incompatible earlier version.

ITL_THES_MSG_DICT_NOT_EXIST

Explanation: Thesaurus dictionary *dictionary name* does not exist.

ITL_THES_MSG_DICT_EXIST

Explanation: Thesaurus dictionary *dictionary name* already exists.

Cannot be overwritten.

ITL_THES_MSG_NORMALIZE_ERROR

Explanation: Error in normalizing a term.

Error in the thesaurus definition file.

ITL_THES_MSG_INTERNAL_ERROR

Explanation: Internal error.

ITL_THES_MSG_INPUT_ERROR

Explanation: Error in the thesaurus definition file *file name* at line *line number*.

ITL_THES_MSG_ERROR_IN_FILE

Explanation: Error in file *file name*.

ITL_THES_MSG_IE_EMPTY

Explanation: The thesaurus definition file *file name* is empty.

ITL_THES_MSG_IE_BLOCK_START

Explanation: No block starting line was found in file *file name* at line *line number*.

ITL_THES_MSG_IE_REL_SYNTAX

Explanation: Relationship is specified incorrectly in *file name* at line *line number*.

ITL_THES_MSG_IE_USER_DEF

Explanation: Relationship is specified incorrectly in *file name* at line *line number*.

ITL_THES_MSG_IE_USER_DEF_DOMAIN

Explanation: A relationship number is out of range in *file name* at line *line number*.

ITL_THES_MSG_IE_NO_TERM

Explanation: No terms are defined in *file name* at line *line number*.

ITL_THES_MSG_IE_TERM_LEN

Explanation: A thesaurus term is longer than 64 characters.

ITL_THES_MSG_IE_STRENGTH_SYNTAX

Explanation: A strength value is specified incorrectly.

Syntax: After the term, type [:20] for a strength of 20.

ITL_THES_MSG_IE_STRENGTH_DOMAIN

Explanation: Strength is out of range.

Valid values are 1 - 100; the default is 100.

Messages returned by the thesaurus tools

Appendix N. Windows system errors

The following is a list of Windows system errors:

System errors

1	Incorrect function.
2	The system cannot find the file specified.
3	The system cannot find the path specified.
4	The system cannot open the file.
5	Access is denied.
6	The handle is invalid.
8	Not enough storage is available to process this command.
14	Not enough storage is available to complete this operation.
15	The system cannot find the drive specified.
29	The system cannot write to the specified device.
30	The system cannot read from the specified device.
32	The process cannot access the file because it is being used by another process.
36	Too many files opened for sharing.
38	Reached the end of the file.
39	The disk is full.
80	The file exists.
82	The directory or file cannot be created.
100	Cannot create another system semaphore.
101	The exclusive semaphore is owned by another process.
102	The semaphore is set and cannot be closed.
103	The semaphore cannot be set again.
104	Cannot request exclusive semaphores at interrupt time.
105	The previous ownership of this semaphore has ended.
110	The system cannot open the device or file specified.

© Copyright IBM Corp. 1995, 2003

111	The file name is too long.
112	There is not enough space on the disk.
121	The semaphore timeout period has expired.
126	The specified module could not be found.
127	The specified procedure could not be found.
147	Not enough resources are available to process this command.
155	Cannot create another thread.
161	The specified path is invalid.
164	No more threads can be created in the system.
170	The requested resource is in use.
183	Cannot create a file when that file already exists.
187	The specified system semaphore name was not found.
206	The filename or extension is too long.
267	The directory name is invalid.
288	Attempt to release mutex not owned by caller.
298	Too many posts were made to a semaphore.
998	Invalid access to memory location.
1051	A stop control has been sent to a service that other running services are dependent on.
1052	The requested control is not valid for this service.
1053	The service did not respond to the start or control request in a timely fashion.
1054	A thread could not be created for the service.
1055	The service database is locked.
1056	An instance of the service is already running.
1057	The account name is invalid or does not exist.
1058	The service cannot be started, either because it is disabled or because it has no enabled devices associated with it.
1059	Circular service dependency was specified.
1060	The specified service does not exist as an installed service.
1061	The service cannot accept control messages at this time.
1062	The service has not been started.

1063	The service process could not connect to the service controller.
1064	An exception occurred in the service when handling the control request.
1066	The service has returned a service-specific error code.
1067	The process terminated unexpectedly.
1068	The dependency service or group failed to start.
1069	The service did not start due to a logon failure.
1070	After starting, the service hung in a start-pending state.
1071	The specified service database lock is invalid.
1072	The specified service has been marked for deletion.
1073	The specified service already exists.
1078	The name is already in use as either a service name or a service display name.
1079	The account specified for this service is different from the account specified for other services running in the same process.
1082	No recovery program has been configured for this service.
1154	One of the library files needed to run this application is damaged.
1219	The credentials supplied conflict with an existing set of credentials.
1242	The service is already registered.
1243	The specified service does not exist.
1244	The operation being requested was not performed because the user has not been authenticated.
1245	The operation being requested was not performed because the user has not logged on to the network. The specified service does not exist.
1392	The file or directory is corrupted and unreadable.
1455	The paging file is too small for this operation to complete.
1793	The user's account has expired.

Appendix O. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

© Copyright IBM Corp. 1995, 2003

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both.

AIX
DB2
DB2 Extenders
Informix
DB2 Universal Database
IBM DRDA
z/OS

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary defines terms and abbreviations used in this manual. If you do not find the term you are looking for, refer to the index or to the *Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

access function. A user-provided function that converts the data type of text stored in a column to a type that can be processed by DB2 Net Search Extender.

B

Boolean search. A search in which one or more search terms are combined using Boolean operators.

C

catalog view. A view of a system table created by DB2 Net Search Extender Text for administration purposes. A catalog view contains information about the tables and columns that have been enabled for use by DB2 Net Search Extender Text.

CCSID. Coded Character Set Identifier.

code page. An assignment of graphic characters and control function meanings to all code points. For example, assignment of characters and meanings to 256 code points for an 8-bit code.

command line processor. A program called db2text that:

- Allows you to enter DB2 Net Search Extender commands
- Processes the commands
- Displays the result

count. A keyword used to specify the number of levels (the depth) of terms in the thesaurus that are to be used to expand the search term for the given relation.

D

DBCS. Double-byte character set.

disable. To restore a database to its condition before it was enabled for DB2 Net Search Extender Text by removing the items created during the enabling process.

document. See *text document*.

document model. The definition of the structure of a document in terms of the sections that it contains. A document model makes DB2 Net Search Extender aware of the sections within documents when indexing. A document model lists the markup tags that identify the sections. For each tag you can specify a descriptive section name for use in queries against that section. You can specify one or more document models in a document models file.

E

enable. To prepare a database for use by DB2 Net Search Extender.

escape character. A character indicating that the subsequent character is not to be interpreted as a *masking character*.

expand. The action of adding to a search term additional terms derived from a thesaurus.

F

format. The type of a document, such as ASCII, or HTML.

© Copyright IBM Corp. 1995, 2003

free-text search. A search in which the search term is expressed as free-form text – a phrase or a sentence describing in natural language the subject to be searched for.

function. See *access function*.

fuzzy search. A search that can find words whose spelling is similar to that of the search term.

H

hybrid search. A combined *Boolean search* and *free-text search*.

I

index. To extract significant terms from text, and store them in a *text index*.

index characteristics. Properties of a *text index* determining:

The frequency with which the index is updated

When the first index update is to occur

L

log table. A table created by DB2 Net Search Extender containing information about which text documents are to be indexed. *Triggers* are used to store this information in a log table whenever a document in an enabled text column is added, changed, or deleted.

M

masking character. A character used to represent optional characters at the front, middle, and end of a search term. Masking characters are normally used for finding variations of a term in a precise index.

match. The occurrence of a search term in a text document.

P

periodic indexing. Indexing at predetermined time intervals, expressed in terms of the day, hour, and minute, and the minimum number of documents names that must be listed in the *log table* for indexing, before indexing can take place.

R

retrieve. To find a text document using a search argument in one of DB2 Net Search Extender's search functions.

S

SBCS. Single-byte character set.

Score. An absolute value of type DOUBLE between 0 and 1 that indicates how well a document meets the search criteria relative to the other found documents. The value indicates the number of matches found in the document in relation to the document's size.

search argument. The conditions specified when making a search, consisting of one or several search terms, and search parameters.

T

text column. A column containing *text documents*.

text document. Text of type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB datatypes, stored in a DB2 table.

text index. A collection of significant terms extracted from text documents. Each term is associated with the document from which it was extracted. A significant improvement in search time is achieved by searching in the index rather than in the documents themselves.

tracing. The action of storing information in a file that can later be used in finding the cause of an error.

trigger. A mechanism that automatically adds information about documents that need to be indexed to a *log table* whenever a document is added, changed, or deleted from a text column.

U

UDF. User-defined function.

UDT. User-defined type.

update frequency. The frequency with which a text index is updated, expressed in terms of the day, hour, and minute, and the minimum number of document names that must be listed in the *log table* for indexing, before indexing can take place.

user-defined type (UDT). A data type created by a user of DB2, in contrast to a data type provided by DB2 such as LONG VARCHAR.

user-defined function (UDF). An SQL function created by a user of DB2, in contrast to an SQL function provided by DB2.

W

wildcard character. See *masking character*.

Index

Special characters

A

B

C

D

© Copyright IBM Corp. 1995, 2003

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

IBM DB2 Universal Database
Net Search Extender
Administration and User's Guide
Version 8.1

Publication No. SH12-6740-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SH12-6740-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

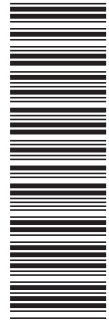
Fold and Tape **Please do not staple** Fold and Tape

SH12-6740-02

Cut or Fold
Along Line

Part Number: CT202NA

Printed in Denmark by IBM Danmark A/S



SH12-6740-02