IBM® DB2® Universal Database

Text Extender
Administration and Programming

Version 7.1.0.1

IBM® DB2® Universal Database

Text Extender
Administration and Programming

Version 7.1.0.1

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

Contents

About this book

This book describes how to use DB2 Text Extender to prepare and maintain a DB2(R) database for retrieving text data. It also describes how you can use DB2 Text Extender-provided SQL functions and application programming interfaces (APIs) to access and manipulate these types of data. By incorporating DB2 Text Extender's functions in your program's SQL statements, and incorporating APIs,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. Guide"

This part gives an overview of DB2 Text 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 Text Extender and want to learn how to use the DB2 Text Extender functions and APIs to search for text.

"Part 2. Reference"

This part presents reference information for DB2 Text Extender functions, APIs, commands, and diagnostic information such as messages and codes.

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

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).

```
►►──required item──────────────────────────────────────────────────────►◄
```

- Optional items appear below the main path.

```
►►──────────────────────────────────────────────────────────────────────►◄
     └─optional item─┘
```

- 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.

```
►►──┬─required choice1─┬─────────────────────────────────────────────────►◄
    └─required choice2─┘
```
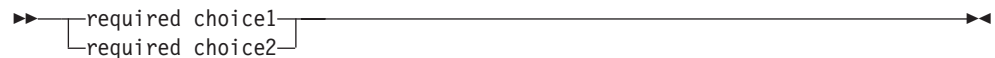
  If choosing none of the items is an option, the entire stack appears below the main path.

```
►►──────────────────────────────────────────────────────────────────────►◄
    ├─optional choice1─┤
    └─optional choice2─┘
```

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

```
      ┌──────────────────┐
►►────▼──────────────────┴───────────────────────────────────────────────►◄
        ├─optional choice1─┤
        └─optional choice2─┘
```

- 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.

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 any of the following methods to provide comments:

- Send your comments from the Web. Visit the Web site at:

  http://www.ibm.com/software/data/db2/extenders

  The Web site has a feedback page that you can use to enter and send 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).

- Fill out a Readers' Comments form at the back of this book and return it by mail, by fax, or by giving it to an IBM representative. The mailing address is on the back of the form. The fax number is +49-(0)7031-16-4892.

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.

Summary of Changes

Changes to this document for Release 7.1.0.1 include:

- **Support of additional languages for indexing and search**

  The additional languages are: Albanian, Belorussian, Bulgarian, Croatian, Estonian, Greek, Hindi, Latvian, Lithuanian, Macedonian, Romanian, Serbian (latin), Serbian (cyrillic), Slovenian, Turkish, Ukrainian, and Vietnamese. For a complete list of supported languages, see "Dictionaries, stop-word lists, abbreviation lists, and language parameters" on page 43.

- **New maintenance command TXSERVICE**

  This command allows you to copy information about the current Text Extender and DB2 setup to a file. If there are any problems with DB2 Text Extender, you can use this file to assist IBM support.

Summary of Changes

Part 1. Guide

Chapter 1. An overview of DB2 Text Extender

DB2 Text Extender is one of a family of DB2 Extenders. It enables programmers to include SQL queries for text documents in their applications.

The other Extenders can search for images, video and voice data, and can handle XML documents and spatial data.

DB2 Text Extender adds the power of full-text retrieval to SQL queries by making use of features available in DB2 UDB that let you store unstructuredtext documents in databases.

DB2 Text Extender offers DB2 UDB users and application programmers a fast, versatile, and intelligent method of searching through such text documents. DB2 Text Extender's strength lies in its ability to search through many thousands of large text documents at high speed, finding not only what you directly ask for, but also word variations and synonyms.

You are not restricted to searching only in text documents stored in DB2 UDB databases, you can also search in text documents stored in files.

At the heart of DB2 Text Extender is IBM's high-performance linguistic search technology described in "Chapter 15. Linguistic processing for linguistic and precise indexes" on page 221. It allows your applications to access and retrieve text documents in a variety of ways. Your applications can:

- Search for documents that contain specific text, synonyms of a word or phrase, or sought-for words in proximity, such as in the same sentence or paragraph.
- Do wildcard searches, using front, middle, and end masking, for word and character masking.
- Search for documents of various languages in various document formats.
- Make a "fuzzy" search for words having a similar spelling as the search term. This is useful for finding words even when they are misspelled.
- Make a free-text search in which the search argument is expressed in natural language.
- Search for words that sound like the search term.

You can integrate your text search with business data queries. For example, you can code an SQL query in an application to search for text documents that are created by a specific author, within a range of dates, and that contain a particular word or phrase. Using the DB2 Text Extender programming interface, you can also allow your application users to browse the documents.

By integrating full-text search into DB2 UDB 's SELECT queries, you have a powerful retrieval function. The following SQL statement shows an example:

```
SELECT * FROM MyTextTable
  WHERE version = '2'
  AND DB2TX.CONTAINS (
              DB2BOOKS_HANDLE,
              '"authorization"
                IN SAME PARAGRAPH AS "table"
                AND SYNONYM FORM OF "delete"') = 1
```

DB2TX.CONTAINS is one of several DB2 Text Extender search functions. DB2BOOKS_HANDLE is the name of a handle column referring to column DB2BOOKS that contains the text documents to be searched. The remainder of the statement is an example of a search argument that looks for `authorization`, occurring in the same paragraph as `table`, and `delete`, or any of `delete`'s synonyms.

DB2 Text Extender in the DB2 client/server environment

Figure 1 shows how DB2 Text Extender is integrated into the DB2 client/server environment.
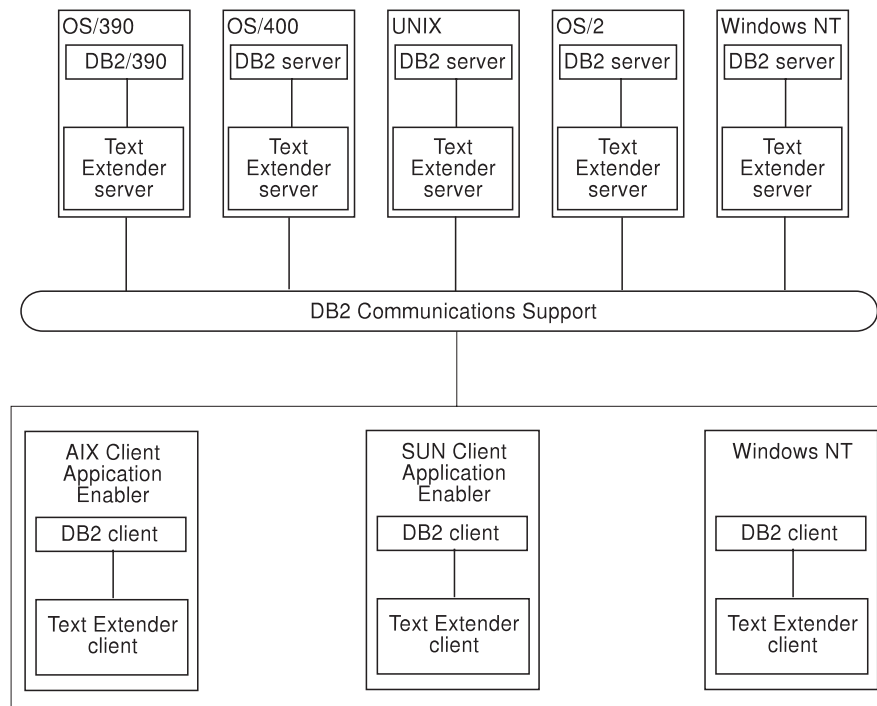
Figure 1. Integration of DB2 Text Extender into the DB2 client/server environment

For a list of the DB2 Communications Support protocols (such as TCP/IP or NETBIOS) for a client, see the *DB2 Quick Beginnings Guide* for the appropriate platform.

The main part of DB2 Text Extender is installed on the same machine as the DB2 server. Only one DB2 Text Extender server instance can be installed with one DB2 server instance.

A DB2 Text Extender installation is flexible and can comprise:
- One or several DB2 Text Extender servers on any of the operating systems shown in Figure 1, where UNIX includes AIX(R), SUN-Solaris, and HP-UX workstations.
- AIX, SUN-Solaris, Windows NT, and Windows 2000 clients with access to one or several remote DB2 Text Extender servers.
- AIX clients containing a local server and having access to remote servers.

Figure 2 on page 5 shows a typical DB2 Text Extender configuration. To run DB2 Text Extender from a client, you must first install a DB2 client and some DB2 Text

Extender utilities. These utilities constitute the DB2 Text Extender "client" although it is not a client in the strict sense of the word. The client communicates with the server via the DB2 client connection.
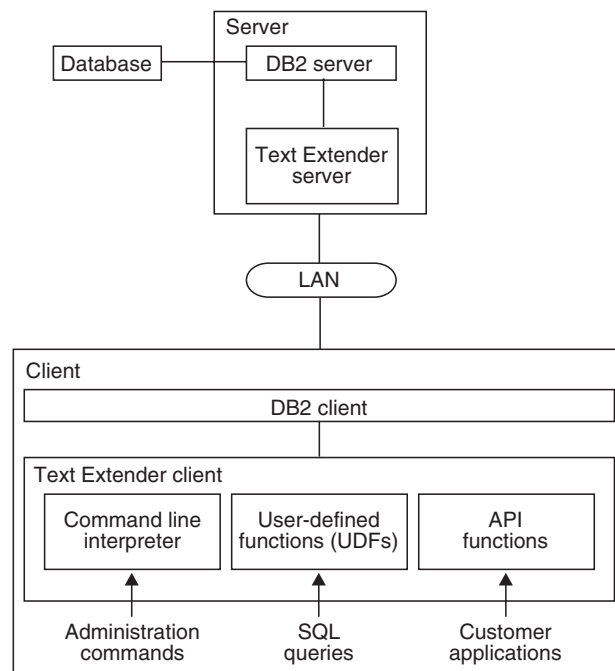
Figure 2. A DB2 Text Extender configuration

DB2 Text Extender has the following main components:

- **A command line interpreter.** Commands are available that let you prepare text in columns for searching, and maintain text indexes.

- **SQL functions.** Functions are available that you can include in SQL queries for searching in text, and finding, for example, the number of times the search term occurs in the text. For clarity, the figure shows the SQL functions on the client because they can be used as part of an SQL query. In fact, they are part of the server installation and are executed there. However, these functions can be used from any DB2 client without the need to install the DB2 Text Extender client.

- **An application programming interface (API)** consisting of functions that can be called in C programs for searching in text and displaying the search results.

> **Tip**
>
> You only need to install the DB2 Text Extender client if you want to undertake Text Extender administrative tasks from the client, or use the API.
>
> If you need only the search capability on the client using DB2 UDB SQL statements, you do not need to install the DB2 Text Extender client. All communication is handled by DB2 UDB and the DB2 Text Extender search engine runs only on the server.

DB2 Text Extender in the partitioned database environment

Figure 3 shows how DB2 Text Extender is integrated into the DB2 partitioned database environment.
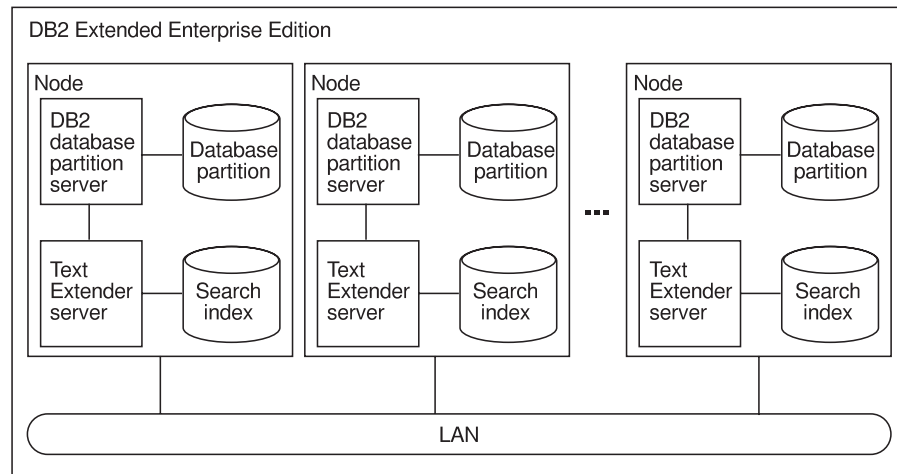
Figure 3. Integration of DB2 Text Extender into the DB2 partitioned database environment

Each database partition server, known as a *node*, can be assigned to a separate machine, or they can be located on a single machine. Database partition servers located on separate machines are known as *physical nodes*. Those located on a single machine are known as *logical nodes*. DB2 Text Extender supports both configurations.

A *nodegroup* is a named subset of one or more database partition servers. If a DB2 Text Extender-enabled table is assigned to a multipartition nodegroup, the text indexes that are created by DB2 Text Extender are located on the same node as each corresponding table partition. This ensures that data does not have to be moved between nodes during indexing.

A DB2 Text Extender server runs on every partition of a nodegroup and manages the indexed rows of the corresponding partition. You interact with DB2 Text Extender through one database partition. Each database partition can be used to issue DB2 Text Extender requests. The requests are routed to all the involved database partitions.

Chapter 2. Installation, setting up, and maintenance

This chapter describes how to install DB2 Text Extender in UNIX and Windows systems. To install DB2 Text Extender for OS/2, see the installation README file on the CD-ROM.

Installation in AIX, Sun-Solaris, and HP-UX systems

The DB2 Text Extender Version 7.1.0.1 for AIX, Sun-Solaris, and HP-UX can operate in a nonpartitioned database environment. For AIX and Sun-Solaris it can also exploit the partitioned database support provided by DB2 Extended Enterprise Edition.

These are the installation steps:

1. Install the product components in the target machine.
2. Establish the DB2 Text Extender instance.
3. Start the DB2 Text Extender server.

Installation requirements

Before you install DB2 Text Extender, ensure you meet the following installation requirements for their server and client components.

Server requirements
The server components require the installation of DB2 Version 7.1 (or above).

Client requirements
The client components require installation of the DB2 Version 7.1 (or above) Personal Edition, DB2 Version 7.1 (or above) Client Application Enabler, or DB2 Version 7.1 (or above) Software Developer's Kit.

Step 1 for AIX: Install the product components

During installation, a group called `smadmin` is created automatically.

1. Log in at the target client or server machine as the root user.
2. Use the System Management Interface Tool (SMIT) to run `installp` from a prompted interface, or enter the `installp` command.
3. Enter the `smit install_latest` command. The Software Installation and Maintenance Tool menu opens.
4. Type in the location of the package in the `INPUT device / directory for software` field.
5. Click the `DO` button or press ENTER. This confirms the installation directory.
6. Identify in the `SOFTWARE TO INSTALL` field whether the client or server components are to be installed.
7. Click the `DO` button or press ENTER. You are prompted for confirmation of the installation parameters. To confirm, press ENTER.
8. Identify in the `SOFTWARE TO INSTALL` field which (if any) dictionaries you want to install in the client. If you do not specify a dictionary, the US English dictionary is installed.

   Table 1 on page 8 lists the DB2 Text Extender dictionaries. To install all the dictionaries in the client, enter `db2tx_07_01.dic` in the `SOFTWARE TO INSTALL` field.

Installation in UNIX systems

Table 1. DB2 Text Extender dictionaries

Dictionary	Language
db2tx_07_01.dic.Ar_AA	Arabic
db2tx_07_01.dic.Da_DK	Danish
db2tx_07_01.dic.Nl_NL	Dutch
db2tx_07_01.dic.En_GB	English (U.K.)
db2tx_07_01.dic.En_US	English (U.S.)
db2tx_07_01.dic.Fi_FI	Finnish
db2tx_07_01.dic.Fr_FR	French
db2tx_07_01.dic.Fr_CA	French (Canada)
db2tx_07_01.dic.De_DE	German
db2tx_07_01.dic.De_CH	German (Swiss)
db2tx_07_01.dic.Iw_IL	Hebrew
db2tx_07_01.dic.Is_IS	Icelandic
db2tx_07_01.dic.It_IT	Italian
db2tx_07_01.dic.No_NO	Norwegian
db2tx_07_01.dic.Pt_PT	Portuguese
db2tx_07_01.dic.Ru_RU	Russian
db2tx_07_01.dic.Es_ES	Spanish and Catalan
db2tx_07_01.dic.Sv_SE	Swedish
db2tx_07_01.dic.Th_TH	Thai

9. Log out.

Step 1 for Sun-Solaris: Install the product components

During installation, a group called `smadmin` is created automatically.

1. Log in at the target client or server machine as the root user.
2. Use `tar` to get the installable package out of the fixpack file:

   ```
   tar -xvf IBMdb2tx.tar
   ```
3. Use `pkgadd` to transfer the files from your directory to the target machine. You can choose one of two ways to do this:
   - Install the DB2 Text Extender client and server with the US English dictionary:

     ```
     pkgadd -d . db2txclnt db2txsrv db2txcom db2txmsgs db2txmsc db2txdius
     ```
   - Indicate if you want to install a DB2 Text Extender client or server, and which dictionaries you want to install:

     ```
     ./db2tx_install
     ```
4. Log out.

Step 1 for HP-UX: Install the product components

During installation, a group called `smadmin` is created automatically.

1. Log in at the target client or server machine as the root user.
2. Use `tar` to get the installable package out of the fixpack file:

   ```
   tar -xvf b2tx.tar
   ```
3. Run the swinstall program:

```
swinstall
```

This opens the Software Selection window and the Specify Source window.

4. If necessary, change the Source Host Name.

5. Enter your `tar` directory using `IBMdb2tx` as the value for the Source Depot Path field.

6. In the Change Software View field, select `Products`.

7. Select `OK` to return to the Software Selection window, which contains a list of the available software to install.

8. Now do one of the following:
   - To install DB2 Text Extender with all dictionaries and with documentation in all provided languages, highlight the `DB2TX Product`.
   - To install parts of DB2 Text Extender:

      a. Choose `Open Item` from the `Actions` menu.

      b. For every fileset you want to install, highlight it and choose `Choose Mark for Install` from the `Actions` menu. You can choose to install the DB2 Text Extender server or client:

      ```
      Text Extender Server  -  DB2TX.db2txsrv
      Text Extender Client  -  DB2TX.db2txclnt
      ```

      c. Choose the dictionaries (if any) you want to install. Table 2 lists the DB2 Text Extender dictionaries.

Table 2. DB2 Text Extender dictionaries

Dictionary	Language
db2txdiaa	Arabic
db2txdidk	Danish
db2txdinl	Dutch
db2txdigb	English (U.K.)
db2txdius	English (U.S.)
db2txdifi	Finnish
db2txdifr	French
db2txdica	French (Canada)
db2txdide	German
db2txdich	German (Swiss)
db2txdiil	Hebrew
db2txdiis	Icelandic
db2txdiit	Italian
db2txdino	Norwegian
db2txdipt	Portuguese
db2txdies	Russian
db2txdiru	Spanish and Catalan
db2txdise	Swedish
db2txdith	Thai

9. Choose `Install` from the `Actions` menu to begin product installation and open the Install Analysis window.

10. Select the OK button in the Install Analysis window when the Status field displays a Ready message.

11. Select the Yes button at the Confirmation window to confirm that you want to install the DB2 for HP-UX software products.

12. View the Install window to read processing data while the software is being installed. Installation is completed when the Status field indicates Ready and the Note Window opens.

    The swinstall program loads the fileset, and runs the control scripts for the fileset.

    The HP-UX operating system provides detailed help for swinstall. Help is available by typing:

    ```
    man swinstall
    ```

13. Log out.

Step 2: Establish the DB2 Text Extender instance

1. Ensure you are active as the root user.

2. For AIX, enter:

   ```
   cd /usr/lpp/db2ext_07_01/instance
   ```

   For Solaris and HP-UX, enter:

   ```
   cd /opt/IBMdb2tx/V7.1/instance
   ```

3. If you have an existing instance, enter:

   ```
   ./txiupdt instanceid db2instanceid
   ```

   to update your instance with the new files.

   Otherwise, to create a new DB2 Text Extender instance, enter:

   ```
   ./txicrt instanceid db2instanceid
   ```

   where *instanceid* is an existing DB2 instance user ID or a user ID to be associated with a DB2 instance ID.

   Running txicrt creates the /u/*instanceid*/db2tx directory. Do not create additional files or directories under /u/*instanceid*/db2tx; these files will be lost if the instance is deleted.

   Additional parameters are required if you are using a partitioned database environment. For further information, see "TXICRT" on page 146.

4. Log out.

Step 3: Start the DB2 Text Extender server

1. Log in as *instanceid*.

2. Stop and restart DB2.

3. To start the DB2 Text Extender server, enter:

   ```
   txstart
   ```

4. Log out.

Installation in Windows 95, Windows 98, Windows NT, and Windows 2000

DB2 Text Extender is installed as part of the DB2 Extenders package.

Server requirements

The server components for the DB2 Extenders can be installed in a Windows NT or Windows 2000 server.

The server components to be operated in a nonpartitioned database environment require the installation of DB2 Version 7.1 (or above).

The server components to be operated in a partitioned database environment require the installation of DB2 Extended Enterprise Edition Version 7.1 (or above).

Client requirements

The client components for the DB2 Extenders can be installed on Windows 95, Windows 98, Windows NT, or Windows 2000.

The client components require installation of the DB2 Version 7.1 (or above) Personal Edition, DB2 Version 7.1 (or above) Client Application Enabler, or DB2 Version 7.1 (or above) Software Developer's Kit for Windows.

Installing the DB2 Extenders

To install the DB2 Extenders in Windows:

1. Insert the CD-ROM into the CD-ROM drive on your client workstation.
2. If the CD autostart feature is enabled, the DB2 Extender LaunchPad automatically starts. Otherwise, click the `Start` button on the task bar. Select RUN. Type:

   ```
   d:\db2ext\winnt95\setup
   ```

   in the open field, where "d" is the CD-ROM drive letter.
3. From the LaunchPad, you can view the DB2 Extender ReadMe file. Look at the installation prerequisites or install the product.
4. Select the `Install` button on the left to begin installation.
5. In the first window, select which version of DB2 Extenders to install: Client/Server for EEE, Client/Server for non-EEE, or Client only.
6. In the next window, choose the folder where the Extenders will be installed.
7. Then select the type of install you want: Typical, Custom, or Compact.

   The **Typical** installation includes the DB2 Extenders client/server, sample files, SDK, US English online documentation, and the US English DB2 Text Extender dictionary.

   The **Compact** installation installs only the client/server and the US English DB2 Text Extender dictionary.

   The **Custom** installation lets you choose from the following components:
   - Select **DB2 Extenders Server (EEE)** if you are installing the DB2 Extenders in a partitioned database environment (DB2 Extended Enterprise Edition). This also installs the **DB2 Extenders Client** component.
   - Select **DB2 Extenders Server (non-EEE)** if you are installing the DB2 Extenders in a nonpartitioned database environment. This also installs the **DB2 Extenders Client** component.
   - Select **DB2 Extenders Client** if you are installing the DB2 Extenders client environment.
   - Select **DB2 Extenders SDK** if you want to install header files and libraries for developing DB2 Extenders applications.
   - Select **Sample Files** if you want to install the sample programs and sample media files.

- Select **Online Documentations** if you want to install the Extender publications in softcopy format. Once you make this selection, you can click the `Change` button in the Description section to see the available language-specific publications. Select the ones that you would like to install from the list.
- Select **Text Extender Dictionaries** if you want to install lingustic support for selected languages for DB2 Text Extender. Once you make the selection, you can click the `Change` button in the Description section to see the available language-specific dictionaries. Select the ones you would like to install from the list. US English is always required.

If you do not have enough disk space for your selected components, press `Back` to reselect the installation directory. Otherwise, press `Next`.

8. If you are installing the Extender server (EEE) in a partitioned database environment, and are on an instance-owning machine, you are prompted to specify a range of TCP/IP ports. The same share name must exist on all nodes associated with the current DB2 instance.

   If you are installing the DB2 Extenders on a machine that is not an instance-owning machine, you are prompted to enter the name of the instance-owning machine. TCP/IP port ranges and data path information will be obtained from the instance-owning machine.

9. Click the `Next` button to install the components you selected.

   When started, the installation program checks that your workstation fulfills the DB2 Extenders software requirements.

10. Shut down and restart the workstation.

Creating and managing instances

You can create and use multiple instances of the DB2 Extenders server. You should create multiple instances if you have created multiple instances of the DB2 server for Windows. Each instance of the DB2 Extenders server is associated with an instance of the DB2 server for Windows.

A default DB2 Extenders instance named DB2 is created when you install a DB2 extender in Windows in a nonpartitioned database environment.

A default DB2 Extenders instance named DB2MPP is created when you install the DB2 Extenders in Windows in a partitioned database environment. In a partitioned database environment, no default instance is created for DB2 Text Extender. To create an instance, use the command `txicrt`. For more information, see "Creating the DB2 Text Extender instance" on page 16.

Preparing a sample database for installation verification

DB2 Text Extender offers utilities for preparing a sample database. They are useful for quickly preparing text for testing DB2 Text Extender's search capabilities immediately after installation, and for general test purposes at any time after that.

These utilities are in the following directories:
- For OS/2(R) and Windows NT, and Windows 2000: *%DMBMMPATH%*\samples
  where *%DMBMMPATH%* is the installation path.
- For AIX, HP-UX, and SUN-Solaris: $HOME/db2tx/samples

To prepare a sample database, either at the server or at a client workstation:

1. At the operating system prompt, run:

   `TXVERIFY database-name [user-id] [password]`

   This command creates and enables a sample database.

2. At the operating system prompt, run:

   `TXSAMPLE database-name [user-id] [password]`

   The user ID and password are required only if you are working from a client workstation. This command does the following:
   Connects to the specified database
   Creates the table `db2tx.sample`
   Imports sample English documents to fill the table
   Enables the text column in the table, with the following index types:
   > precise
   > precise normalized
   > linguistic
   > Ngram
   > Ngram case-enabled

   Waits for the text index to be built.

Configuration

This section describes the DB2 Text Extender environment variables and configuration information. Both of these let you specify default values for many parameters needed by DB2 Text Extender.

Environment variables

The environment variables set the default values of environment parameters. To display the current setting of environment variables, use the GET ENVIRONMENT command described in "Displaying the settings of the environment variables" on page 87.

DB2INSTANCE	DB2 UDB instance name.
DB2DBDFT	Default database name. The name of the DB2 UDB database that is assumed if no database name is specified.
DB2TX_INSTOWNER	DB2 Text Extender instance name. This is the login name of the user that owns the instance. It is required if a DB2 Text Extender client connects to a UNIX server.
DB2TX_INSTOWNERHOMEDIR	
	Instance owner's home directory (UNIX only).

In a Windows environment, the DB2 Text Extender environment variables are specified as system variables in the system properties page. They are added to your system settings during installation.

In an OS/2 environment, the DB2 Text Extender environment variables are added to the `CONFIG.SYS` file during installation.

In a UNIX environment, the DB2 Text Extender environment variables except DB2DBDFT are specified in the following profile shell scripts:

> `db2txcshrc` for the C shell

`db2txprofile` for the Korn shell and the Bourne shell.

When you create an instance, the file `db2txprofile` or `db2txcshrc` is copied to the `.profile`.

When you create an instance, the `db2txprofile` is added to the `.profile` and the `db2profile` of the instance owner. The shell script is stored in $DB2TX_INSTOWNERHOMEDIR/db2tx.

You also need the following environment variables:

LANG　　　　Specifies the language setting

NLSPATH　　Specifies the path to the DB2 Text Extender message file

DB2ENVLIST　List of environment variables that are passed by DB2 to the DB2 Text Extender functions

RAHUSER　　Specifies the name of the instance owner used for the DB2 command `db2all`. Used only in a DB2 EEE environment.

If you are the instance owner, these environment variables are set automatically for you when the instance is created. If you are not the instance owner, include the `db2tx` profile from the instance owner, rather than trying to set these environment variables manually.

Tip

Whenever you change any environment variables, to make the changes effective, you must stop and restart DB2 and DB2 Text Extender.

Text configuration settings

Each database has text configuration settings consisting of:
* Text characteristics
* Index characteristics
* Processing characteristics

These are set when you enable the database for use by DB2 Text Extender. The ENABLE DATABASE command takes either the settings that you specify in the command, or it takes the initial settings described here. You can display and change these default settings; see "Displaying the text configuration settings" on page 87 and "Changing the text configuration" on page 15.

The text characteristics
"Chapter 4. Planning for your search needs" on page 23 describes the document formats, languages, and CCSIDs supported by DB2 Text Extender. Default values for these are required by various commands.

FORMAT　　　Initial setting: TDS

LANGUAGE　　Initial setting: The LANGUAGE that was set for the database

CCSID　　　　Initial setting: The CCSID that was set for the database

The index characteristics

DIRECTORY　Directory to be used to store the index.

Initial setting for UNIX systems:
DB2TX_INSTOWNERHOMEDIR/db2tx/indexes

Initial setting for OS/2, Windows NT, and Windows 2000:
DMBMMPATH\instance*instance-name* \db2tx\indexes where
DMBMMPATH is the path in which DB2 Text Extender is installed.
For Windows clients, DIRECTORY must be set to the name of a
directory on the server machine.

INDEXTYPE   Index type to be used. See "Types of search" on page 32 for a
description.

Initial setting: NGRAM

UPDATEFREQ

Frequency for periodic index update. See "Updating an index" on
page 36 for a description.

Initial setting: NONE

The processing characteristics

UPDATEINDEX

Setting to determine when the first index update occurs: either
immediately during the enabling step, or later according to the
update frequency settings (NOUPDATE), or as a result of an
explicit UPDATE INDEX command.

Initial setting: UPDATE

COMMITCOUNT

Setting to determine after how many insert or update statements
DB2 Text Extender issues a DB2 UDB commit statement. See
"Enabling a text column in a large table" on page 57.

Initial setting: 0

Changing the text configuration

When DB2 Text Extender is first installed, default values are set for the text
configuration. To display the current text configuration values, see "Displaying the
text configuration settings" on page 87.

To change the text configuration to be used as default values when indexes are
created, enter:

```
db2tx CHANGE TEXT CFG USING settings
```

Examples:

To change the default index type and the default index directory for future
indexes:

```
        DB2TX CHANGE TEXT CONFIGURATION USING
        INDEXTYPE   precise
        INDEXOPTION normalized
        DIRECTORY   /myfs/indexes
```

To change the default update frequency for indexes so that they are updated at
12:00 or 15:00, on Monday to Friday, if there is a minimum of 100 text documents
queued:

```
db2tx CHANGE TEXT CONFIGURATION USING
        UPDATEFREQ min(100) d(1,2,3,4,5) h(12,15) m(00)
```

To stop the periodic updating of an index:

```
db2tx CHANGE TEXT CONFIGURATION USING
        UPDATEFREQ none
```

To check the settings, use:

```
db2tx CHANGE TEXT CONFIGURATION
```

Setting up and maintaining a DB2 Text Extender server

The next step for a newly installed DB2 Text Extender is to set up the DB2 Text Extender server for use by DB2 Text Extender clients. This is normally done by a DB2 Text Extender server administrator and involves:

1. Creating the DB2 Text Extender instance
2. Starting a DB2 Text Extender server

Other tasks for maintaining a DB2 Text Extender server are:

1. Backing up and restoring DB2 Text Extender indexes and enabled databases
2. Tracing faults

See "Chapter 10. Administration commands for the server" on page 145 for the command syntax.

Creating the DB2 Text Extender instance

Before you can start to work with DB2 Text Extender, you must create the DB2 Text Extender instance, which offers an administration environment to maintain and store the indexes. Each instance that you create offers an isolated environment in which you can maintain indexes, storing them in separate directories.

To create an instance in a UNIX single-node environment, enter:

```
txicrt  instance-name db2-instance-name
```

where *instance-name* is the user ID of the instance owner, and *db2-instance-name* is the name of the DB2 UDB instance to be associated with it.

In Windows NT and Windows 2000 a default instance is created during installation. To create an additional instance, enter:

```
txicrt db2-instance-name
```

To create instances in a multinode environment, see "TXICRT" on page 146.

To drop an instance in UNIX systems, enter:

```
txidrop instance-name  db2-instance-name
```

To drop an instance in Windows NT and Windows 2000, enter:

```
txidrop db2-instance-name
```

To determine which instance you want to work with, set the appropriate environment variable:

```
set DB2INSTANCE = instance-name
```

then stop and restart DB2 and DB2 Text Extender.

An instance has to be created only once and remains active until it is dropped.

Starting and stopping a DB2 Text Extender server

Before you can index and search your documents, you have to start the DB2 Text Extender server. **To start the DB2 Text Extender server,** log on to the instance owner user ID (UNIX systems only), then enter:

```
txstart
```

To display the status of the DB2 Text Extender server, enter:

```
txstatus
```

To stop the DB2 Text Extender server, enter:

```
txstop
```

> **Tip for UNIX users**
>
> If you have problems starting a DB2 Text Extender server, try cleaning up shared resources. See "TXIPCLEAN (UNIX only)" on page 150.

Backing up and restoring indexes and enabled databases

You can back up and restore enabled databases and the text indexes that DB2 Text Extender has created.

To **back up**:
1. Find out which tables have been enabled by DB2 Text Extender. To do this, enter

   ```
   db2tx GET STATUS
   ```
2. Find out the names of the index directories used by the database. To do this, enter

   ```
   db2tx GET INDEX SETTINGS table-name
   ```
3. Ensure that no index update is running, and then stop the DB2 Text Extender server with the command:

   ```
   TXSTOP
   ```
4. Back up the index directories and their subdirectories `index` and `work`.
5. Back up the file `desmastr.dat` which is located in:

   ```
       instance_owner_home_directory/db2tx/txins000 (UNIX)
   or
       %DMBMMPATH%\instance\%DB2INSTANCE%\db2tx\txins000 (OS/2, Windows NT, and Windows 2000)
   ```

   Note that, for partitioned databases, there are several instance directories, named `txins000, txins001, ...`, so it would be more convenient to back up the whole `.../db2tx` directory rather than the files in the individual directories.
6. Restart the DB2 Text Extender server:

   ```
   TXSTART
   ```

To **restore**:
1. Stop the DB2 Text Extender server:

   ```
   TXSTOP
   ```
2. Save the existing `desmastr.dat` file.
3. Restore the backup copy of the `desmastr.dat` file.
4. Restore the backup copies of the index directories to the same path as before.
5. Restart the DB2 Text Extender server:

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. Use the trace facility only as directed by an IBM Support Center representative, or by your technical support representative.

System performance is affected when tracing is switched on, so use it only when error conditions are occurring.

To turn tracing on, enter:

```
txtrace on [options]
```

The syntax, and lists of the events and components are given in "TXTRACE" on page 164. Other options are also described there.

You can filter the trace by specifying a "mask" which causes the trace to accept or reject each trace record on the basis of its ID. The default is to trace everything.

A mask has four parts separated by periods, for example: 2.2-6.1,3.* where:

2   indicates DB2 UDB DB2 Text Extender.

2-6   includes only entries with an event ID between 2 and 6.

1,3   includes only those events reported by components 1 and 3.

*   includes all functions of the components.

You can exclude system errors below a certain severity, and you can specify, if the trace buffer becomes full, whether to keep the first or the last records.

To generate the trace information, issue the following command:

```
txtrace format > formatted-file
```

Note that the file is written to the current directory if no input is specified. After you have written the trace information to a file, turn tracing off using:

```
txtrace off
```

Chapter 3. Getting started

Use this chapter to become familiar with the basics of making text searchable. It assumes that you are working with a running DB2 Text Extender system, one that has been installed and configured, and where a DB2 Text Extender instance has been created and started.

> **Tip**
> This chapter describes only the basics of making text searchable. Before preparing your own text for searching, read "Preparation before making text searchable" on page 47.

There are two methods of becoming familiar with making text searchable:
- You can walk through a simple example, entering the commands manually, or
- You can run a sample indexing and search wizard.

A simple example of making text searchable

1. **In Windows systems, start the DB2 command line environment**

   At the operating system prompt, enter:
   ```
   =>db2cmd
   ```

2. **Start the DB2 Text Extender command line processor**

   In UNIX systems, at the operating system prompt, enter:
   ```
   =>db2tx
   ```

   In Windows systems, at the DB2 prompt, enter:
   ```
   db2=>db2tx
   ```

3. **Connect to a database**

   Choose a databasethat contains text that you want to make searchable. To connect to the database, enter:
   ```
   db2tx=>CONNECT TO mydatabase
   ```

4. **Enable the database for text search**

   To enable the connected database, enter:
   ```
   db2tx=>ENABLE DATABASE
   ```

5. **Enable a text table for text search (optional)**

   You need to enable a text table only if you want to create a single index for the whole table.

> **One index or several?**
>
> "Creating one or several text indexes for a table" on page 35 explains that you can make a table searchable by creating either one text index for the whole table, or by creating several indexes, one for each text column.
> - To create one index for the whole table, you would run ENABLE TEXT TABLE at this point to create an empty index, and then you run ENABLE TEXT COLUMN (see the next step) several times, once for each text column, to fill the single index.
> - Alternatively, to create a separate index for each text column, you skip the step to create an index for the whole table, that is, you skip ENABLE TEXT TABLE, and you run ENABLE TEXT COLUMN several times to create and fill indexes, one for each text column.
>
> For this example, you will not create an index for the whole table. Instead, continue with the following step to create an index for a text column.

6. **Enable a text column for text search**

   Enter the following command to enable DB2 Text Extender to search in text column `mycolumn` in table `db2tx.sample`, and to assign the name `myhandle` to the handle column that this command creates.

   ```
   db2tx=>ENABLE TEXT COLUMN db2tx.sample mycolumn HANDLE myhandle
   ```

   This command creates a text index. Default values are used for the type of documents being indexed and for the index characteristics.

7. **Check the status of the index you are creating**

   Enter:

   ```
   db2tx=>GET INDEX STATUS db2tx.sample HANDLE myhandle
   ```

8. **Leave the DB2 Text Extender command line processor**

   Enter:

   ```
   db2tx=>QUIT
   ```

9. **Start the DB2 command line processor**

   At the operating system prompt, enter:

   ```
   =>db2
   ```

10. **Search for text**

    Now your documents can be searched. Try this SELECT command which finds all occurrences of `searchterm` in the text that you have just indexed:

    ```
    db2=>SELECT COUNT (*)
                FROM sample
                WHERE DB2TX.CONTAINS (myhandle,'"searchterm"') = 1
    ```

The sample wizard for indexing and search

An indexing and search wizard is provided as an example of how to create indexes on selected tables or columns, and to search on those indexes. This sample wizard leads you through the steps required to be able to search on selected columns of a table using the DB2 Text Extender search functions.

To start the sample wizard, enter:

```
txwizard
```

You can use the Java source code that is provided with the sample wizard to either build your own text preparation and search application or to learn how the DB2 Text Extender functions can be used and called within a Java environment.

In AIX, the Java source code of the wizard is stored in directory
`/usr/lpp/db2tx_06_01/samples/wizard`

In Windows NT and Windows 2000, it is stored in directory
`%DMBMMPATH%\samples\wizard`

Getting started

Chapter 4. Planning for your search needs

Before you begin the steps described in "Chapter 5. Making text searchable" on page 47, you must find out:
- What format and code page your documents have, and what language they are in
- How to avoid code page problems
- What kind of search functionality you will need
- What disk space you will need
- What a text index is, and whether you want a common index for a DB2 table or a separate index for each table column.

This chapter describes why and how to collect this information.

There are several types of index to choose from: linguistic, precise, and Ngram. The choice of index type is significant. For example, if you choose *linguistic* as the index type, you can search for word variations and synonyms of the search term. The index type also affects indexing performance and the size of the index. You can also make use of the search capabilities of more than one index type by creating several indexes, each having a different index type, per text column.

Why text documents need to be indexed

A fast information retrieval system does not sequentially scan through text documents; this would take too long. Instead, it operates on a previously built text index. You can think of a text index as consisting of significant terms extracted from the text documents, each term stored together with information about the document that contains it.

A text index contains only relevant information; insignificant words, such as "and", "of", and "which", are not indexed. (No stop-word filtering is done for Ngram indexes.) DB2 Text Extender uses a list of these words, known as *stop words* to prevent them from being indexed. The retrieval system searches through the index for the terms requested to find which text documents contain those terms.

> **Tip**
> If you need to modify the list of stop words, do it only once, and at installation time.

A list of stop words per language is stored in a file that you can modify (see "Modifying the stop-word and abbreviation files" on page 45), but, because there is one file for the whole system, you should change it only once while you are setting up DB2 Text Extender for the first time. If you change the file later, existing indexes will not reflect the change.

As an example, let's say that some documents contain the name of a weekly magazine called "Now". If you remove this word from the stop words, it will be indexed and can be found by future searches. However, any indexes created before you removed the stop word will not contain the word "now", and a search for it will be unsuccessful.

Why text documents need to be indexed

If you do decide to change the stop words, and you want this change to be reflected throughout, you must recreate all your indexes.

Indexing is a two-step process. The first step is to record in a *log table* the text documents that need to be indexed. This occurs automatically through DB2 *triggers* whenever you insert, update, or delete a text document in a column.

The second step is to index the text documents listed in the log table. This may be done periodically. The terms of those documents that were inserted or changed in the column are added to the index. The terms of those documents that were deleted from the column are removed from the index.
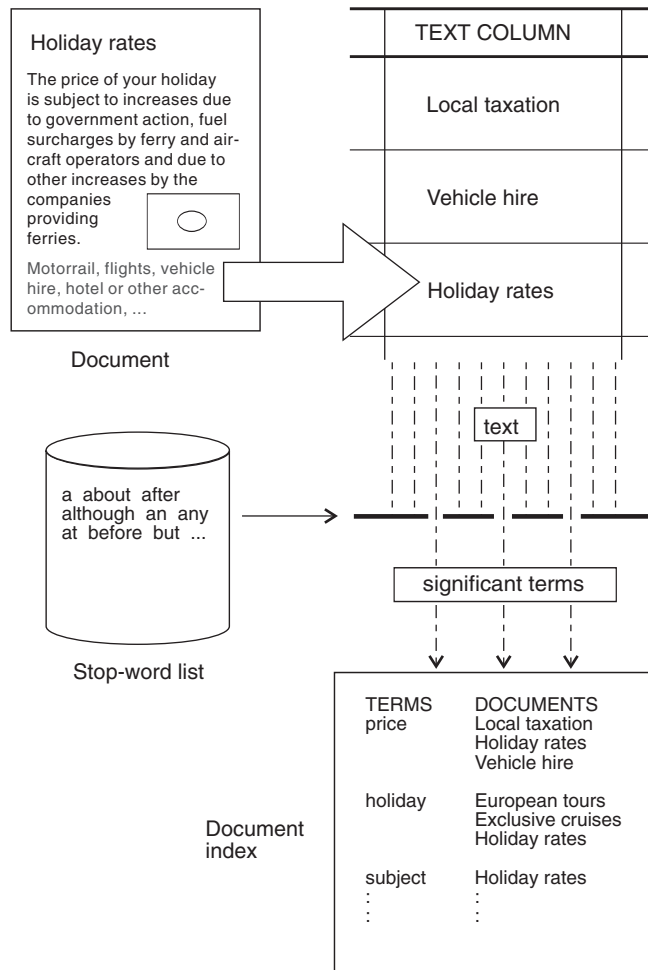
Figure 4. Indexing only significant terms

Which document formats are supported

DB2 Text Extender needs to know the format (or type) of text documents, such as HTML or ASCII, that you intend to search. This information is needed when indexing text documents.

The text document formats supported are:

HTML   Hypertext Markup Language

XML   Extended Markup Language.

ASCII_SECTIONS	Structured ASCII containing sections
TDS	Flat ASCII

The following document formats are supported for backward compatibility reasons:

AMI	AmiPro Architecture Version 4
FFT	IBM Final Form Text: Document Content Architecture
MSWORD	Microsoft Word, Versions 5.0 and 5.5
RFT	IBM Revisable Form Text: Document Content Architecture
RTF	Microsoft Rich Text Format (RTF), Version 1
WP5	WordPerfect (OS/2 and Windows), Versions 5.0, 5.1, and 5.2

HTML documents, special considerations

The treatment of umlauts and diacritical characters in HTML documents depends on the code page of the document:

* For code pages 37, 273, 277, 278, 280, 284, 297, 437, 500, 819, 850, 858, 860, 863, 865, 871, 923, 924, and 1252 the following apply:
  - Entity notation is used for umlauts and special characters, for example, ä for ä.
  - Only those characters that have a code point on code page 819 (ASCII) or 500 (EBCDIC) are valid.
  - During indexing, documents containing language-specific characters, such as ä, lead to errors in word recognition if the document's code page is neither 819 nor 500.
* For all other single-byte character set code pages, the following apply:
  - Entities are not resolved.
  - Special characters must be written in language-specific code points.

XML documents, special considerations

XML documents must contain an encoding string and this is used during parsing. The following encodings are supported:

```
UTF8, utf8, utf-8, ibm-1208, utf_8,
UTF16_BigEndian, utf-16be, utf16
UTF16_LittleEndian, utf-16le
LATIN_1, latin1, latin-1, ascii, ibm-819, iso-8859-1,  Latin-1
ibm-912, iso-8859-2
ibm-913, iso-8859-3
ibm-914, iso-8859-4
ibm-915, iso-8859-5
ibm-1089, iso-8859-6
ibm-813, iso-8859-7
ibm-916, iso-8859-8i
ibm-920, iso-8859-9
ibm-0037, ebcdic-cp-us, ebcdic-cp-ca, ebcdic-cp-nl, ebcdic-cp-dk, ebcdic-cp-no,
          ebcdic-cp-fi, ebcdic-cp-se, ebcdic-cp-it, ebcdic-cp-es, ebcdic-cp-gb
ibm-297, ebcdic-cp-fr, ebcdic-cp-ar1, ebcdic-cp-he, ebcdic-cp-ch, ebcdic-cp-roece,
         ebcdic-cp-yu, ebcdic-cp-is, ebcdic-cp-ar2
ibm-954, euc-jp eucjis
```

```
ibm-943, shift_jis, sjis, shiftjis, shift-jis
ibm-950 , big-5, big5
ibm-949, iso2022kr, euc-kr
ibm-878, koi8-r
```

The default code page is ignored for XML documents.

If the index has section support, only the leaf-level section identifier is used when indexing, there is no support for resolving text in nested sections.

Using unsupported document formats

For nonsupported document formats, specify a numeric ID. Valid values are 1 to 100. This value is passed as the source format to the user exit that converts the original format to TDS.

If, during indexing, there is a document that is not one of the supported types, DB2 Text Extender provides an exit that writes the document to a disk and calls a program that you provide to extract the text into one of the supported formats. The user exit must be registered in both the server configuration file (DESSRV.INI) and the client configuration file (DESCL.INI).

Update the USEREXIT option in the [DOCUMENTFORMAT] section with the name of the user exit.

To enable the user exit, edit the following ASCII files:

```
Windows NT and Windows 2000:
%DMBMMPATH%\instance\%DB2INSTANCE%\db2tx\descl.ini
%DMBMMPATH%\instance\%DB2INSTANCE%\db2tx\txinsnnn\dessrv.ini

UNIX:
$DB2TX_INSTOWNERHOMEDIR/db2tx/descl.ini
$DB2TX_INSTOWNERHOMEDIR/db2tx/txinsnnn/dessrv.ini
```

by adding the following statements:

```
[DOCUMENTFORMAT]
USEREXIT=name_of_executable
```

where <name_of_executable> is the name of the user exit. You can specify a fully qualified file name, or, if the user exit is stored in a directory that is in the PATH statement, you can specify only the file name.

To call the user exit use the following syntax:

```
<name_of_user_exit>  -sourcefile   <sourcefilename>
                     -targetfile   <targetfilename>
                     -sourceccsid  <sourceccsid>
                     -targetccsid  <targetccsid>
                     -sourceformat <sourceformat>
                     -targetformat <targetformat>
```

sourcefilename
> The file to be converted by the user exit program. The file name is fully qualified and is located in the working directory specified either in the client profile or the server instance.

targetfilename
> The file containing the output of the user exit. This file is then used for processing by DB2 Text Extender. The file name is fully-qualified and points to the working directory specified either in the client profile or the

server instance. The entries in the client profile are used for the API call EhwGetMatches and those in the server instance for the API call EhwUpdateIndex.

sourceccsid
: The code page of the source file. This is the default code page.

targetccsid
: The code page expected by DB2 Text Extender. The code page is 850.

sourceformat
: The format of the source file. This is the default format.

targetformat
: The format of the file expected by DB2 Text Extender. Currently, only the flat-file format (TDS) or, for section-enabled indexes, ASCIISECTION are supported.

The user exit must be able to return the following values:

0
: Format conversion was successful.

>0
: Format conversion was not successful. During indexing, the error messages are written to the document error table. Use the `desmsgix` command to display the error messages.

Languages

DB2 Text Extender also needs to know in which language a document is written so that the correct dictionary can be used for the linguistic processing that occurs. Table 5 on page 43 shows a list of the language parameters that you can specify when you enable a text column or external documents.

CCSIDs

> ┌─ **Tip** ──
> Before specifying a CCSID when enabling a text column, read "Avoiding code page problems when storing and enabling text" on page 30
> └──

Documents can be indexed if they are in one of the CCSIDs in the table below.

Note: CCSIDs 806, 858, 861, 865, 878, 923, 924, and 4946 are not supported by DB2 UDB . To index documents having these CCSIDs, store the documents in a column with a binary data type (BLOB or FOR BIT DATA).

EBCDIC

37	US, Canadian English
273	Austrian, German
277	Danish, Norwegian
278	Finnish, Swedish
280	Italian
284	Spanish, Latin American

CCSIDs

285	UK English
297	French
420	Arabic
424	Hebrew
500	International Latin-1
838	Thai
870	Latin-2
871	Icelandic
875	Greek
924	Latin-9
1025	Russian
1026	Turkish
1112	Latvian
1122	Estonian
1123	Ukranian
1130	Vietnamese

ASCII

437 OS/2	US English
806	Hindi
813 AIX, HP, SUN	Greek
819 AIX, HP, SUN	Latin-1
850 AIX, OS/2	Latin-1
852 OS/2	Latin-2
855 OS/2	Bulgarian
857 OS/2	Turkish
858 AIX, OS/2	Latin-1 with Euro symbol
860 OS/2	Portuguese
861 See note	Icelandic
862 OS/2	Hebrew
863 OS/2	Canadian
864 OS/2	Arabic
865 See note	Danish, Norwegian
866 OS/2	Russian
869 OS/2	Greek
874 AIX, HP, WIN, OS/2	Thai
912 AIX, HP	Latin-2
915 AIX, OS/2, HP	Russian

916 AIX	Hebrew
920 AIX, HP	Turkish
921 AIX, OS/2, WIN	Latvian, Estonian
922 AIX, OS/2, WIN	Estonian
923	Latin-9 (ISO 8859–15)
1046 AIX	Arabic
1089 AIX, HP	Arabic
1124 AIX	Ukranian
1125 OS/2	Ukranian
1129 AIX	Vietnamese
1131	Vietnamese
1250 WIN	Latin-2, Belorussian
1251 WIN	Russian
1252 WIN	Latin-1
1253 WIN	Czech
1254 WIN	Turkish
1255 WIN	Hebrew
1256 WIN	Arabic
1257 WIN	Greek
1258 WIN	Vietnamese

DBCS

878 (identical to RFC 1489)	Russian
932 AIX, OS/2	Japanese, combined SBCS/DBCS
933 ERCDIC	Korean
935 ERCDIC	Chinese (simplified)
937 ERCDIC	Chinese (traditional)
942 OS/2	Japanese, combined SBCS/DBCS
943 OS/2, WIN, AIX	Japanese, combined SBCS/DBCS
5039 HP	Japanese, combined SBCS/DBCS
954 AIX, HP, SUN	Japanese
949 OS/2	Korean
970 AIX, HP, SUN	Korean
1363 WIN	Korean
948 OS/2	Chinese (traditional), combined SBCS/DBCS
950 AIX, HP, OS/2, SUN, WIN	Chinese (traditional), combined SBCS/DBCS
964 AIX, HP, SUN	Chinese (traditional), combined SBCS/DBCS

1381 OS/2, WIN	Chinese (simplified), combined SBCS/DBCS
1383 AIX, HP, SUN	Chinese (simplified), combined SBCS/DBCS
1386 AIX, OS/2, WIN	Chinese (simplified), combined SBCS/DBCS
1388 ERCDIC	Chinese (simplified)
4946 See note	Latin-1 (CP850)
5026 ERCDIC	Japanese Katakana
5035 ERCDIC	Japanese Latin

UNICODE

1208	UTF8
13488	UCS2

Avoiding code page problems when storing and enabling text

The following areas have code page settings:
- The active application environment
- Each document
- Each DB2 database
- Each DB2 Text Extender index

When you store documents in a DB2 database column having a character data type, such as VARCHAR and CLOB, DB2 assumes that each document has the same code page as the active application environment and converts the document from that code page to the code page of the database. The code page of the database is either already the same as that of the active application environment (no conversion takes place), or it is the code page that you specified when you created the database and which was different to the application code page. (conversion does take place).

When you store data in a DB2 database in a column having a binary data type, such as BLOB or FOR BIT DATA, DB2 does not convert the data, and the documents retain their original CCSIDs.
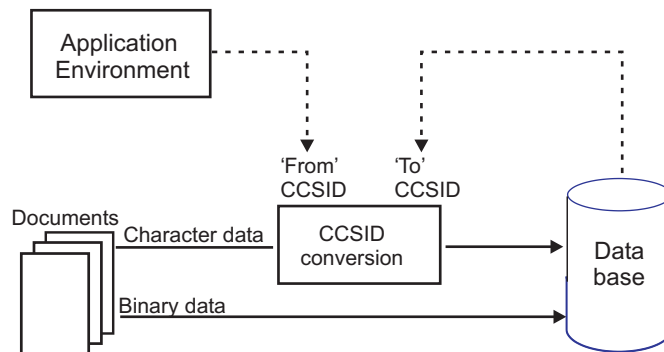
Figure 5. How DB2 sets the CCSID of a database document

When you enable a text column for use by DB2 Text Extender, that is, when you use the ENABLE TEXT COLUMN command to create an index for searching, the code page of the index is set either to the code page of the database (default), or to the current default which can be set using CHANGE TEXT CONFIGURATION

command, or to the code page you specified in the ENABLE TEXT COLUMN command.
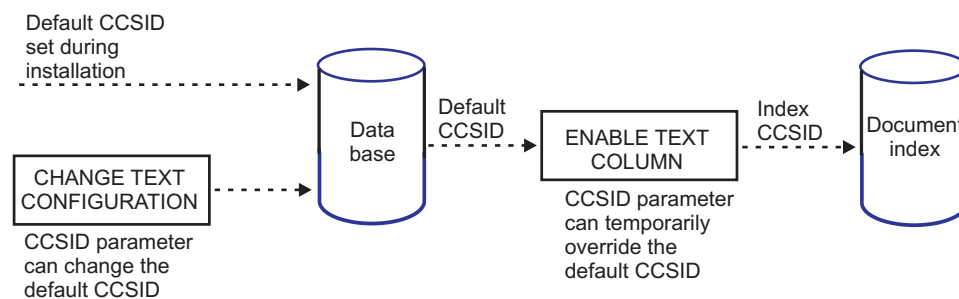
Figure 6. How DB2 Text Extender sets the CCSID of a text index

During a search, the CCSID of the database is used to interpret the CCSID of the search string.
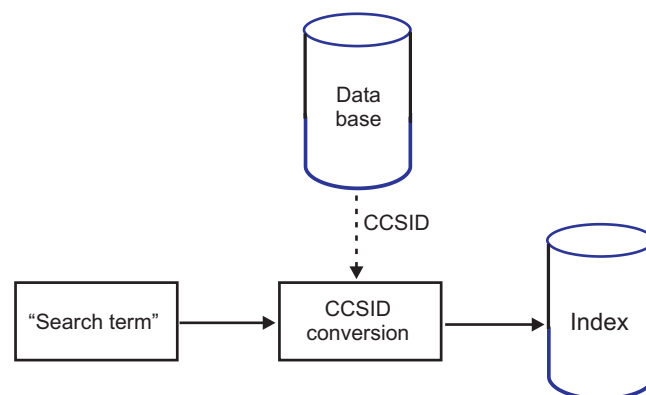
Figure 7. Search uses the database CCSID

Here's how to check the code page settings:
* To check the database code page, use the DB2 command:

  ```
  db2 get db cfg
  ```
* To check the default index code page, use the DB2 Text Extender command:

  ```
  db2tx get text cfg
  ```

To avoid code page problems it is important to coordinate these code-page settings correctly:
* **Example 1:** Environment 850, Document 850, Database 850, Index 850

  Correct. DB2 correctly assumes that the document has the same code page as the environment, and, because the database has the same code page as the environment, DB2 makes no conversion, storing the document with code page 850 in the database.. When you then enable the text column, and take the default code page setting (850, the code page of the database), the document is indexed correctly into the 850-code page index.
* **Example 2:** Environment 1252, Document 1252, Database 850, Index 850

  Correct. DB2 correctly assumes that the document has the same code page as the environment, and makes a 1252-to-850 code page conversion when storing the document in the database. When you then enable the text column, and take the default code page setting 850 (the code page of the database), the document is indexed correctly into the 850 code page index.

- **Example 3:** Environment 1252, Document 850, Database 850, (Index ANY)

  Error. DB2 incorrectly assumes that the document (code page 850) has the same code page as the environment (code page 1252) and makes an incorrect 1252-to-850 code page conversion when storing the document in the database.

- **Example 4:** Environment 1252, Document 850, Database 1252, Index 850

  Correct. DB2 incorrectly assumes that the document has the same code page as the environment, but, because the environment code page is the same as the database codepage, DB2 makes no conversion, storing the document in the database in code page 850. When you then enable the text column, however, you must specify a document code page of 850 so that the document is correctly indexed into the 850-code page index.

- **Example 5:** Environment 1252, Document 1252, Database 850, Index 1252

  Potential error. DB2 correctly assumes that the document has the same 1252 code page as the environment and converts the document to code page 850 when storing it in the database. The potential error occurs if you then specify the document's original code page of 1252 when you enable the text column for the 850 code page index. The correct action would be to take the default code page setting 850 (the code page of the database).

Types of search

You can assign one of these index types and various options to a column containing text to be searched: *linguistic, precise,* and *Ngram.* You must decide which index type to create before you prepare any such columns for use by DB2 Text Extender. For a more detailed description of how each type of index affects linguistic processing, read "Chapter 15. Linguistic processing for linguistic and precise indexes" on page 221.

Summary

- For **searching for linguistic word variations**, use a **linguistic index.**

  Finds word variations based on normalization and stemming and on the use of a dictionary; uses the least disk space.

- For **making exact searches**, use a **precise index.**

  Finds the term exactly as entered; indexing and search are faster; uses more disk space. If NORMALIZED, searches are case-insensitive.

- For **searching for character variations**, use an **Ngram index.**

  Finds words even if spelled incorrectly; if CASE_ENABLED to allow case-sensitive search, the index uses more space, and searches can take longer.

- For **searching in DBCS documents**, use an **Ngram index.**

  The only choice for DBCS documents, but can also be used for SBCS documents of type TDS.

DB2 Text Extender offers a wide variety of search options, though not all are available for all index types. See Table 8 on page 194 and Table 9 on page 194 before making your decision about which index type to use.

> **Note**
>
> Not all of the index types are available for languages which are supported by the Text Extender. See the table in "Dictionaries, stop-word lists, abbreviation lists, and language parameters" on page 43 for further information.

Linguistic search

For a linguistic index, linguistic processing is applied while analyzing each document's text for indexing. This means that words are reduced to their base form before being stored in an index; the term "mice", for example, is stored in the index as `mouse`.

For a query against a linguistic index, the same linguistic processing is applied to the search terms before searching in the text index. So, if you search for "mice", it is reduced to its base form `mouse` before the search begins. Table 18 on page 221 summarizes how terms are extracted for indexing when you use a linguistic index.

The advantage of this type of index is that any variation of a search term matches any other variation occurring in one of the indexed text documents. The search term `mouse` matches the document terms "mouse", "mice", "MICE" (capital letters), and so on. Similarly, the search term `Mice` matches the same document terms.

This index type requires the least amount of disk space. However, indexing and searching can take longer than for a precise index.

The types of linguistic processing available depend on the document's language. Here is a list of the types:

- Word and sentence separation.
- Sentence-begin processing.
- Dehyphenation.
- Normalizing terms to a standard form in which there are no capital letters, and in which accented letters like "ü" are changed to a form without accents. For example, the German word "Tür" (door) is indexed as `tuer`.
- Reducing terms to their base form. For example, "bought" is indexed as `buy`, "mice" as `mouse`.

> **Tip**
>
> Word fragments (words masked by wildcard characters) cannot be reduced to a base form. So, if you search for `swu%`, you will not find the word "swum", because it is reduced to its base form in the index. To find it, you must search for `swi%`.
>
> Variations of words not having the correct spelling cannot be reduced to a base form. Take, for example, the German word `röstete` which is indexed correctly in its base form, the verb `rösten`. A search term `röstete` or `rösteten` is normalized correctly to the base form `rösten`, and the term is found in the index. A search term `rostete` cannot be normalized `rösten`, and the term is not found in the index.

- Word decomposition, where compound words like the German "Wetterbericht" (weather report) are indexed not only as `wetterbericht`, but also as `wetter` and `bericht`.
- Stop-word filtering in which irrelevant terms are not indexed. "A report about all animals" is indexed as `report` and `animal`.
- Part-of-speech filtering, which is similar to stop-word filtering; only nouns, verbs, and adjectives are indexed. "I drive my car quickly" is indexed as `drive` and `car`. The words "I" and "my" are removed as stop words, but additionally the adverb "quickly" is removed by part-of-speech filtering.

Precise search

In a precise index, the terms in the text documents are indexed exactly as they occur in the document. For example, the search term `mouse` can find "mouse" but not "mice" and not "Mouse"; the search in a precise index is case-sensitive.

In a query, the same processing is applied to the query terms, which are then compared with the terms found in the index. This means that the terms found are exactly the same as the search term. You can use masking characters to broaden the search; for example, the search term `experiment%` can find "experimental", "experimented", and so on.

Table 19 on page 222 gives some examples of how terms are extracted from document text for indexing when you use a precise index.

The advantage of this type of index is that the search is more precise, and indexing and retrieval is faster. Because each different form and spelling of every term is indexed, more disk space is needed than for a linguistic index.

The linguistic processes used to index text documents for a precise index are:
  Word and sentence separation
  Stop-word filtering.

Make a fuzzy search or search in DBCS documents

An Ngram index analyzes text by parsing sets of characters. This analysis is not based on a dictionary.

If your text contains DBCS characters, you must use an Ngram index. No other index type supports DBCS characters.

This index type supports "fuzzy" search, meaning that you can find character strings that are similar to the specified search term. For example, a search for `Extender` finds the mistyped word `Extendrrs`. You can also specify a required degree of similarity.

Note: Even if you use fuzzy search, the first three characters must match.

To make a case-sensitive search in an Ngram index, it is not enough to specify the PRECISE FORM OF keyword in the query. This is because an Ngram index normally does not distinguish between the case of the characters indexed. You can make an Ngram index case-sensitive, however, by specifying the CASE_ENABLED option when the index is created. Then, in your query, specify the PRECISE FORM OF keyword.

When the CASE_ENABLED option is used, the index needs more space, and searches can take longer.

See "CCSIDs" on page 27 for a list of the CCSIDs supported by Ngram indexes. An Ngram index supports a list of native CCSIDs. For all other CCSIDs, the data is mapped from this CCSID to UTF8.

Although the Ngram index type was designed to be used for indexing DBCS documents, it can also be used for SBCS documents.

Note also that not all of the search syntax options are supported. See the summary of rules and restrictions in "Chapter 12. Syntax of search arguments" on page 187.

Changing the index type

If you decide that the index type you are using is not suitable, first delete the index by disabling the text column or text table, and then recreate the index by re-enabling the text column or text table.

Creating one or several text indexes for a table

"Chapter 5. Making text searchable" on page 47 describes how to prepare tables so that you can search in them for text. Before you do this preparation, however, you must decide to create either one text index that is common to all indexed text columns in a table, or several text indexes, one for each indexed text column. A table having a separate index for each text column is known as a *multi-index table*.

> **Tip**
>
> If you intend to index external files (see "Enabling external text files" on page 59), the associated table must be a multi-index table.

Using multiple indexes has these benefits:
- Creating a different index type for each text column

  This gives you flexibility in the characteristics that are associated with a text column, such as when its index is periodically updated, and in which directory the index is stored. See "ENABLE TEXT COLUMN" on page 119 for a description of these characteristics.
- Indexing columns at different times

  Indexing can be a time- and resource-consuming activity. By having a multi-index table, you can spread this activity over a period of time by indexing the columns at different times.

If you do not need the flexibility offered by a multi-index table, a common index makes DB2 Text Extender easier to maintain; when you enable a text table, you set the indexing parameters which are used as default values for all its text columns. Also, if you need to disable the columns you can do it using one command by disabling the text table.

Calculating the size of an index

The disk space you need for an index depends on the size and type of data to be indexed, and on the index type. Text documents written with word processors need less space because much of their content is taken up with control characters. As a guideline, for a linguistic index reserve disk space for about 0.7 times the size of the documents being indexed, then multiply this by 2 to reserve temporary space for reorganizing the index. For an Ngram index you'll need almost twice as much disk space.

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.

Updating an index

When a text document is added to a database, or when an existing document in a database is changed, the document must be indexed to keep the content of the index synchronized with the content of the database. When a text document is deleted from a database, its terms must be removed from the index.

Information about which documents are new, changed, and deleted is automatically stored by triggers in a log table. The documents listed in the log table are indexed the next time an index update takes place.

The UPDATE INDEX command lets you update an index immediately on request. Typically, however, you automatically update an index at intervals specified in the environment variable DB2TXUPDATEFREQ. The environment variable determines the default settings. The default settings can be overridden when creating an index using the ENABLE TEXT COLUMN or ENABLE TEXT TABLE commands. The update frequency can be changed for an existing index using the CHANGE INDEX SETTINGS command.

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

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

Syntax

```
►►──MIN──(──mindocs──)──D──(─────*────────────)──H──────────────────►
                                 │     ┌─,─┐   │
                                 └──▼──0...6────┘
```

```
►─(─┬──────*──────┬─)─M─(─┬─0...59─┬─)────────────────────►◄
    │      ┌─,─┐  │        ▼   ┌─,─┐
    │      ▼      │        └───┘
    └──────0...23─┘
```
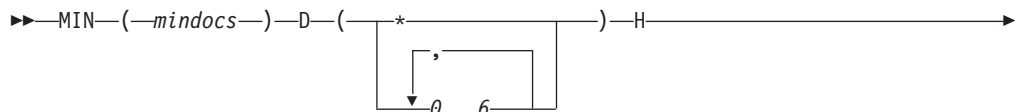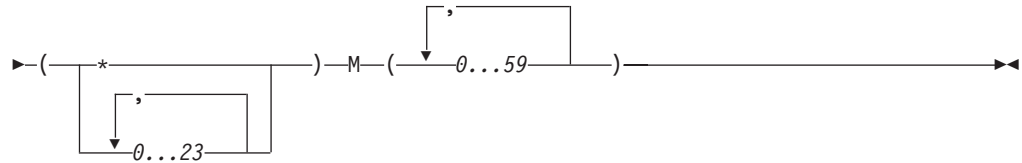
MIN mindocs
> The minimum number of text documents that must be queued before the index can be updated.

D The day(s) of the week when the index is updated:

> ***** Every day
>
> **0** Sunday
>
> **1** Monday
>
> **2** Tuesday
>
> **3** Wednesday
>
> **4** Thursday
>
> **5** Friday
>
> **6** Saturday

H The hour(s) of the specified day(s) when the index is updated:

> ***** Every hour
>
> **0...23** At the specified hour

M The minute(s) of the specified hour(s) when the index is updated:

> **0...59** At the specified minute

Example: `min(100) d(1,2,3,4,5) h(12,15) m(0)`

If, at 12:00 or 15:00, on Monday to Friday, there are at least 100 text documents queued, the index is updated.

You can combine several frequency specifications:

Example: `min(1) d(*) h(22) m(0) ; min(100) d(1,2,3,4,5) h(12,15) m(0)`

Index updating is scheduled on Monday to Friday at 12:00 and 15:00 as before, but, in addition, each day at 22:00 the index is updated even if there is only one text document in the log table.

Maintaining indexes in a multiple-node environment

If the text column that you are enabling belongs to a table that is part of a multiple-node nodegroup, the index directory that you specify must be available on all physical nodes. If you use the default directory specified in the text configuration, make sure that the path is available on all nodes of the nodegroup. If this is not convenient, you can specify a specific path for each node in the ENABLE TEXT COLUMN command.

If you change the node configuration of a nodegroup that contains a table that is enabled for DB2 Text Extender, you must reindex the table. You can check if the node configuration has been changed by using the command `TXNCHECK`.

Maintaining indexes in a multiple-node environment

> **Tip**
>
> If you are using DB2 UDB Extended Enterprise Edition, and have tables partitioned among several nodes, there may be a difference between the system time of the nodes (when one of the nodes is on summer time, for example). To prevent problems, ensure that the difference between the system time settings of the nodes is within the value of DB2's MAX_TIME_DIFF setting. In Windows NT and Windows 2000, use the NET TIME command to synchronize the system time settings.

Working with structured documents (section support)

Section support allows you to index and search specific sections in a structured document, for example, in the title, author, or description. The documents can be in XML or HTML format or flat-file documents with HTML-like tags. You define the markup tags and their corresponding section names in a *document model*. The document model defines which sections in the documents are indexed and therefore available for searching. The section names are descriptive names used in queries against that section.

To make section support available, you must specify INDEXPROPERTY SECTIONS_ENABLED when you enable the text column that contains the documents.

A *document model file* lists all the defined document models for the server instance. When a server instance is created, a sample document models file, DESMODEL.INI, is created automatically in the server instance subdirectory. The file is in ASCII code page.

The document model information is copied to the index directory. If you change the document models file for the server instance after you create the index, it does not affect the section support for the created index.

A search on an index with section support, for example, to search for McDaniel in the section Author, might look as follows. The section, in this case Author, is always prefixed by the model name.

```
db2 "select count (*)
    from db2tx.htmltable
    where db2tx.contains(myhandle,'MODEL myhtmlmodel SECTION (author) "McDaniel"') = 1
```

Attribute sections

Sections can be of different types. Plain-text sections have no type. Sections with a declared type are known as *attribute sections*. The supported attribute types are:

- DATE
- TIME
- FLOAT
- INTEGER

You can use value ranges to search documents containing attribute sections. The section must have the appropriate section type. In the following example, the section ABC must have the type TIME for the query to be supported:

```
retrieve all documents containing values between
20:00:00 and 22:00:00 in the section ABC
```

A section can have more than one attribute value. Separate values with a blank or a new line. You cannot nest attribute sections.

Supported date formats

You can include dates, for example, the creation date or publishing date in the list of supported sections in a document model. The following date formats are supported:

Table 3. Supported date formats

Format name	Format	Example
ISO	YYYY-MM-DD	1986-12-25 100-12-25 10123-2-17
	DD-MM-YYYY	01-10-1999 5-8-1990
IBM USA Standard (USA)	MM/DD/YYYY	08/18/1999 5/13/1977
	YYYY/MM/DD	1999/08/18
IBM European Standard (EUR)	DD.MM.YYYY	20.10.1999 1.5.1980

Supported time formats

The following time formats are supported:

Table 4. Supported time formats

Format name	Format	Example
ISO	hh:mm:ss hh:mm hh.mm.ss hh.mm hh hhmm hhmmss	19:00:23 19:00 19.00.23 19.00 19 1900 190013

You can omit leading zeros for hours. Values outside the following ranges are not valid:

Hours	Minutes	Seconds
0–23	0–59	0–59

Supported float formats

The following formats are supported:

- A sequence of digits in the range 1.175494351e–38F through 3.402823466e+38F
- Scientifice notation using e or E.

Use blanks or new lines to separate float values.

Structured documents (section support)

The following are examples of correct and incorrect formats for float values:

Correct format	Incorrect format
1000 1 1.0e3 1.0E3 1E3 .1E4 0.1E4	1 000 1,000 e3

Supported integer formats

Integers in the range –2147483648 through 2147483647 are supported. Use blanks to separate integers. This means that the string "42 234" in a section of type INTEGER resolves to the integer values "42" and "234" and not "42234".

The following are examples of correct and incorrect formats for integer values:

Correct format	Incorrect format
1000	1.000 1,000 1 000
1000000	100.000 100,00 100 000

> **Note**
> Language-specific separators and monetary formats are not supported.

Flat files and HTML documents

For flat files, the sections are marked up using HTML-like tags, such as <title> and <subject>. A document with marked-up sections might look as follows:

```
<title> IBM Dictionary of Computing
<author> McDaniel, George
<subject> Computers, Reference, ....
```

A document models file for flat files or HTML documents might look as follows. The model names and section names can contain only A-Z, a-z, and 0-9. Model names are always case sensitive. Section names can be either case sensitive or case insensitive; you specify the setting when you create the model.

```
;list of document models
;model always starts with 'modelname' and the name of the model
[MODELS]
modelname=sample
modelname=sample2
modelname=sample3

; a 'sample' document model definition
; left  - logical section name identifier used for searching in a document
; right - section name tag
[sample]
Title = title
Author = author
Subject = subject
Abstract = abstract
```

```
Content = content
Date=publishingdate

[sample2]
Title = title
Author = author
Subject = subject
[sample3]
Title = title
Author = author
Abstract = abstract
Docnum = docnum
```

If a document contains a marked-up section that is not defined in the document model, the contents of the section are included in the previously defined section for indexing and searching. For example, a document contains the following marked-up sections:

```
<title> IBM Dictionary of Computing
<subject> Computers, Reference, ....
<author> McDaniel, George
<abstract> Contains up-to-the-minute coverage of information processing systems,
communication products and facilities, personal computers, and office systems, as
well as the full range of IBM hardware and software products.
```

The document model, book, is defined as:

```
[MODELS]
modelname=book
[book]
Title = title
Author = author
Abstract = abstract
```

The <subject> section is not included in the book document model. When the document is indexed, the contents of the subject section are indexed with the contents of the title section. They are also available for searching within the title section.

If you specify a list of models when you create the index, the default model is the first in the list. You can change the default model using the desmodix command.

XML documents

For section-enabled indexes, XML documents must be correctly structured and contain a root element. The name of the root element must be the same as one of the defined model names and the case must match. The model description in the document models file must be a subset of the document model defined in the DTD (document type definition) file for the document.

The model description must begin with the root element. For each XML element you want to use as a section, you must include its complete hierarchy in the model description. If a section is of type date, this section must be a leaf in the document model tree. Nesting of attribute sections is not supported.

A model description for XML documents might look as follows:

```
; list of document models
[MODELS]
modelname = LETTER
; sample for XML model definition
; left-hand side  = logical section name identifier used for searching
; right-hand side = section name tags specifying the tags for each
;                   element of the path through the tree down to
```

Structured documents (section support)

```
;                      the specified node. Tag delimiter is /.
[LETTER]
LETTER = LETTER
LETTER/date = LETTER/DATE
LETTER/address = LETTER/ADDRESS
LETTER/address/City = LETTER/ADDRESS/CITY
LETTER/Content = LETTER/CONTENT
LETTER/Content/Greetings = LETTER/CONTENT/GREETINGS
```

A sample for the attribute sections might look as follows:

```
[MODELS]
modelname = SAMPLE4
[SAMPLE4]
Title = title
Date = publishingdate,date
Docnum = docnum, integer
```

> **Note**
>
> For the logical section names, you can either use logical names without tag
> delimiters, or use logical names with the complete path. For example:
>
> ```
> LETTER = LETTER
> date = LETTER/DATE
> address = LETTER/ADDRESS
> ```

An XML document might look as follows. It also shows how the sections that are
not defined in the model are indexed.

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "letter.dtd">

<LETTER>
 <HEADER>This tag has been skipped in the definition, to this text will
          be added to the section named LETTER
 </HEADER>
 <DATE>
          01.01.2000  03.02.2000
 </DATE>
 <ADDRESS>
          Text will be added to the section named LETTER/address.
   <CITY>
            Text will be added to section named LETTER/address/City.
   </CITY>
 </ADDRESS>
 <CONTENT>
          Text will be added to the section named LETTER/Content.

   <NOSECTION>Text will be added to the section named LETTER/Content
            because NOSECTION is not defined.
   </NOSECTION>
   <GREETINGS>
            Text will be added to section named LETTER/Content/Greetings.
   </GREETINGS>
  </CONTENT>
</LETTER>
```

Dictionaries, stop-word lists, abbreviation lists, and language parameters

Table 5 shows the supported languages and the names of the files that are provided as dictionaries, stop-word lists, and lists of abbreviations. The dictionary files are in binary format and cannot be changed. The stop-word files and abbreviation files (if they exist) are in flat-file format and can be changed. If you change any of these files, ensure that you use the code page for the language.

This table also shows which language parameter you must specify when you enable a text column or external documents. This tells DB2 Text Extender in which language the documents are written so that the correct dictionary can be used for the linguistic processing that occurs.

Table 5. List of supported languages and index types

Language	Keyword	Filename (1)	LINGUISTIC	PRECISE / PRECISE NORMALIZED	NGRAM / NGRAM CASE ENABLED
Albanian	ALBANIAN		no	yes	yes
Arabic	ARABIC	arabic	yes	yes	yes
Belorussian	BELORUSSIAN		no	yes	yes
Brazilian	BRAZILIAN	brazil	yes	yes	yes
Bulgarian	BULGARIAN		no	yes	yes
Canadian French	CAN_FRENCH	canadian	yes	yes	yes
Catalan	CATALAN	catala	yes	yes	yes
Chinese (simplified)	S_CHINESE		no	no	yes
Chinese (traditional)	T_CHINESE		no	no	yes
Croatian	CROATIAN		no	yes	yes
Czech	CZECH		no	yes	yes
Danish	DANSK	dansk	yes	yes	yes
Dutch Dutch/ Belgium	DUTCH	nederlnd	yes	yes	yes
English (UK)	UK_ENGLISH	uk	yes	yes	yes
English (US)	US_ENGLISH	us	yes	yes	yes
Estonian	ESTONIAN		no	yes	yes
Finnish	FINNISH	suomi	yes	yes	yes
Greek	GREEK		no	yes	yes
French French/ Switzerland French/ Belgium	FRENCH	francais	yes	yes	yes
German	GERMAN	deutsch	yes (2)	yes	yes

Dictionaries, stop-word lists, and abbreviation lists

Table 5. List of supported languages and index types (continued)

Language	Keyword	Filename (1)	LINGUISTIC	PRECISE / PRECISE NORMALIZED	NGRAM / NGRAM CASE ENABLED
Hebrew	HEBREW	hebrew	yes	yes	yes
Hindi	HINDI		no	no	yes
Hungarian	HUNGARIAN		no	yes	yes
Icelandic	ICELANDIC	islensk	yes	yes	yes
Italian	ITALIAN	italiano	yes	yes	yes
Japanese	JAPANESE		no	no	yes
Korean	KOREAN		no	no	yes
Latvian	LATVIAN		no	yes	yes
Lithuanian	LITHUANIAN		no	yes	yes
Macedonian	MACEDONIAN		no	yes	yes
Norwegian Bokmal	BM_ NORWEGIAN	norbook	yes	yes	yes
Norwegian Nynorsk	NN_ NORWEGIAN	nornyn	yes	yes	yes
Norwegian Bokmal and Nynorsk	BMNN_ Norwegian	norbook nornyn	yes	yes	yes
Polish	POLISH		no	yes	yes
Portuguese	PORTUGUESE	portugal	yes	yes	yes
Romanian	ROMANIAN		no	yes	yes
Russian	RUSSIAN	russian	yes	yes	yes
Serbian (latin)	SERBIAN		no	yes	yes
Serbian (cyrillic)	CYR_SERBIAN		no	yes	yes
Slovakian	SLOVAKIAN		no	yes	yes
Slovenian	SLOVENIAN		no	yes	yes
Spanish	SPANISH	espana	yes	yes	yes
Swedish	SWEDISH	svensk	yes	yes	yes
Swiss German	SWISS_ GERMAN	dschweiz	yes	yes	yes
Thai	THAI	thai	yes	yes	yes
Turkish	TURKISH		no	yes	yes
Ukrainian	UKRAINIAN		no	yes	yes
Vietnamese	VIETNAMESE		no	no	yes

(1) The filename of the stop-word file (extension STW) and abbreviation file (extension ABR)

(2) Linguistic processing uses both old and new German spelling.

The files are distinguished by their extension.

Content	Extension
Dictionary	DIC
Stop-word list	STW
Abbreviation list	ABR

Modifying the stop-word and abbreviation files

There is one stop-word file and one abbreviation file per language. To understand the implications of editing these files, see "Why text documents need to be indexed" on page 23.

Tip

Before you begin editing one of these files, make a backup copy.

The stop word and abbreviation files on OS/2 and Windows systems are in:

drive:\%DMBMMPATH%\db2tx\dicts

On AIX, HP-UX, and SUN-Solaris systems, they are in:

DB2TX_INSTOWNER/db2tx/dicts

Use your own editor to edit these files. They use CCSID 850, so ensure that your application CCSID is also set to 850 before you begin.

Remove words and abbreviations that you want to be indexed. Add words that you do not want to be indexed.

Chapter 5. Making text searchable

"Chapter 3. Getting started" on page 19 helps you become familiar with making text searchable by DB2 Text Extender by walking you through a simple example. This chapter describes making text searchable in more detail, and describes all the aspects that you should consider before you begin.

The steps for making text searchable are:

1. Prepare thoroughly
2. Start the DB2 Text Extender command line processor
3. Connect to a database
4. Enable a database for text search
5. Enable a text table for text search (not required if you are creating one index per text column)
6. Enable a text column for text search

Preparation before making text searchable

> **Tip**
>
> Read this section carefully. It lists the options that you need to know about *before* making your text searchable.

- **For UNIX users, set up your profile**

  If you are not a DB2 Text Extender instance owner, refer to "Configuration" on page 13 to find out how to set up your profile.

- **Create one index for the whole table?**

  You must decide whether to create one index for a whole text table, or a separate index for each text column. "Creating one or several text indexes for a table" on page 35 will help you decide.

- **Know your documents**

  When you make documents searchable you must specify their CCSIDs, languages, and the formats of the text. For more information, see "Chapter 4. Planning for your search needs" on page 23.

- **Decide the type of text indexes you need**

  The type of index that you need is determined by the kind of searches you want to make (precise, fuzzy, and so on) and on whether your documents are SBCS or DBCS. You'll find more information in "Chapter 4. Planning for your search needs" on page 23.

- **Decide where to store indexes**

  When you make documents searchable, DB2 Text Extender creates a text index. You must specify in which directory you want the index to be stored. Be sure that there will be enough disk space (see "Calculating the size of an index" on page 36).

- **Check the default database name**

  The default database name in the DB2 environment variable DB2DBDFT (see "Environment variables" on page 13) is the name that DB2 Text Extender uses if you do not specify a database name when making text searchable.

Preparation before making text searchable

- **Set up the text configuration**

  The text configuration determines the default settings for the index CCSID, the documents' language, and the documents' format, the index type, the index update frequency, the tablespace name, and the index directory.

  You can override these settings when you make text searchable, but it is more convenient to have the defaults set correctly beforehand. The initial text configuration settings when DB2 Text Extender is installed are described in "Text configuration settings" on page 14. To change the installation settings and set up your own default values, use "CHANGE TEXT CONFIGURATION" on page 109.

- **Set up section support**

  If you need to restrict searches to a particular section of a document, read "Working with structured documents (section support)" on page 38 to learn how to specify models in the document models file.

- **Modify the stop word and abbreviation lists**

  Read "Why text documents need to be indexed" on page 23 and "Modifying the stop-word and abbreviation files" on page 45 to understand the concept of *stop word* lists and abbreviation lists, and decide whether to modify them before you begin indexing.

Once you have collected the information and made the decisions described in "Preparation before making text searchable" on page 47, you are ready to make your text searchable.

Starting the DB2 Text Extender command line processor

> **Summary**
>
> **When**   Optional. At the beginning of each session.
>
> **Command**   db2tx
>
> **Authorization** Any

You can enter DB2 Text Extender client commands at the operating system prompt. These are similar to DB2 commands, but instead of preceding them with db2, you precede them with db2tx.

An alternative to prefixing every DB2 Text Extender command with db2tx is to start the DB2 Text Extender command line processor. This has the advantage that your database connection is not broken after each command; you remain connected. (Without the command line processor you are automatically reconnected each time you execute a db2tx command, but you must ensure that the name of the database you want to connect to is set in the DB2DBDFT environment variable.)

1. **(Windows only): Start the DB2 command line environment**

   At your operating system prompt, enter:
   ```
   =>db2cmd
   ```

2. **Start the DB2 Text Extender command line processor**

   In UNIX systems, at the operating system prompt enter:
   ```
   =>db2tx
   ```

   In Windows systems, at the DB2 prompt enter:

```
db2=>db2tx
```

The db2tx prompt is displayed:

```
db2tx=>
```

and all subsequent commands are interpreted as DB2 Text Extender commands.

To leave this mode, enter:

```
db2tx=>quit
```

For Windows 95 and Windows 98, you can use the db2tx command as described above, or you can click the Start button, point to Programs, point to DB2 Client for Windows 95/98, and then click DB2 Command Window.

If you leave out this step, you can issue DB2 Text Extender commands directly from the operating system prompt by prefixing them with db2tx. Here is an example of a command issued from the operating system prompt:

```
=>db2tx enable database
```

> **Tip**
>
> If you are working from a client, you must be in the DB2 Text Extender command environment and use the full form of the connect statement, specifying your user ID and password.
>
> ```
> db2tx => connect to database-name user user-id using  password
> ```

Command line processor help

To display a list of commands, enter:

```
db2tx ?
```

To display the syntax of an individual command, enter:

```
db2tx ? command
```

For example:

```
db2tx ? CHANGE TEXT CONFIGURATION
```

Connecting to a database

> **Summary**
>
> | | |
> |---|---|
> | **When** | Optional. To start a session with a database other than the default specified in the DB2DBDFT environment variable. |
> | **Command** | CONNECT |
> | **Authorization** | CONNECT on the database |

Before you can issue further commands in a DB2 Text Extender session, you must be connected to a database. If you issue a command without being connected to a database, DB2 Text Extender automatically connects you to the default database specified in the DB2DBDFT environment variable. You can connect to a database explicitly by using the DB2 Text Extender CONNECT TO command.

```
db2tx CONNECT TO sample-database
```

Connecting to a database

The following is displayed (as seen on AIX):

```
>------------Database Connection Information ------------<
Database Product    = DB2/6000 05.02.0000
User                = user-name
Database alias       = sample-database
```

Note: If you subsequently issue a DB2 Text Extender command outside of the command processor by prefixing it with db2tx, the current database connection is lost and a new connection is made to the default database.

If you are using an OS/2 client, and this is the first time that you are connecting to a database after you have started the system, you are prompted for a user ID and a password by User Profile Management (UPM). If you are already logged on, UPM takes the user ID and password from this session and tries to connect to the remote database on the server.

Enabling a database

> **Summary**
>
> | | |
> |---|---|
> | **When** | Once for each database that contains columns of text to be searched in. |
> | **Command** | ENABLE DATABASE |
> | **Authorization** | SYSADM or DBADM |

This command takes no other parameters. It prepares the connected database for use by DB2 Text Extender.

This command also declares the DB2 Text Extender functions and DB2 Text Extender distinct types to DB2. These are the SQL functions that you use later to search for text. They are described in "Chapter 11. Search functions" on page 171. These declarations apply to all future sessions.

A catalog view, TEXTINDEXES, is created that keeps track of enabled text columns. See "Working with the DB2 Text Extender catalog view" on page 92.

This command creates text configuration information for the database, containing default values for index, text, and processing characteristics. They are described in "Text configuration settings" on page 14.
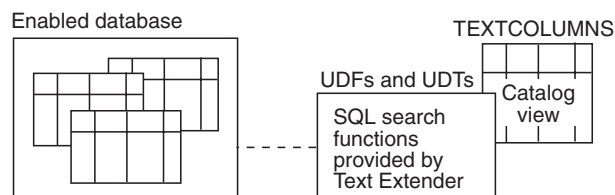
Figure 8. Enabling a database

Once a database has been enabled, it remains so until you disable it. To reverse the changes made by ENABLE DATABASE, refer to "Disabling a database" on page 95.

> **Tips**
>
> If the environment variable DB2TX_INSTOWNER is used, it must be set to the name of the instance owner before the database is enabled. This is particularly important for UNIX users because, in UNIX, this variable is set by default.
>
> If you later decide to drop an enabled database, you should first disable it to ensure that the declared DB2 Text Extender functions, the catalog view, and so on, are removed.

Enabling a text table (optional)

> **Summary**
>
> **When** Optional. Use this command only to create a common index for all text columns in a table. See "Creating one or several text indexes for a table" on page 35.
>
> **Command** db2tx=>ENABLE TEXT TABLE ... (see the examples)
>
> **Authorization** ALTER, SELECT, UPDATE on the table

This step determines whether you have one common index for all the text columns in the table, or several indexes, that is, a separate index for each text column.

To have a common index, run ENABLE TEXT TABLE, then run ENABLE TEXT COLUMN for each text column. To have separate indexes, skip ENABLE TEXT TABLE, and run only ENABLE TEXT COLUMN for each text column. This is shown in Figure 9 and Figure 10 on page 53.

During this step, DB2 Text Extender creates an empty text index that is common to all subsequently enabled text columns. You specify the type of index, how frequently the index is to be updated, and in which directory the index is to be stored. Default values for any parameters that you do not specify are taken from the text configuration settings.

> **Tip**
>
> If a setting, such as the index type, should be the same for most text columns, use the text configuration information to specify default settings. See "Text configuration settings" on page 14.

This step also creates an empty log table for recording which documents in the table are added, changed, or deleted. Triggers are created to keep the log table updated.

You cannot run ENABLE TEXT TABLE for a table that already contains a text column that has been enabled for DB2 Text Extender.

To delete an index created by ENABLE TEXT TABLE, see "Disabling a text table" on page 95.

Enabling a text table (optional)

> **Tip**
> If you later decide to drop an enabled text table, you should first disable it to
> ensure that the index, the log table, and so on, are removed.

Examples

The following example enables text table DB2TX.MYTABLE:

```
db2tx ENABLE TEXT TABLE db2tx.mytable
```

Default values for the index characteristics are taken from the text configuration
settings.

The next example explicitly sets the characteristics of the common index that is
created for the table. For a UNIX server:

```
db2tx ENABLE TEXT TABLE    db2tx.mytable
                  INDEXTYPE  linguistic
                  UPDATEFREQ min(100) d(1,2,3,4,5) h(12,15) m(00)
                  DIRECTORY  DB2TX_INSTOWNERHOMEDIR/db2tx/indexes
```

For an OS/2 or Windows NT or Windows 2000 server:

```
db2tx ENABLE TEXT TABLE    db2tx.mytable
                  INDEXTYPE  linguistic
                  UPDATEFREQ min(100) d(1,2,3,4,5) h(12,15) m(00)
                  DIRECTORY  d:\dmb\db2tx\indexes
```

The example sets the index type and the index directory, and then sets the index
update frequency so that the index is updated at 12:00 or 15:00, on Monday to
Friday, if there is a minimum of 100 text documents queued.

1. Enable text table

2. Enable text column A

3. Enable text column B
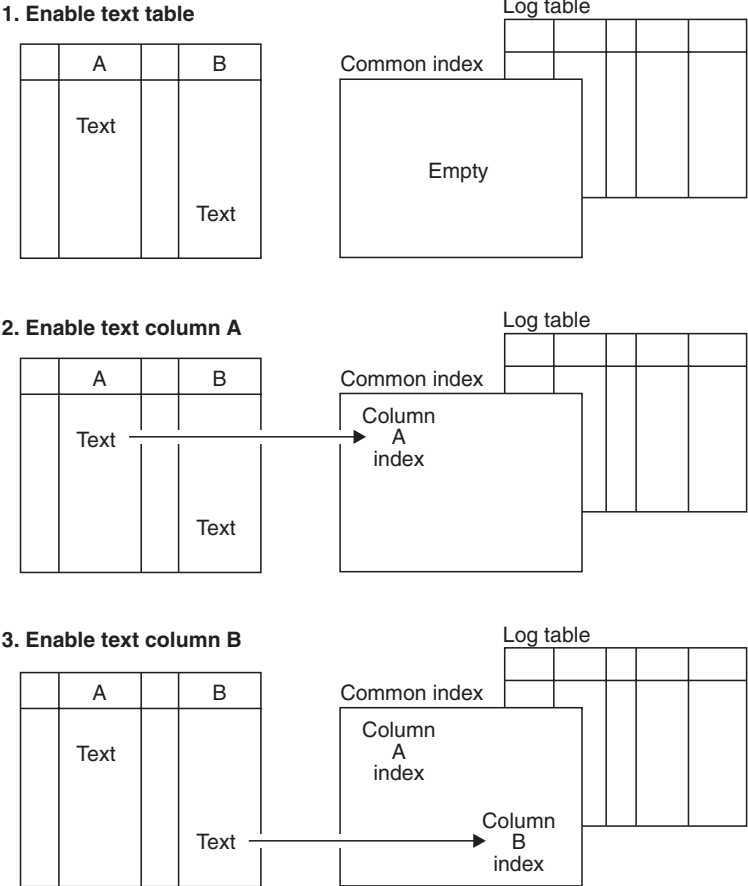
Figure 9. Creating a common index for all text columns in a table

1. Enable text column A

2. Enable text column B
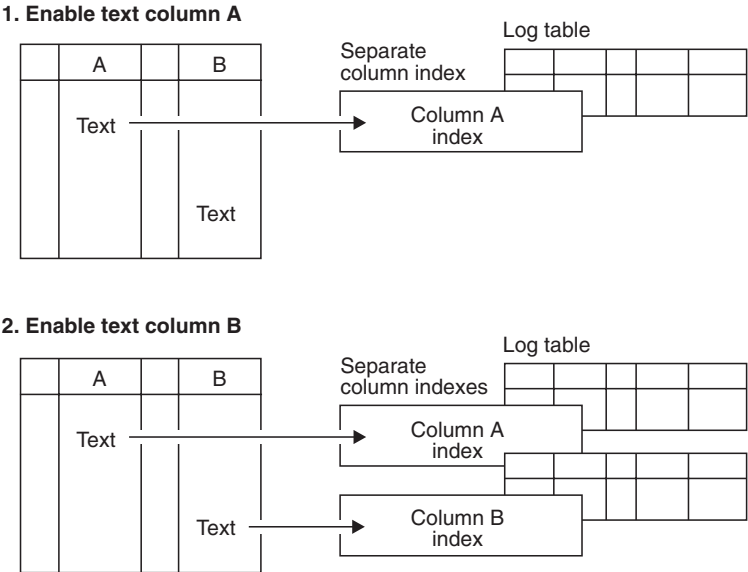
Figure 10. Creating a separate index for each text column

Enabling a text column

Summary

When	Once for each column that contains text to be searched.
Command	db2tx=>ENABLE TEXT COLUMN ... (see the examples)
Authorization	ALTER, SELECT, UPDATE on the table

Tip

If a setting, such as the index type, should be the same for most text columns, use the text configuration information to specify default settings. See "Text configuration settings" on page 14.

To reverse the changes made by ENABLE TEXT COLUMN, use the DISABLE TEXT COLUMN command. To disable all enabled text columns in a table, use the DISABLE TEXT TABLE command.

When you enable a text column, a handle column is added to the table, the document information (format, language, CCSID) is set, a log table is created, and an index is created.

A handle column is added

During this step, DB2 Text Extender adds to the table a 60-byte VARCHAR handle column – a column that contains handles associated with the text column that is being enabled. Handles contain information about the text in the associated text column and in the associated external files. This information includes a unique document ID, the document's language, format, and CCSID, and the index name.

Tip

This version of DB2 Text Extender cannot work with indexes created using an earlier release. You must disable and then re-enable all DB2 Text Extender-enabled databases and tables.

DB2TX.SAMPLE

DOCID	AUTHOR	SUBJECT	DATE	COMMENT
Data	Data	Data	Data	Text

Figure 11. Structure of the DB2TX.MYTABLE table—before enabling

The column containing text blocks is COMMENT. Before you can search through the text in this column, you must prepare the database and the COMMENT column for use by DB2 Text Extender.

After this preparation step, the DB2TX.MYTABLE table contains an additional column for handles.

DB2TX.SAMPLE

DOCID	AUTHOR	SUBJECT	DATE	COMMENT	COMMENTHANDLE
Data	Data	Data	Data	Text	Text handles

Figure 12. Structure of the DB2TX.MYTABLE table—after enabling

Note: When you subsequently search for text, you specify the handle column, not the text column, as the column to be searched.

The document information is set

You specify the type of text documents you typically store in this text column: their format (such as ASCII), their language, and their CCSID. Defaults for this information can be specified in the text configuration settings. See "Text configuration settings" on page 14.

A log table is created

During this step, a log table and a view called LOGIXnnnnnn is created, where *IXnnnnnn* is the index name (available from the catalog view).

If a default tablespace is specified in text configuration, the log table is stored there; otherwise, it is stored in the DB2 system default tablespace. To optimize performance and the use of disk space, you can specify a different tablespace to be used for the log tables.

Triggers are also created that add information to the log table whenever a document in the column is added or changed. This information causes these documents to be indexed the next time indexing takes place.

If external files are added or changed, these triggers are not aware of the changes. In such cases, to cause the triggers to add the information to the log table, use an UPDATE statement as shown in the example in "Updating an index for external files" on page 83.

If errors occur during indexing, such as when a document queued for indexing could not be found, so-called *error events* are added to the log table and can be displayed, as described in "Displaying error events" on page 89.

> **Tip**
> If you run out of log space in this step, see "Enabling a text column in a large table" on page 57 for possible solutions.

In partitioned databases, each table is assigned to a tablespace and a nodegroup. It is important that the log table is assigned to a tablespace that belongs to the same nodegroup as the enabled user table. DB2 Text Extender checks this during the ENABLE command.

An index is created

If you intend to have a separate index for each text column, that is, you have skipped the step ENABLE TEXT TABLE, DB2 Text Extender creates a separate index for the text column during this step. You specify the type of index, how frequently the index is to be updated, and in which directory the index is to be stored. If, on the other hand, you intend to have one index for the whole table,

Enabling a text column

then you have already run ENABLE TEXT TABLE and specified the index parameters; they are ignored if you repeat them here.

Use the UPDATEINDEX keyword to determine whether the indexing of the text documents in the specified text column begins immediately, or when periodic indexing is next scheduled. If you do not use this keyword, the value specified in the text configuration settings is taken.

Creating indexes of various types for a text column. You can create more than one index for a text column. This can be useful if you want to allow, for example, linguistic and fuzzy search on the same text column, by associating the column with different index types, such as linguistic and Ngram indexes. You do this by running ENABLE TEXT COLUMN again, specifying not only the additional type of index to be created, but also a unique handle column name.

> **Tip**
>
> If you are using a partitioned database, and you decided to change the configuration of a nodegroup and start a redistribution of the table rows, you must recreate the text indexes and log tables, that is, you must disable and enable again all DB2 Text Extender-enabled tables of that nodegroup.

Examples

The following example enables text column COMMENT in table DB2TX.MYTABLE, and assigns the name COMMENTHANDLE to the handle column that is created:

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                 HANDLE     commenthandle
```

Default values for the text information and for the index characteristics are taken from the text configuration settings.

The next example explicitly sets the values for the type of documents that are in the COMMENT column. Default values for the index characteristics are taken from the text configuration settings.

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                 HANDLE     commenthandle
                 CCSID      819
                 LANGUAGE   uk_english
                 FORMAT     rft
```

The next example explicitly sets the values for the characteristics of the index that is created for the COMMENT column. The example sets the index type and the index directory, and sets the index update frequency so that the index is updated at 12:00 or 15:00, on Monday to Friday, if there is a minimum of 100 text documents queued. Default values for the text information are taken from the text configuration settings.

For a UNIX server:

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                 HANDLE      commenthandle
                 INDEXTYPE   linguistic
                 UPDATEFREQ  min(100) d(1,2,3,4,5) h(12,15) m(00)
                 UPDATEINDEX UPDATE
                 DIRECTORY   DB2TX_INSTOWNERHOMEDIR/db2tx/indexes
```

For an OS/2 or Windows NT or Windows 2000 server:

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                  HANDLE      commenthandle
                  INDEXTYPE   linguistic
                  UPDATEFREQ  min(100) d(1,2,3,4,5) h(12,15) m(00)
                  UPDATEINDEX UPDATE
                  DIRECTORY   d:\dmb\db2tx\indexes
                  TABLESPACE  mytablespace
                  STOGROUP    mystoragegroup
```

Enabling a text column in a large table

If you are working with a table that has a large row length, keep in mind that enabling a text column adds a handle column of type DB2TEXTH (VARCHAR 60). This could be significant if the table is approaching its maximum row length as determined by DB2.

Also when you enable a text column in large table, use the DB2 UDB REORGANIZE utility to check whether the table needs to be reorganized. When you enable a large table for the first time, the following steps make indexing faster:

1. Enable the table using the NOUPDATE option. This creates the handles, but does not yet index the documents.
2. Reorganize the table using the DB2 UDB REORGANIZE utility.
3. Create the index by running UPDATE INDEX.

When you enable a text column or external files, DB2 Text Extender adds a handle column to the table and initializes the handle values, thereby causing DB2 UDB log entries to be written. If there is an unusually large number of log entries to be written, DB2 UDB can run out of log space.

There are two ways to handle this situation; the first is better for performance reasons:

- Increase the available log space by using the DB2 UDB UPDATE DB CFG command to modify the database configuration parameters for LOGPRIMARY, LOGSECOND, and LOGFILSIZ. The following values are taken from experience; you may need to change them to suit your installation.

  ```
  LOGSECOND     50
  ```

  Ensure that the sum of LOGPRIMARY and LOGSECOND is not greater than 128. You should also increase the application heap size.

  ```
  APPLHEAPSZ    512
  ```
- Force DB2 UDB to make an intermediate COMMIT by using the COMMITCOUNT configuration parameter described in "Configuration" on page 13. The value you specify indicates after how many insert or update statements DB2 Text Extender issues a DB2 UDB commit statement. This reduces the size required for log tables, although it also increases the time required for the enabling step.

Enabling text columns of a nonsupported data type

Text columns must be CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB to be enabled by DB2 Text Extender. If the documents are in a column of a different type, such as a user-defined distinct type (UDT), you must provide a function that takes the user type as input and provides as output type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB.

Enabling a text column

Use the FUNCTION keyword in ENABLE TEXT COLUMN to specify the name of this function.

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

1. Create a user-defined distinct 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 MYTABLE (author VARCHAR(50),
                          text   COMPRESSED_TEXT)
db2 INSERT ...
```

> **Note**
>
> Creating a table is outside the parameters of this guide. See the *DB2: SQL Reference* guide for further information

To enable the text column for use by DB2 Text Extender:

1. Create a user-defined function (UDF) called, for example, UNCOMPRESS, that receives a value of type COMPRESSED_TEXT and returns the corresponding uncompressed text as, for example, a CLOB(10M) value.

2. Enable the text column using the FUNCTION keyword to identify the UNCOMPRESS UDF:

```
db2tx ENABLE TEXT COLUMN MYTABLE  text
                         FUNCTION uncompress
                         HANDLE   handle
                         ...
```

Enabling columns containing data type DATALINK

DB2 Text Extender supports the data type DATALINK. DB2's Data Link technology provides referential integrity between data stored in DB2 and data in external files. It does this by providing a new DATALINK data type that holds references to an object stored external to the database. A DATALINK value stored in a DB2 table encodes the name of the Data Links server containing the file and the file name in terms of a Uniform Resource Location (URL).

If the DATALINK data is on external files rather than in a DB2 table, then follow the description in "Enabling external text files" on page 59.

Before enabling a text column containing the DATALINK data type, it is important that you run the installation verification sample (see "Preparing a sample database for installation verification" on page 12).

AIX example: The following statement shows how a data link value is inserted into your text column using data type DATALINK in AIX:

```
db2 "insert into mytable values ('Peter',
                  dlvalue ('http://dlfs.boeblingen.ibm.com/test/documents/letter1',
                           'URL', 'this is a letter from Peter'))"
```

In this example, *dlfs.boeblingen.ibm.com* is the file-server name, *test* is the file-system name, and *documents/letter1* is the path and file name.

Ensure that the file system is mounted on the DB2 server with the same name as specified in the URL.

Windows NT or Windows 2000 example: The following statement shows how a data link value is inserted into your text column using data type DATALINK in Windows:

```
db2 "insert into mytable ('Peter',
            dlvalue ('unc:\\dlfs.boeblingen.ibm.com\cdrive\documents\letter1',
                     'URL', 'this is a letter from Peter'))"
```

In this example, *cdrive* is the share name of the drive where the file is stored.

If you encounter problems while indexing documents stored in a datalink column, make sure that you can access the documents from your DB2 server using the APIs provided by DB2.

Enabling external text files

This option is provided only to maintain compatibility with previous releases. The preferred method for indexing files not stored in DB2 tables is to use a column having a DATALINK data type. See "Enabling columns containing data type DATALINK" on page 58.

Summary	
When	Optional. Once for each table associated with external files that are to be searched.
Command	db2tx=>ENABLE TEXT FILES ... (see examples)
Authorization	ALTER, SELECT, UPDATE on the table

DB2 Text Extender can search not only in text stored in DB2 UDB tables, but also in text documents stored in files. This preparation step is needed if you intend to search for text in external files. The table associated with the external text files must not have been enabled by the command ENABLE TEXT TABLE.

An index is created, a log table is created, and the document information is set, in the same way as described in "Enabling a text column" on page 54.

Tips
If you run out of log space in this step, see "Enabling a text column in a large table" on page 57 for possible solutions.
If the table you are enabling is partitioned and using a nodegroup that has multiple physical nodes, make sure that the external files you are referencing in the table columns are located on the node where their respective table partition resides.

A handle column of type DB2TEXTFH is added to an existing DB2 UDB table. The handle column will hold the references for the external files, each handle containing index and the document information (CCSID, format, and language).

See "Handles for external files" on page 66 for a description.

You can specify additional parameters, such as the default index characteristics, in the same way as for enabling a text column.

Enabling external text files

After the index has been created, you can move or delete the external files. You can still search on the files. You can insert new rows in the table and use UPDATE INDEX to update the index with the new file references.

If the table you are enabling is using a nodegroup with multiple physical nodes, make sure that the external files you are referencing in the columns of your table are located on the node where the table partition resides.

Examples

1. Create a table DB2TX.EXTFILE having at least one column.
2. Insert data into the table column.

   For each row in a new table:

   ```
   db2 INSERT INTO db2tx.EXTFILE (FILEHANDLES)
   VALUES (db2tx.INIT_TEXT_HANDLE
              (850, 'TDS' 'US_ENGLISH',
              'd:\dmb\samples\tx')
   ```

3. Add handle column FILEHANDLES to table DB2TX.EXTFILE

   ```
   db2tx ENABLE TEXT FILES  db2tx.extfile
              HANDLE      filehandles
              INDEXTYPE   linguistic
              UPDATEFREQ  min(100) d(1,2,3,4,5) h(12,15) m(00)
              UPDATEINDEX NOUPDATE
              DIRECTORY   \any\db2tx\indices
   ```

4. Initialize the handle Update the handle columns to reflect the external file reference, specifying the name of the external file:

   ```
   db2 UPDATE db2tx.EXTFILE
   SET FILEHANDLES = db2tx.file(FILEHANDLES,'d:\dmb\samples\tx')
   WHERE DOCID = 'doc1'
   ```

   > **Tip**
   > Do not use INIT_TEXT_HANDLE for updating handle columns that refer to external files.

5. Update the index

   ```
   db2tx UPDATE INDEX  db2tx.extfile
              HANDLE  filehandles
   ```

Ending the session

You have now completed the steps to prepare your text documents to be searched.

If you specified NOUPDATE for the UPDATEINDEX keyword when you enabled the text column, DB2 Text Extender does not index the text immediately, but waits for the next periodic indexing. To update the index now, see "Updating an index" on page 83.

When indexing of the documents has finished, you can begin retrieving information as described in "Chapter 6. How to search" on page 63.

> **Tip**
> Use GET INDEX STATUS to determine when indexing has finished.

To end the DB2 Text Extender command processor, enter:

```
db2tx=>quit
```

Ending the session

Chapter 6. How to search

DB2 Text Extender provides SQL functions that enable you to include text search subqueries in SQL queries. These functions are provided in addition to those normally available in SQL, and are referred to here as DB2 Text Extender functions.

Refer to "Chapter 11. Search functions" on page 171 for a description of the syntax of these functions.

Before searching, read "Types of search" on page 32, and also use GET INDEX SETTINGS to find out which index type is associated with the text you are searching in. A search can produce different results according to the index type. The index type assumed in the examples in this chapter is linguistic.

This chapter describes:
- The sample DB2 Text Extender functions
- The sample table DB2TX.SAMPLE
- Handles for external files
- Setting the function path to give SQL access to the DB2 Text Extender functions
- Searching for text, using CONTAINS, NO_OF_MATCHES, and RANK
- Specifying search arguments in DB2 Text Extender functions, using examples of CONTAINS
- Refining a previous search, using CONTAINS and REFINE
- Setting and extracting information in handles, using INIT_TEXT_HANDLE, CCSID, FORMAT, and LANGUAGE
- Improving search performance, using SEARCH_RESULT.

Where to find syntax examples of search functions

DB2 Text Extender provides a command line processor input file called `txsample.udf`. It contains examples of DB2 Text Extender search functions that run against the sample table. Use this file to see examples of the syntax of the DB2 Text Extender text preparation and search function, and of the syntax used in search arguments.

The sample file is stored in the `samples` subdirectory

To run this file, enter:
```
db2 -t -v -f txsample.udf
```

The sample table DB2TX.SAMPLE

The sample table DB2TX.SAMPLE is used in many of the search syntax examples. You can try these examples yourself using DB2 Text Extender.

To create table DB2TX.SAMPLE, see "Preparing a sample database for installation verification" on page 12.

An extract from the DB2TX.SAMPLE table is shown in Table 6 on page 64.

The sample table DB2TX.SAMPLE

Table 6. An extract from the example table DB2TX.SAMPLE

DOCID	AUTHOR	SUBJECT	DATE	COMMENT
doc 5	RSSHERM at CHGVMIC1	LIBDB2E.A error	1995-07-25 -20.13.59	Customer is getting a 'No such file or directory' on LIBDB2E.A. It does not appear to be the same error message that relates to the asynchronous I/O driver. He is using beta 4 on 3.2.5. I have had him compare the permissions and ownership of /usr/lpp/db2_02_01/lib files with mine, and they are now the same. His .profile and ENV also look good. He has, unfortunately, COMMITTED the install. What else could be wrong.
doc 6	EDWARDSC at SYDVM1	Lowercase Userid and Password from DDCS/2	1995-07-25 -20.15.20	After rechecking, the instance where I had problems with case-sensitivity was using a DB2/2 gateway to MVS. It didn't like it when I passed a lower case userid (didn't care about passwd). Connection was only successful if I actually typed an upper case userid. So, I guess this doesn't help your situation. Sorry.
doc 7	SKY at TOROLAB4	ODBC & Stored Procedures	1995-07-25 -20.42.27	There are two sets of sample programs explaining the use of Stored Procedures using CLI (ODBC). The C file inpsrv2.c (placed on the server), and the C file inpcli2.c (placed on the client) make up the sample that demonstrates using stored procedures for input. The files outsrv2.c and outcli2.c make up the sample that demonstrates using stored procedures for output. These files are part of the .../sqllib/samples/cli files. The MAKE file will automatically build them and transfer the server file to the correct subdirectory.
doc 8	ADAMACHE at TOROLAB2	DB2SYS.DLL access violation	1995-07-25 -21.13.22	Did you have a previous beta version installed? If so, did you remove it using Software Installer? Did you remove the database directories (SQLDBDIR and SQL00001, etc.) from previous beta drivers?

Table 6. An extract from the example table DB2TX.SAMPLE (continued)

DOCID	AUTHOR	SUBJECT	DATE	COMMENT
doc 9	ADAMACHE at TOROLAB2	CREATE DB = SYS3175: db2sysc.exe in db2eng.dll	1995-07-25 -21.40.09	Many DB2/2 beta users delete a previous beta with Software Installer, install beta 5 (or golden code now), create a database, and get: SYS3175: db2sysc.exe in db2eng.dll This happens because the directory format has changed between beta4 and beta5. Our DB2/2 installation does not migrate the sqldbdir directory between beta drivers. You should remove all occurrences of sqldbdir and sql000x directories and \sqllib\db2\sqldbdir directory. What you should do is delete the previous beta with Software Installer, remove all occurrences of sqldbdir and sql000x directories and \sqllib\db2\sqldbdir directory, and then install the new code.
doc 10	RSSHERM at CHGVMIC1	DB2/NT - SNA support	1995-07-25 -22.10.15	Will DB2/NT be able to act as both a server to CAE/WIN clients and also as a client (hopping) to DB2/6000 and/or DB2/MVS over an SNA network? The other alternative would be DRDA from DB2/NT to DB2/6000 and/or DB2/MVS - again via SNA, which I assume is supported?

Here is a part of the table structure showing the first and last columns:

DB2TX.SAMPLE

DOCID		COMMENT
doc 1		Customer is ...
doc 2		After rechecking ...

Figure 13. The structure of the DB2TX.SAMPLE table

The column containing text to be searched is COMMENT. Before you can search through the text in this column, however, you must prepare the COMMENT column for use by DB2 Text Extender using the ENABLE TEXT COLUMN command.

After this preparation step, the DB2TX.SAMPLE table looks like this:

DB2TX.SAMPLE

DOCID		COMMENT	COMMENTHANDLE
doc 1		Customer is ...	X'..handle..'
doc 2		After rechecking ...	X'..handle..'

Figure 14. The DB2TX.SAMPLE table after being enabled

The table now has an additional column for handles, and each text object has a unique handle that represents it.

The sample table DB2TX.SAMPLE

When you later insert text into an enabled text column, an insert trigger creates a handle for it.

DB2TX.SAMPLE

DOCID		COMMENT	COMMENTHANDLE	
doc 1		Customer is ...	X'..handle..'	Handles created by
doc 2		After rechecking ...	X'..handle..'	ENABLE TEXT COLUMN

Inserted row:

doc 11		I have installed ...	X'..handle..'	Handle created by an insert trigger

Figure 15. The handle for an inserted row is created by a trigger

A handle contains the following information:

A document ID

The name and location of the associated index

The document information: CCSID, format, and language.

The SQL functions provided by DB2 Text Extender take a handle as a parameter and store, access, search for, and manipulate the text as part of the SQL processing of the table.

Handles for external files

DB2 Text Extender can search not only in text stored in DB2 UDB tables, but also in text files stored elsewhere. "Enabling external text files" on page 59 describes the preparation step that makes it possible to search in text documents that are not stored in DB2 UDB tables. In this step, the ENABLE TEXT FILES command creates a handle column of type DB2TEXTFH for external-file handles. The handle column is added to an existing table.

You could, for example, create a table that contains columns for the name of the author and for the date when the document was created.

You initialize the files' handles using INIT_TEXT_HANDLE. Each handle contains not only a document ID, the name and location of the associated index, and the document information (CCSID, format, and language), but also the reference to the external file.

Setting the current function path

```
►►──SET──CURRENT FUNCTION PATH───────────────DB2TX, ...──────────────────◄
                              └─ = ─┘
```

Use the SQL statement SET CURRENT FUNCTION PATH to add DB2TX to your current path names so that SQL can find the DB2 Text Extender functions. If you decide not to do this, you can qualify the function names explicitly by typing, for example, DB2TX.CONTAINS for the CONTAINS function.

The examples in this chapter use the qualified form for DB2 Text Extender functions. You can use the example statements exactly as they are written without having to set the current function path.

> **Tip**
> Remember to set the current function path each time you connect to a database.

Searching for text

```
►►─┬─CONTAINS──────┬──(─handle─,─search-argument─)────────────────────►◄
   ├─NO_OF_MATCHES─┤
   └─RANK──────────┘
```

```
►►──SEARCH_RESULT──(─schema─,─table─,─handle─,─search-argument─)─────►◄
```

This section describes how to use the SQL functions provided with DB2 Text Extender to search in DB2 databases containing text. It tells you how to:

- Make a query
- Determine how many matches were found in a text document
- Get the rank of a found text document.

The use of SEARCH_RESULT is described in "Improving search performance" on page 81.

Each of these DB2 Text Extender functions searches in the text index for occurrences of the search argument. If there are, say, 100 000 text documents in the table, the CONTAINS, RANK, or NO_OF_MATCHES function is called 100 000 times. But the text index is not searched 100 000 times. Instead, the first time the function is called, an internal list of all the documents containing the search term is created; subsequent calls of the function determine if the document concerned is in the list.

> **Tip**
> When you use the DB2 Text Extender functions to search in a table, be sure to pass the handle column to the function, rather than the text column. If you try to search in a text column, SQL responds with a message indicating that the data type is wrong, for example:
> ```
> No function by the name "CONTAINS" having compatible
> arguments was found in the function path.
> ```

If you search for text immediately after issuing the ENABLE TEXT TABLE or ENABLE TEXT COLUMN command, an error RC_SE_EMPTY_INDEX can occur which indicates that the index being created by the command does not yet exist. The time taken for an index to be created depends on factors such as the number of documents being indexed, and the performance of the system doing the

indexing. It can vary from several minutes to several hours, and should be done when the system is lightly loaded, such as over night.

If this message occurs, try searching again later, or use GET INDEX STATUS to check whether indexing errors have occurred.

Making a query

This example demonstrates how the CONTAINS function searches for text in documents identified by a handle. It returns 1 if the text satisfies the search argument, otherwise it returns 0.

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"compress"') = 1
```

In this example, you search for the term compress in the text referred to by the handles in the column COMMENTHANDLE. The handles in the COMMENTHANDLE column indicate where the COMMENT text is indexed.

> **Tip**
>
> If you have created mixed-case identifiers for tables or columns, that these names must be enclosed in double quotes. For example:
>
> ```
> db2=>SELECT DATE, SUBJECT
>         FROM "db2tx.sample"
>         WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"compress"') = 1
> ```
>
> If you specify DB2 UDB select statements from the command line, the operating system command-line parser removes special characters such as double quotes from the command string, so you must use a backslash to mask these special symbols. For example:
>
> ```
>         WHERE DB2TX.CONTAINS (COMMENTHANDLE, '\"compress\"') = 1
> ```

> **Tip**
>
> If your queries result in database timeout errors, you should increase the UDF_TIME_OUT value of your QQAINI file settings.

Searching and returning the number of matches found

Use the NO_OF_MATCHES function to determine how often the search criteria are found in each text document.

```
db2=>WITH TEMPTABLE(DATE, SUBJECT, MATCHES)
        AS (SELECT DATE, SUBJECT,
                    DB2TX.NO_OF_MATCHES(COMMENTHANDLE,'"compress"')
            FROM DB2TX.SAMPLE)
        SELECT *
            FROM TEMPTABLE
            WHERE MATCHES > 0
```

NO_OF_MATCHES returns an integer value.

Searching and returning the rank of a found text document

RANK is an absolute value that indicates how well the document met the search criteria relative to other found documents. The value indicates the number of

matches found in the document in relation to the document's size. You can get the rank of a found document by using the RANK function.

Here is an example:

```
db2=>WITH TEMPTABLE(DATE, SUBJECT, RANK)
     AS (SELECT DATE, SUBJECT,
                DB2TX.RANK(COMMENTHANDLE,'"compress"')
         FROM DB2TX.SAMPLE)
SELECT *
        FROM TEMPTABLE
        WHERE RANK > 0
        ORDER BY RANK DESC
```

RANK returns a DOUBLE value between 0 and 1.

Specifying search arguments

Search arguments are used in CONTAINS, NO_OF_MATCHES, HANDLE_LIST, and RANK. This section uses the CONTAINS function to show different examples of search arguments in DB2 Text Extender functions.

Searching for several terms

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:

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
          '("compress", "compiler", "pack", "zip", "compact")') = 1
```

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

Searching with the Boolean operators AND and OR

(See also "Searching with the Boolean operator NOT" on page 73.)

Search terms can be combined with other search terms using the Boolean operators "&" (AND) and "!" (OR). For example:

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              '"compress" | "compiler"') = 1
```

You can combine several terms using Boolean operators:

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              '"compress" | "compiler" & "DB2"') = 1
```

If you use more than one Boolean operator, DB2 Text Extender evaluates them from left to right, but the logical AND operator (&) binds stronger than the logical OR operator (!). For example, if you do not include parentheses,

```
    "DB2" & "compiler" | "support" & "compress"
```

is evaluated as:

```
    ("DB2" & "compiler") | ("support" & "compress")
```

So in the following example you must include the parentheses:

```
"DB2" & ("compiler" | "support") & "compress"
```

If you combine Boolean operators with search terms chained together using the comma separator, like this:

```
("compress", "compiler") & "DB2"
```

the comma is interpreted as a Boolean OR operator, like this:

```
("compress" | "compiler") & "DB2"
```

Searching for variations of a term

If you are using a **precise** index, DB2 Text Extender searches for the terms exactly as you type them. For example, the term media finds only text that contains "media". Text that contains the singular "medium" is not found.

If you are using a **linguistic** index, DB2 Text Extender searches also for variations of the terms, such as the plural of a noun, or a different tense of a verb.

For example, the term drive finds text that contains "drive", "drives", "driving", "drove", and "driven".

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 represent optional characters at the front, middle, or end of a search term. They increase the number of text documents found by a search.

> **Tip**
> If you use masking characters, you cannot use the SYNONYM FORM OF keyword.

Masking characters are particularly useful for finding variations of terms if you have a precise index. If you have a linguistic index, many of the variations found by using masking characters would be found anyway.

Note that word fragments (words masked by wildcard characters) cannot be reduced to a base form. So, if you search for passe%, you will not find the words "passes" or "passed", because they are reduced to their base form "pass" in the index. To find them, you must search for pass%.

DB2 Text Extender uses two masking characters: percent (%) and underscore (_):

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

```
db2=>SELECT DATE, SUBJECT
     FROM DB2TX.SAMPLE
     WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"%name"') = 1
```

This search term finds text documents containing, for example, "username", "file_name", and "table-name".

% can also represent a **whole word**: The following example finds text documents containing phrases such as "graphic function" and "query function".

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"% function"') = 1
```

- _ represents **one character** in a search term: The following example finds text documents containing "CLOB" and "BLOB".

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"_LOB"') = 1
```

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 by a so-called *escape* character, and then identify the escape character using the ESCAPE keyword.

For example, to search for "100% interest":

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                          '"100!% interest" ESCAPE "!"') = 1
```

The escape character in this example is "!".

Searching for terms in any sequence

If you search for "hard disk" as shown in the following example, you find the two terms only if they are adjacent and occur in the sequence shown, regardless of the index type you are using.

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"hard disk"') = 1
```

To search for terms in any sequence, as in "data disks and hard drives", for example, use a comma to separate the terms:

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE, '("hard", "disk")') = 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:

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                       '"compress" IN SAME SENTENCE AS "decompress"') = 1
```

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

```
db2=>SELECT DATE, SUBJECT
        FROM DB2TX.SAMPLE
        WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                          '"compress" IN SAME PARAGRAPH AS "decompress"
                                          AND "encryption"') = 1
```

Searching for terms in sections of structured documents

Here is an example of a search argument that finds text documents in which the search term Williams occurs in the subsection author in section play of structured

documents. The document structure is specified by model `play` which is described in a document models file. See "Working with structured documents (section support)" on page 38 for more information.

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                     'MODEL play SECTIONS (play/author) "williams"') = 1
```

Searching for synonyms of terms

For a linguistic index, you can make your searches more flexible by looking not only for the search terms you specify, but also for words having a similar meaning. For example, when you search for the word "book", it can be useful to search also for its synonyms. To do this, specify:

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE, 'SYNONYM FORM OF "book"') = 1
```

When you use SYNONYM FORM OF, it is assumed that the synonyms of the term are connected by a logical OR operator, that is, the search argument is interpreted as:

```
"book" | "article" | "volume" | "manual"
```

The synonyms are in a dictionary that is provided with DB2 Text Extender. The default dictionary used for synonyms is always US_ENGLISH, not the language specified in the text configuration settings.

You can change the dictionary for a particular query by specifying a different language. Here is an example:

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                     'SYNONYM FORM OF UK_ENGLISH "programme"') = 1
```

> **Tip**
> You cannot use the SYNONYM keyword if there are masking characters in a search term, or if NOT is used with the search argument.

Making a linguistic search

DB2 Text Extender offers powerful linguistic processing for making a search based on the search terms that you provide. The linguistic functions are applied when the index is linguistic. The linguistic functions are described in "Chapter 15. Linguistic processing for linguistic and precise indexes" on page 221.

An example of this is searching for a plural form, such as "utilities", and finding "utility". The plural is reduced to its base form `utility`, using an English dictionary, before the search begins.

The English dictionary, however, does not have the information for reducing variations of terms in other languages to their base form. To search for the plural of a term in a different language you must use the dictionary for that language.

If you specify GERMAN, for example, you can search for "geflogen" (flown) and find all variations of its base form "fliegen" (fly)—not only "geflogen", but also "fliege", "fliegt", and so on.

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              'STEMMED FORM OF GERMAN "geflogen"') = 1
```

> **Tip**
> When searching in documents that are not in U.S. English, specify the
> language in the search argument *regardless of the default language*.

If you always specify the base form of a search term, rather than a variation of it, you do not need to specify a language.

To understand why, consider what happens when the text in your database is indexed. If you are using a linguistic index, all variations of a term are reduced to their base form before the terms are stored in the index. This means that, in the DB2TX.SAMPLE table,although the term "decompress" occurs in the first entry in the COMMENT column, "decompression" occurs in the second entry, the index contains only the base form "decompress" and identifies this term (or its variations) as being in both entries.

Later, if you search for the base form "decompress", you find all the variations. If, however, you search for a variation like "decompression", you cannot find it directly. You must specify an appropriate dictionary for the search, so that the variation can first be converted to its base form.

Searching with the Boolean operator NOT

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

```
("compress", "compiler") & NOT "DB2"
```

Any text documents containing the term "DB2" are excluded from the search for "compress" or "compiler".

You cannot use the NOT operator in combination with IN SAME SENTENCE AS or IN SAME PARAGRAPH AS described in "Searching for terms in the same sentence or paragraph" on page 71, neither can you use it with SYNONYM FORM OF described in "Searching for synonyms of terms" on page 72.

You can use the NOT operator only with a search-primary, that is, you cannot freely combine the &, |, and NOT operators (see "Search argument syntax" on page 188).

Example of the use of NOT that is **not** allowed:

```
    NOT("compress" & "compiler")
```

Allowed is:

```
    NOT("compress" , "compiler")
```

Fuzzy search

"Fuzzy" search searches for words that are spelled in a similar way to the search term. It is available for Ngram indexes.

For example:

```
db2=>SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              'FUZZY FORM OF 2 "compress"') = 1
```

This search could find an occurrence of the misspelled word conpress.

The match level, in the example "2", specifies the degree of accuracy. Five levels are supported, where level 1 gives the loosest matching of about 20 percent, and level 5 gives the tightest matching of about 90 percent. Use a fuzzy search when the misspellings are possible in the document, as is often the case when the document was created using an Optical Character Recognition device, or phonetic input.

Respecting word-phrase boundaries

"Bound" search has been developed for the Korean language. It ensures that DB2 Text Extender respects word boundaries during the search. For example:

```
db2=>SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              'BOUND "korean-expression"') = 1
```

Searching for similar-sounding words

"Sound" search finds words that sound like the search argument. This is useful when documents can contain words that sound alike, but are spelled differently. The German name that is pronounced my-er, for example, has several spellings.

```
db2=>SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                              'SOUNDS LIKE "Meyer"') = 1
```

This search could find occurrences of "Meyer", "Mayer", and "Maier".

Thesaurus search

Thesaurus search is another of DB2 Text Extender's powerful search-term expansion functions. The additional terms searched for are taken from a thesaurus that you build yourself, so you have direct control over them. You search for "database", for example, and could find terms like "repository" and "DB2".

This type of search is intended for specific areas of interest in which you make frequent searches; an area in which it is worth the investment in time to build a thesaurus in order to produce significantly more effective search results.

See "Thesaurus concepts" on page 231 for more information and a description of how to build a thesaurus. The example in Figure 18 on page 232 is a small extract from a thesaurus on the subject of databases. It is used in the following examples that demonstrate the syntax for using thesaurus expansion.

This example takes the term "object relational database management system" and expands it, adding all *instances* of this term found in the thesaurus "myterms". Here, "DB2" is added to the search.

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                    'THESAURUS "myterms"
                    EXPAND "INST"
                    TERM OF "object relational database management system"') = 1
```

The next example takes the term "document management system" and expands it, adding all its *synonyms*. There is one synonym – "library".

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                    'THESAURUS "myterms"
                    EXPAND "SYN"
                    TERM OF "document management system"') = 1
```

Free-text and hybrid search

"Free-text search" is a search in which the search term is expressed 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 are not relevant, although for the set of query terms, at least one of the terms must occur in the documents to be searched. Furthermore, so-called *lexical affinities* are supported for ranking. In retrieval, these are certain pairs of words occurring in a free-text query term, and occurring in the document collection, with a certain minimal frequency and a certain minimal distance. The distance for English documents is five words.

Note that the masking of characters or words is not supported for search strings in a free-text argument.

For example:

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
            'IS ABOUT "everything related to AIX installation"') = 1
```

Hybrid search is a combination of Boolean search and free-text search. For example:

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE,
            '"DB2" & IS ABOUT "everything related to AIX installation"') = 1
```

Refining a previous search

When a search argument finds too many occurrences, it can often be useful to narrow, or *refine*, the search by combining the initial search argument with a second search argument in a Boolean-AND relationship.

You can refine search results without using the REFINE function, by storing the results in a table and making the next search against this table. However, depending on the number of qualifying terms, this method is less efficient than that of storing the latest search argument and using REFINE.

Refining a previous search

The following steps show how to make a search, and then refine it using the REFINE function. The REFINE function returns a search argument that is a Boolean-AND combination of its two input parameters. The combined search argument returned by REFINE is a value of type LONG VARCHAR.

1. Create a table for old search arguments.

   Create a table PREVIOUS_SEARCHES to hold the search arguments of searches that have already been made.

   ```
   db2=>CREATE TABLE PREVIOUS_SEARCHES (step INT,
                                        searchargument LONG VARCHAR)
   ```

   PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT

2. Search for the first search argument.

   Search for the word "compress" in the sample table.

   ```
   db2=>SELECT COMMENT
          FROM DB2TX.SAMPLE
          WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"compress"') = 1
   ```

   Insert the search argument into the PREVIOUS_SEARCHES table for use by further steps.

   ```
   db2=>INSERT INTO PREVIOUS_SEARCHES
          VALUES (1, '"compress"')
   ```

   PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT
1	"compress"

3. Refine the search.

   Assuming that the search returns too many text documents, refine the search by combining the previous search term with the word "compiler" using the REFINE function.

```
db2=>WITH LAST_STEP(STEP_MAX)
         AS (SELECT MAX(STEP)
             FROM PREVIOUS_SEARCHES),
         LAST_SEARCH(LAST_SEARCH)
         AS (SELECT SEARCHARGUMENT
             FROM PREVIOUS_SEARCHES,LAST_STEP
             WHERE STEP = STEP_MAX)
     SELECT COMMENT
         FROM DB2TX.SAMPLE, LAST_SEARCH
         WHERE DB2TX.CONTAINS(COMMENTHANDLE,
                   DB2TX.REFINE(LAST_SEARCH, '"compiler"')) = 1
```

   Insert the refined search argument into the PREVIOUS_SEARCHES table for use by further steps.

```
db2=>INSERT INTO PREVIOUS_SEARCHES
        WITH LAST_STEP(STEP_MAX)
           AS (SELECT MAX(STEP)
               FROM PREVIOUS_SEARCHES)
        SELECT STEP_MAX+1, DB2TX.REFINE(SEARCHARGUMENT, '"compiler"')
          FROM PREVIOUS_SEARCHES, LAST_STEP
```

PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT
1	"compress"
2	"compress" & "compiler"

   You can repeat this step until the number of text documents found is small enough.

Setting and extracting information in handles

Handles contain the CCSID, format, and language of their text documents. Handles for external files contain additionally a pointer to the external file. These handles are created when you enable a text column or external files.

The DB2 Text Extender functions described here let you set and extract the text information in the handles.

Setting text information when inserting new text

```
►►──INIT_TEXT_HANDLE──┬─(─format─,─language─)─────────────────────┬─►◄
                      └─(─CCSID─,─format─,─language─,─filename─)──┘
```

When you run the ENABLE TEXT COLUMN command to enable a text column that already contains text, you can implicitly set the format and language of the text to the values specified in the text configuration settings. These format and language settings are then stored in the handle. If you want different format and language values, you can specify them explicitly in the ENABLE TEXT COLUMN command.

When you run the ENABLE TEXT FILES command, you can also set the document's CCSID and location.

When you later insert a row containing text, an insert trigger creates a handle and sets the text format and language to the values that were used when the text column was enabled.

To set the format and language to values that are different from these values, use the INIT_TEXT_HANDLE function in the INSERT command. While the row is being inserted, the INIT_TEXT_HANDLE function creates a partially initialized handle that contains the language and format values you specify. The insert trigger then fills in the other values in the handle.

In the following example, INIT_TEXT_HANDLE presets the language and format in an initialized handle. The INSERT command places this handle in the COMMENTHANDLE column.

```
db2=>INSERT INTO DB2TX.SAMPLE (DOCID, COMMENT, COMMENTHANDLE)
          VALUES ('doc 101',
                  'I have installed...',
                  DB2TX.INIT_TEXT_HANDLE('AMI', 'GERMAN') )
```

The value returned by INIT_TEXT_HANDLE is type DB2TEXTH, or DB2TEXTFH.

Extracting information from handles

```
►►──┬─CCSID────┬──(─handle─)──────────────────────────►◄
     ├─FORMAT───┤
     ├─LANGUAGE─┤
     └─FILE─────┘
```
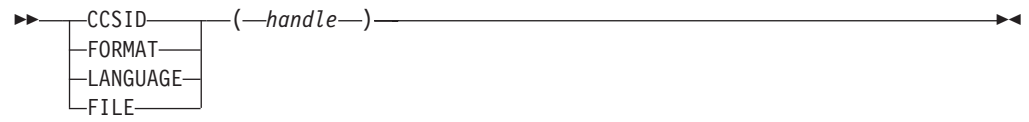
Setting and extracting information in handles

Here is an example of extracting a CCSID from a handle:

```
db2=>SELECT DISTINCT DB2TX.CCSID(COMMENTHANDLE)
          FROM DB2TX.SAMPLE
```

In the same way, you can extract the format or the language of a text document, or the location of external files. Here is an example that illustrates the use of the FORMAT function. It returns the number of ASCII (TDS) documents in the sample table.

```
db2=>SELECT COUNT(*)
          FROM DB2TX.SAMPLE
          WHERE DB2TX.FORMAT(COMMENTHANDLE) = 'TDS'
```

Changing information in handles

```
►►─┬─FORMAT──(─handle─,─format─)──────────┬──────────────────────►◄
   ├─LANGUAGE──(─handle─,─language─)───────┤
   └─FILE──(─handle─,─file-path-and-name─)─┘
```
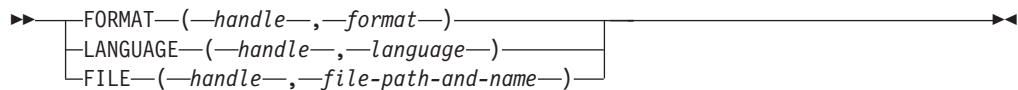
The FORMAT and LANGUAGE functions can also change the corresponding specification in a handle. These functions return the changed handle as a value of type DB2TEXTH, or DB2TEXTFH.

The following example shows how to change the language setting of a text document.

```
db2=>UPDATE DB2TX.SAMPLE
          SET COMMENTHANDLE = DB2TX.LANGUAGE(COMMENTHANDLE, 'FRENCH')
          WHERE ...
```

Using the LANGUAGE function again, you can see that the change has occurred:

```
db2=>SELECT DISTINCT DB2TX.LANGUAGE(COMMENTHANDLE)
          FROM DB2TX.SAMPLE
```

Improving search performance

The SEARCH_RESULT function exploits the DB2 concept of table-valued functions. The function is used in the FROM clause of an SQL statement, and returns an intermediate table with the search result of the specified search string. The syntax of the search string is the same as described in "Chapter 12. Syntax of search arguments" on page 187. The advantage of this function compared with CONTAINS or RANK is a significant performance improvement when large tables are involved.

Improving search performance

The returned table has the following structure:

```
Column Name          Data type
          HANDLE                DB2TX.DB2TEXTH, DB2TX.DB2TEXTFH, DB2TX.DB2TEXTSH

NUMBER_OF_MATCHES    INTEGER
RANK                 DOUBLE
```

Example:

```
db2=>SET CURRENT FUNCTION PATH = db2tx

db2=>SELECT author
     FROM db2tx.sample T1,
     table(DB2TX.SEARCH_RESULT('DB2TX','SAMPLE','MYDOCHANDLE','"compress"')) T2
     where T1.handle = T2.mydochandle
```

If you have not set the current path, the last line of the example is:

```
where  cast(T1.handle as varchar(60)) = cast(T2.mydochandle as varchar (60))
```

SELECT NUMBER_OF_MATCHES, RANK, HANDLE causes all three items to be returned, but you can specify them in any combination. You may wish, for example, to omit RANK to avoid the intensive processing associated with it.

If you need only the HANDLE value, you can simply use SELECT COUNT(*).

Chapter 7. Administration

This chapter describes how to maintain text indexes, how to get useful information, and how to reverse the text preparation process.

Maintaining text indexes

These are the index maintenance tasks:

- Updating an index
- Changing the settings of an index
- Resetting the status of an index
- Deleting index events
- Reorganizing an index.

You can run these tasks at any time and in any sequence.

Updating an index

> **Summary**
>
> **When** When an index must be updated immediately without waiting for periodic indexing to occur. (See "Enabling a text column" on page 54 for information about periodic indexing.)
>
> **Command**
> UPDATE INDEX
>
> **Authorization**
> ALTER, SELECT, UPDATE on the table

This example updates the index for a common-index table:

```
db2tx UPDATE INDEX db2tx.mytable
```

This example updates the index for a column of a multi-index table:

```
db2tx UPDATE INDEX db2tx.mytable HANDLE commenthandle
```

Use this command to update the index immediately, without waiting for the next periodic indexing to take place automatically. This is useful when you have added several text documents to a database and want to search them immediately.

DB2 Text Extender indexes the text documents in this column (or all columns in the table) that have been inserted or changed, and removes from the index the terms from documents that have been deleted. The log table associated with the index contains information about which documents have been inserted, updated, and deleted.

Updating an index for external files

A log table does not automatically contain information about changes to any external files that you may have indexed (see "Enabling external text files" on page 59), such as replacing a document by a newer version having the same

absolute path name. Updates occurring on such files cannot be monitored by DB2 Text Extender in log tables because the updates do not occur within the scope of DB2 UDB .

To have updates on external files reflected in a DB2 Text Extender index, you can do the following:

1. Force a "change" entry to be placed in the log table by issuing an update statement on the corresponding handle column that effectively does nothing:

```
UPDATE table
SET    filehandlecol = filehandlecol
WHERE  DB2TX.FILE(filehandlecol) = filename
```

where *filename* is the absolute path name of the external file that was updated.

2. Run UPDATE INDEX to bring the index up to date, including the change made to the external file.

Changing the settings of an index

> **Summary**
>
> **When**   When the update frequency of an index has to be changed.
>
> **Command**
>   CHANGE INDEX SETTINGS
>
> **Authorization**
>   ALTER, SELECT, UPDATE on the table

Use this command to change the update frequency of an index.

Update frequency
See "Updating an index" on page 36 for more information. If you do not specify an update frequency, the current settings are left unchanged.

Use the UPDATEINDEX keyword to determine whether the indexing of the text documents begins immediately, or when periodic indexing is next scheduled. If you do not use this keyword, the current setting is left unchanged.

Examples
To change the update frequency for the index so that it is updated at 12:00 or 15:00, on Monday to Friday, if there is a minimum of 100 text documents queued:

```
db2tx CHANGE INDEX SETTINGS db2tx.mytable
               HANDLE     commenthandle
               UPDATEFREQ min(100) d(1,2,3,4,5) h(12,15) m(00)
```

To stop the periodic updating of an index:

```
db2tx CHANGE INDEX SETTINGS db2tx.mytable
               HANDLE     commenthandle
               UPDATEFREQ none
```

Resetting the index status

```
┌─ Summary ──────────────────────────────────────────────────────────┐
│                                                                     │
│  When   When an index can no longer be searched or updated.         │
│                                                                     │
│  Command                                                            │
│         RESET INDEX STATUS                                          │
│                                                                     │
│  Authorization                                                      │
│         None                                                        │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

Some situations can occur that prevent you from searching in an index, or from updating it. "Displaying the status of an index" on page 88 describes how to determine if one of these events has occurred. RESET INDEX STATUS reactivates the index so that you can use it again.

This example resets the index status for the index of a common-index table:

```
db2tx RESET INDEX STATUS db2tx.mytable
```

The syntax lets you reset the index status for a particular text column. This example resets the index status for the index of a multi-index table column:

```
db2tx RESET INDEX STATUS db2tx.mytable HANDLE commenthandle
```

Deleting index events

```
┌─ Summary ──────────────────────────────────────────────────────────┐
│                                                                     │
│  When   When you no longer need the messages in an index's log table.│
│                                                                     │
│  Command                                                            │
│         DELETE INDEX EVENTS                                         │
│                                                                     │
│  Authorization                                                      │
│         None                                                        │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

If something prevents you from searching in an index, or from updating it, or if a document cannot be indexed, this is known as an indexing *event*. Information about indexing events is stored in the index's log table. It can help you determine the cause of the problem. When you no longer need these messages, you can delete them.

This example deletes messages from the index of a common-index table:

```
db2tx DELETE INDEX EVENTS db2tx.mytable
```

The syntax also lets you delete indexing events for a particular text column. This example deletes the messages for the index of a multi-index table column:

```
db2tx DELETE INDEX EVENTS db2tx.mytable HANDLE commenthandle
```

Reorganizing an index

┌─ **Summary** ──┐

When   When GET INDEX STATUS indicates that an index should be
         manually reorganized.

Command
         REORGANIZE INDEX

Authorization
         None
└──┘

If a text column is often updated, searching the index can become inefficient. To make searching efficient again, the index has to be *reorganized*. Although DB2 Text Extender recognizes when an index needs to be reorganized and does so in the background automatically, there may be situations that require an index to be reorganized manually using REORGANIZE INDEX. You can use the command GET INDEX STATUS to find out if an index needs to be reorganized.

Although searches can be made on the index while REORGANIZE INDEX is running, index updates cannot.

This example reorganizes the index of a common-index table:

```
db2tx REORGANIZE INDEX db2tx.mytable
```

This example reorganizes the index of a multi-index table column:

```
db2tx REORGANIZE INDEX db2tx.mytable HANDLE commenthandle
```

Getting useful information

This section describes the commands for displaying information about:
- The enabled status of databases, tables, columns, and files
- The settings of the environment variables
- The text configuration settings
- The index status
- The error events
- The index settings
- The text settings for a column.

Displaying enabled-status information

┌─ **Summary** ──┐

When   When you need information about the enabled status of databases,
         tables, text columns or external files.

Command
         GET STATUS

Authorization
         None
└──┘

Enter:

```
db2tx GET STATUS
```

Here is an example of the output displayed by GET STATUS. It shows the enabled
status of the database, and of any enabled tables, text columns, or text files that it
contains.

```
Database is enabled for Text Extender

  Table DB2TX.MYTABLE is enabled as a common-index table

  Table DB2TX.MYTABLE is enabled as a common-index table
     TextColumnName      HandleColumnName
     --------------      ----------------
     COMMENT             COMMENTHANDLE


  Table DB2TX.TEST is enabled as a multi-index table
     TextColumnName      HandleColumnName
     --------------      ----------------
     ABSTRACT1           ABSTRACT1HANDLE
     ABSTRACT2           ABSTRACT2HANDLE
```

Displaying the settings of the environment variables

> **Summary**
>
> **When** When you need information about the settings of the environment
> variables.
>
> **Command**
> GET ENVIRONMENT
>
> **Authorization**
> None

Enter:

```
db2tx GET ENVIRONMENT
```

Here is an example of the output displayed by GET ENVIRONMENT. It shows the
current settings of the DB2 Text Extender environment variables.

```
Current database manager instance  (DB2INSTANCE) = db2
Default database name                  (DB2DBDFT) = my_database
Instance name                  (DB2TX_INSTOWNER) = user1
Instance directory      (DB2TX_INSTOWNERHOMEDIR) = /usr/instance1
```

Displaying the text configuration settings

> **Summary**
>
> **When** When you need the default settings for text, index, and process
> information.
>
> **Command**
> GET TEXT CONFIGURATION
>
> **Authorization**
> None

These settings are described in "Text configuration settings" on page 14. To change them, see "Changing the text configuration" on page 15.

To display the text configuration, enter:

```
db2tx GET TEXT CFG
```

Here is an example of the output displayed by GET TEXT CONFIGURATION. It shows the current text configuration settings.

```
Coded character set ID    (CCSID) = 850
Language               (LANGUAGE) = US_ENGLISH
Format                   (FORMAT) = TDS

Index type            (INDEXTYPE) = LINGUISTIC
Update frequency     (UPDATEFREQ) = NONE
Index directory       (DIRECTORY) = user1/db2tx/indexes

Update index option (UPDATEINDEX) = UPDATE
Commit count        (COMMITCOUNT) = 10 000
Tablespace name      (TABLESPACE) = TXLOG
```

Displaying the status of an index

> **Summary**
>
> **When**   When you need to determine whether an index can be searched or
>   updated.
>
> **Command**
>   GET INDEX STATUS
>
> **Authorization**
>   None

Some situations can occur that prevent you from searching in an index, or from updating it. In such situations, messages are stored in the index's log table that can help you determine the cause. So it can be useful to check the status of an index, and whether there are any messages available.

This example displays the index status for the index of a common-index table:

```
db2tx GET INDEX STATUS db2tx.mytable
```

The syntax lets you display the index status for a particular text column. This example gets the index status for the index of a multi-index table column:

```
db2tx GET INDEX STATUS db2tx.mytable HANDLE commenthandle
```

Here is an example of the output displayed by GET INDEX STATUS.

```
Node 1
Search status            = Search available
Update status            = Update available
Reorg status             = started 13.55
Scheduled documents      = 0
Indexed documents        = 187000
Primary index documents  = 130000
Secondary index documents =  57000
Error events             = No error events
```

If the index is split among several nodes, the node information is displayed per node.

Search status

Indicates whether you can use the specified handle column to search in the index. If search is not available, check the indicated reason code for more information about why the situation occurred, and then use RESET INDEX STATUS to be able to work with the index again. See "Chapter 20. Error event reason codes" on page 271.

Update status

Indicates whether you can update the index for the specified table or column. If the index update function is not available, check the indicated reason code for more information about why the situation occurred, and then use RESET INDEX STATUS to be able to work with the index again.

Reorg status

Indicates whether you can reorganize the index for the specified table or column. If the reorganize function is not available, check the indicated reason code for more information about why the situation occurred. A common reason for reorganization not being available is because the index is currently being updated.

Scheduled documents

Shows the number of documents that are listed in the queue for indexing (or for deleting from the index).

Indexed documents

Shows the number of documents that have already been indexed from the queue of scheduled documents.

Primary index documents

Shows the number of documents in the primary index.

Secondary index documents

Shows the number of documents in the secondary index.

Error events

Shows the number of events that are available in the index's log table. You can display this information as described in "Displaying error events". When you no longer need this information, you can delete it as described in "Deleting index events" on page 85.

Displaying error events

When problems occur during indexing, such as a document scheduled for indexing could not be found, these so-called *error events* are written to the index's log table.

The event return codes are described in "Chapter 20. Error event reason codes" on page 271.

You can access the error events in a view of the log table called `db2tx.LOGIXnnnnnn`, where *IXnnnnnn* is the name of the index, obtainable from the catalog view.

To get the name of the index:

```
DB2 SELECT TABLENAME,
           HANDLENAME,
           INDEXNAME
      FROM  DB2TX.TEXTCOLUMNS
```

The error event view has the following layout:

Getting useful information

```
UPDATESTATUS    SMALLINT
EVENTREASON     INTEGER
EVENTMESSAGE    VARCHAR(1024)
UPDATETIME      TIMESTAMP
HANDLE          DB2TEXTH or DB2TEXTFH
NODENUMBER      INTEGER
```

Here is an example showing how to access the information from the index log:

```
DB2 SELECT  EVENTREASON,
            EVENTMESSAGE,
            UPDATETIME,
            HANDLE
     FROM   DB2TX.LOGIXNNNNNN
```

Displaying the index settings

Summary

When When you need information about the settings of an index.

Command
GET INDEX SETTINGS

Authorization
None

This example gets the index settings for the index of a common-index table:

```
db2tx GET INDEX SETTINGS db2tx.mytable
```

This example gets the index settings for the index of a multi-index table column:

```
db2tx GET INDEX SETTINGS db2tx.mytable HANDLE commenthandle
```

If the table is enabled as a multi-index table, this command displays the index settings of all enabled text columns in the table.

Here is an example of the output displayed by GET INDEX SETTINGS for a common-index table. The output for a multi-index table shows similar information for each index. The syntax lets you request the index settings for a particular text column.

```
Current index settings:

    Index type            (INDEXTYPE)   = LINGUISTIC
    Update index option   (UPDATEINDEX) = UPDATE
    Update frequency      (UPDATEFREQ)  = NONE
    Node 1
    Index directory       (DIRECTORY)   = /home/user1/db2tx/indices
```

If the index is split among several nodes, the node information is displayed for the index directory.

Displaying the text settings for a column

Summary

When When you need information about the text settings for a column.

Command
GET TEXT INFO

Authorization
None

This example gets the text information for the index of a common-index table:

Getting useful information

```
db2tx GET TEXT INFO db2tx.mytable
```

This example gets the text information for the index of a multi-index table column:

```
db2tx GET TEXT INFO db2tx.mytable HANDLE commenthandle
```

The syntax lets you specify a table name and the name of a handle column.

If you specify only a table name in the command, the text information for each enabled column in this table is displayed. If you also specify a handle column name, only the text information for that column is displayed.

Here is an example of what is displayed by this command for a multi-index table:

```
        Text information for column ABSTRACT1
              with handle column ABSTRACT1HANDLE:
        Coded character set ID  (CCSID) = 850
        Language             (LANGUAGE) = US_ENGLISH
        Format                 (FORMAT) = TDS

        Text information for column ABSTRACT2
              with handle column ABSTRACT2HANDLE:
        Coded character set ID  (CCSID) = 850
        Language             (LANGUAGE) = US_ENGLISH
        Format                 (FORMAT) = TDS
```

Working with the DB2 Text Extender catalog view

DB2 Text Extender creates and maintains a catalog view called DB2TX.TEXTINDEXES for each database. It is created when you run ENABLE DATABASE. It contains information about the tables and columns that are enabled for DB2 Text Extender.

New entries are created in DB2TX.TEXTINDEXES whenever a table, a column, or external files are enabled. Entries are updated whenever index settings are modified using the CHANGE INDEX SETTINGS command. Entries are deleted if columns or tables are disabled.

Data in the catalog view is available through normal SQL query facilities. However, you cannot modify the catalog view using normal SQL data manipulation commands. You cannot explicitly create or drop the catalog view. Table 7 shows the contents of the catalog view.

Table 7. DB2 Text Extender catalog view

Column name	Data type	Nullable	Description
TABLESCHEMA	CHAR(8)	No	Schema of the table to which this entry applies.
TABLENAME	VARCHAR(18)	No	Name of the table to which this entry applies.
COLUMNNAME	VARCHAR(18)	Yes	Name of a column that has been enabled within this table. This value is null if the table has been enabled, but no column has been enabled.
HANDLENAME	VARCHAR(18)	Yes	Name of a handle column. This value is null if there is no column enabled in the table TABLESCHEMA.TABLENAME.
INDEXNAME	CHAR(8)	No	Name of the text index created during enabling of the text table or a text column.

Table 7. DB2 Text Extender catalog view (continued)

Column name	Data type	Nullable	Description
LOGTABLE	VARCHAR(18)	No	Name of the log table for the index INDEXNAME. The table DB2TX.LOGTABLE contains information about which text documents are scheduled for the next update of the text index, and error events.
INDEXTYPE	VARCHAR(30)	No	Type of index: LINGUISTIC, PRECISE, NGRAM.
MINIMUM	INTEGER	Yes	The smallest number of index update requests required before an index update is performed. See "Updating an index" on page 36. This value is null if the update frequency is set to NONE.
DAYS	VARCHAR(15)	Yes	The days when an update is to be scheduled. See "Updating an index" on page 36. This value is null if the update frequency is set to NONE.
HOURS	VARCHAR(75)	Yes	The hours when an index update is to be scheduled. See "Updating an index" on page 36. This value is null if the update frequency is set to NONE.
MINUTES	VARCHAR(185)	Yes	The minutes when an update is scheduled. See "Updating an index" on page 36. This value is null if the update frequency is set to NONE.
INDEXDIRECTORY	VARCHAR(254)	No	Name of the directory where the text index is stored within the file system.
UPDATEONCREATE	VARCHAR(10)	No	The value "update" or "noupdate", whatever has been specified with the UPDATEINDEX option in ENABLE TEXT TABLE or ENABLE TEXT COLUMN, or in the last CHANGE INDEX SETTINGS.
COMMONINDEX	VARCHAR(4)	No	"yes" if the table TABLESCHEMA.TABLENAME is a common-index table. "no" if the table TABLESCHEMA.TABLENAME is a multi-index table.
CCSID	SMALLINT	Yes	CCSID for the text column TEXTCOLUMN specified with the enable text column command. This value is null if TEXTCOLUMN is null.
LANGUAGE	VARCHAR(30)	Yes	The name of the dictionary used when processing text column TEXTCOLUMN. This value is null if TEXTCOLUMN is null.
FORMAT	VARCHAR(30)	Yes	The format specified for text column TEXTCOLUMN. This value is null if TEXTCOLUMN is null.
FUNCTIONSCHEMA	CHAR(8)	Yes	Schema of the access function specified in the ENABLE TEXT COLUMN command using the FUNCTION option. This value is null if no FUNCTION option is specified.
FUNCTIONNAME	VARCHAR(18)	Yes	Name of the access function specified in the ENABLE TEXT COLUMN command using the FUNCTION option. This value is null if no FUNCTION option is specified.
PROTOTYPEHANDLE	VARCHAR(60)	Yes	A handle for use in performance functions. It contains only the index name which is common for the whole text column.
INDEXOPTION	VARCHAR(30)	Yes	Option used when creating the index: CASE_ENABLED.
INDEXPROPERTY	VARCHAR(30)	Yes	Property used when creating the index: SECTIONS_ENABLED.

Table 7. DB2 Text Extender catalog view (continued)

Column name	Data type	Nullable	Description
NODENUMBER	INTEGER	No	Node number of the table partition.

Reversing the text preparation process

When text is prepared for use by DB2 Text Extender, certain administrative changes are made. This section describes functions that help you to reverse this process.

Disabling a text column

Summary

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

Command
        DISABLE TEXT COLUMN

Authorization
        ALTER, SELECT, UPDATE on the table.

Example:

```
db2tx DISABLE TEXT COLUMN db2tx.mytable HANDLE commenthandle
```

When you disable a text column, the following occurs:

* If this is a multi-index table, that is, the column has its own text index and log table, then the index, the log table, and the log table triggers are deleted.
* If this is a common-index table, that is, there is one index shared by all text columns, then the terms for this column's documents are removed from the common index. If this is the only remaining enabled text column in the table, then the index, the log table, and the log table triggers are deleted.

Disabling text files

Summary

When   When you no longer intend to make text searches in a set of external text files.

Command
        DISABLE TEXT FILES

Authorization
        ALTER, SELECT, UPDATE on the table.

Example:

```
db2tx DISABLE TEXT FILES db2tx.mytable HANDLE commenthandle
```

When you disable external text files, the following occurs:

* The index for this handle column is deleted.

- The log table and triggers are deleted.

Disabling a text table

Summary

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

Command
     DISABLE TEXT TABLE

Authorization
     ALTER, SELECT, UPDATE on the table.

Example:

```
db2tx DISABLE TEXT TABLE db2tx.mytable
```

When you disable a text table, the following occurs:

- If there is a common index for the text columns of the table, this index is deleted. If, instead, there are individual indexes for each text column, *all* the indexes for the text columns are deleted.
- The common log table used to automatically record which text documents are to be indexed is deleted. If, instead, there are individual log tables for each text column, all the log tables are deleted.
- The triggers used to maintain the log tables are deleted.
- The content of the handle columns is set to null.

Disabling a database

Summary

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

Command
     DISABLE DATABASE

Authorization
     SYSADM or DBADM on the database.

To disable the connected database, enter:

```
db2tx DISABLE DATABASE
```

When you disable a database, the following objects are deleted:

- The DB2 Text Extender catalog view that was created when the database was enabled
- The declaration of DB2 Text Extender's SQL functions (UDFs), and DB2 Text Extender distinct types (UDTs) for this database
- All indexes related to any of this database's text tables or text columns
- The log tables used to automatically record which text documents are to be indexed, and the triggers used to maintain them.

Reversing the text preparation process

Because handle columns cannot be deleted, and the handle column is of a distinct type, some distinct types are not deleted.

Chapter 8. Using the API functions for searching and browsing

This chapter describes how to use the search and browse functions of the DB2 Text Extender API. For a detailed description of these functions, refer to "Chapter 13. API functions for searching and browsing" on page 199. Examples of programs that use the API functions are given in "Chapter 14. Sample API program" on page 219. The same chapter describes a sample browse function DesBrowseDocument.

> **Tip**
>
> Before searching, you should read "Types of search" on page 32. A search can produce different results depending on which index type is used. Use GET INDEX SETTINGS to find out which text index type is associated with the text you are searching in.

Setting up your application

An application program that uses the DB2 Text Extender API is a DB2 CLI application, because some of the API functions require a database connection handle as input. So the rules that have to be considered for DB2 CLI applications apply also to applications that use the DB2 Text Extender API.

In your application, include `des_ext.h` which is provided in the `include` subdirectory of the DB2 Text Extender installation directory

To use your application program with the DB2 Text Extender API, link your program to the API.

Linking a UNIX application

You must link the library `libdescl.a` to your application. This library is in the `lib` subdirectory of the DB2 Text Extender installation directory.

Linking an OS/2 or a Windows application

All DB2 Text Extender API functions are contained in the dynamic link library `desclapi.dll`. The external API function calls are resolved in the import library `desclapi.lib`. These are provided in the `dll` and `lib` subdirectories, respectively, of the DB2 Text Extender installation directory.

To link your application with the API dynamic link library `desclapi.dll`, use `desclapi.lib` as library file.

Overview of the API functions

These are the search and browse functions; the first is a search function, the remainder are browse functions:
    DesGetSearchResultTable
    DesGetBrowseInfo
    DesStartBrowseSession
    DesOpenDocument

Overview of the API functions

DesGetMatches
DesCloseDocument
DesEndBrowseSession
DesFreeBrowseInfo.

> **Tip**
>
> Many of the API functions need a connection handle (hdbc). You must provide this handle using the SQLConnect function, but this does not prevent you from calling DB2 Text Extender from embedded SQL programs. The *DB2 Call Level Interface Guide and Reference* describes how to mix CLI statements with embedded SQL statements.

You can use the API functions for:

* **Searching for text**

  In this scenario, only the search function DesGetSearchResultTable is needed. It takes as input a search argument and a handle column name. It searches for text and puts information about the documents found into a result table that you have prepared previously.

  This function is described in more detail in "Searching for text" on page 99. See also "Chapter 14. Sample API program" on page 219.

* **Browsing text**

  Use the following functions in the sequence shown:

```
┌─────────────────────────┐   ┌─────────────────────────────┐
│ 1A. Get Browse Information│   │ 1B. Get Search Result Table │
└─────────────────────────┘   └─────────────────────────────┘
              │                              │
              ▼                              │
┌─────────────────────────┐                  │
│ 2. Start Browse Session  │◄─────────────┐  │
└─────────────────────────┘              │  │
              │                          │  │
              ▼                          │  │
┌─────────────────────────┐              │  │
│ 3. Open Document         │◄──────────┐ │  │
└─────────────────────────┘           │ │  │
              │                        │ │  │
              ▼                        │ │  │
┌─────────────────────────┐           │ │  │
│ 4. Get Matches           │───────────┘ │  │
└─────────────────────────┘              │  │
              │                          │  │
              ▼                          │  │
┌─────────────────────────┐              │  │
│ 5. Close Document        │──────────────┘  │
└─────────────────────────┘                  │
              │                              │
              ▼                              │
┌─────────────────────────┐                  │
│ 6. End Browse Session    │──────────────────┘
└─────────────────────────┘
              │
              ▼
┌─────────────────────────┐
│ 7. Free Browse Information│
└─────────────────────────┘
```
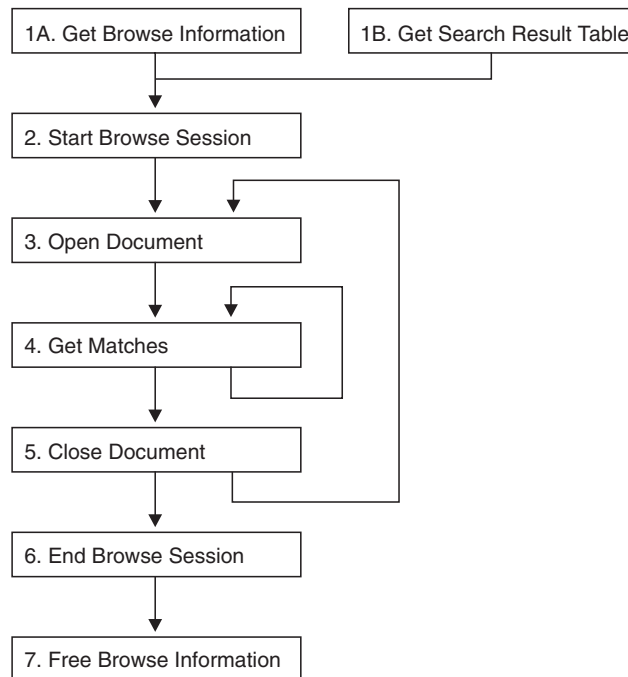
Figure 16. Sequence for using the API functions

These functions get highlighting information, then start a browse session to display a text document and highlight the found terms.

In a browse session, you can open and display further documents using the same highlighting information. These functions are described in more detail in "Browsing text".

Searching for text

There is one API function for searching for text: the DesGetSearchResultTable function.

Get a search result table (DesGetSearchResultTable)

The DesGetSearchResultTable function receives a search argument for searching through text documents in a particular text column, and stores the result in a table. The result table contains the handle of each document found. It can also contain rank information and the number of matches, depending on the search option specified.

You can also obtain this information using the RANK and NO_OF_MATCHES functions. Here is an example:

```
INSERT INTO RESULT
    SELECT COMMENTHANDLE,
           RANK(COMMENTHANDLE, '"stored procedures"'),
           NO_OF_MATCHES(COMMENTHANDLE, '"stored procedures"')
    FROM DB2TX.SAMPLE
    WHERE  CONTAINS(COMMENTHANDLE, '"stored procedures"') = 1
```

DesGetSearchResultTable can be used only on base tables, but it can be faster than using DB2 Text Extender functions if the query is a text-only query; it goes directly to the DB2 Text Extender server to get the rank and the number of matches, and it loops only for the number of matching documents found. In the UDF example, on the other hand, the CONTAINS function is called once for each row in the table; then, for each qualifying row, the RANK and NO_OF_MATCHES functions are called. For each found document, three separate searches are made.

Input
The input parameters are:
- The handle for the database connection
- The table to be searched
- The name of the handle column that is associated with the text column to be searched
- A search argument
- Search options
- A browse option (to return browse information)
- The name of the table where the result is to be stored.

Output
If a browse option is specified, this function returns a pointer to browse information.

Browsing text

This group of functions in Figure 16 on page 98 finds out which terms are to be highlighted. It then starts a browse session, opens a document, and gets match information in the form of a data stream that can be parsed by an application program that calls your browser.

Get browse information (DesGetBrowseInfo)

The DesGetBrowseInfo function receives a search argument and a handle. It returns a pointer to the browse information needed by DesStartBrowseSession. Browse information includes a list of all the terms to be highlighted.

Another method of getting browse information is to specify the Browse option in the function DesGetSearchResultTable.

Input
The input parameters are:

The handle for database connection

A handle

A search argument.

Output
This function returns a pointer to browse information.

Start a browse session (DesStartBrowseSession)

The DesStartBrowseSession function starts a browse session, establishing the environment needed for browsing a text document and highlighting its matches. It receives a pointer to browse information, either from DesGetBrowseInfo or from DesGetSearchResultTable, and returns a browse session handle for use by the other browse functions.

Input
The input parameter is:

A pointer to browse information from DesGetBrowseInfo or DesGetSearchResultTable

A user ID

A password.

Output
This function returns a browse session handle.

Open a document (DesOpenDocument)

The DesOpenDocument function receives a browse session pointer, a handle, and an option DES_FAST or DES_EXTENDED indicating the type of linguistic processing to be used for highlighting found terms. See "Stage 2: Extended matching" on page 230. DES_FAST uses basic text analysis, without the use of a dictionary, to determine which terms are to be highlighted. DES_EXTENDED uses extended matching.

DesOpenDocument prepares the text document that corresponds to the handle to get the document text and highlighting information, and it returns a document handle that is used for iteratively calling DesGetMatches.

Input
The input parameters are:

A browse session handle from DesStartBrowseSession

A text handle

A match option: DES_FAST or DES_EXTENDED.

Output

This function returns a document handle which is used by DesGetMatches and DesCloseDocument.

Get matches (DesGetMatches)

The DesGetMatches function returns a pointer to highlighting information for the text document described by a document handle. The highlighting information is a data stream. It comprises the text context (at least one paragraph) and information for highlighting text in that context. The data stream is described in "Data stream syntax" on page 206. An application program can parse the data stream and process it using the user's own browser.

DesGetMatches returns only a portion of the data stream, indicating the length of the portion in the output structure.

A sequence of calls to DesGetMatches gets the entire text document content. When the end of the text document is reached, an indicator is returned.

Input

The input parameters are:

A browse session handle

A document handle from DesOpenDocument.

Output

This function returns a pointer to a structure containing the data stream portion and its length.

Close a document (DesCloseDocument)

The DesCloseDocument function closes a text document opened by DesOpenDocument, and releases the storage allocated during the return of document text and highlighting information.

Input

The input parameters are:

A browse session handle

A document handle from DesOpenDocument.

Output

None.

End a browse session (DesEndBrowseSession)

The DesEndBrowseSession function ends a browse session started by DesStartBrowseSession, and releases the storage allocated for the browse session.

Input

The input parameter is:

A browse session handle.

Output

None.

Free the browse information (DesFreeBrowseInfo)

The DesFreeBrowseInfo function frees storage allocated for the browse information by DesGetBrowseInfo.

Browsing text

Input

The input parameter is:

A pointer to the browse information.

Output

None.

Part 2. Reference

Chapter 9. Text preparation and administration commands for the client

Command	Purpose	Page
CHANGE INDEX SETTINGS	Changes the characteristics of an index	107
CHANGE TEXT CONFIGURATION	Changes the text configuration settings	109
CONNECT	Connects you to a database	112
DELETE INDEX EVENTS	Deletes index events from a log table	113
DISABLE DATABASE	Disables a database from use by DB2 Text Extender	114
DISABLE TEXT COLUMN	Disables a text column from use by DB2 Text Extender, and deletes its associated index	115
DISABLE TEXT FILES	Disables text files from use by DB2 Text Extender, and deletes their associated index	116
DISABLE TEXT TABLE	Disables a table from use by DB2 Text Extender and deletes the indexes associated with the table	117
ENABLE DATABASE	Prepares a database for use by DB2 Text Extender	118
ENABLE TEXT COLUMN	Prepares a text column for use by DB2 Text Extender and creates an individual text index for the column	119
ENABLE TEXT FILES	Prepares text files for use by DB2 Text Extender and creates an individual text index for the files	126
ENABLE TEXT TABLE	Creates a common text index for a table	129
GET ENVIRONMENT	Displays the current settings of the environment variables	133
GET INDEX SETTINGS	Displays the characteristics of an index	134
GET INDEX STATUS	Displays status information for an index	136
GET STATUS	Displays the enabled status of databases, tables, and columns	137
GET TEXT CONFIGURATION	Displays the text configuration settings	138
GET TEXT INFO	Displays the text information for a text column	139
QUIT	Exits from the DB2 Text Extender command line processor mode	140
REORGANIZE INDEX	Reorganizes an index to improve search efficiency	141
RESET INDEX STATUS	Resets the status of an index to allow it to be used again	142
UPDATE INDEX	Updates a text index	143

This chapter describes the syntax of the text preparation and administration commands for the client. "Chapter 5. Making text searchable" on page 47 and "Chapter 7. Administration" on page 83 describe how to use these commands.

Before you use these commands, start the DB2 Text Extender command line processor by entering the command db2tx. It puts you into an interactive input mode in which all subsequent commands are interpreted as DB2 Text Extender commands. Normally, you would start the command processor at the same time as you start DB2.

To leave this mode, enter QUIT.

As in DB2, you can issue commands directly from the operating system prompt by prefixing them with db2tx, and you can include command line processor requests in a shell script command file.

> **Tip**
>
> A command prefixed with db2tx causes a connection to be made to the default database specified in the environment variable DB2DBDFT. The following sequence of commands does not enable the database MYDATABASE, but instead enable the default database.
>
> ```
>     db2tx CONNECT TO MYDATABASE
>     db2tx ENABLE DATABASE
> ```

CHANGE INDEX SETTINGS

This command changes the characteristics of an index **after** the database has been enabled.

To changes the default settings that are used when a database is first enabled, use "CHANGE TEXT CONFIGURATION" on page 109.

Authorization

You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

```
►►──CHANGE INDEX SETTINGS──table-name──────────────────────────────────────────────►
                              └─HANDLE──handle-column-name─┘


►──┬───────────────────────────────────────────┬──────────────────────────────────►◄
   └─UPDATEFREQ──┬─NONE─────────────────┬─┘
                 └─ update-frequency ─┘
```

update-frequency:

```
├──MIN──(──mindocs──)──D──(──┬──*──────────┬──)──H──(──┬──*───────────┬──)──────────►
                             │   ┌─,─┐      │          │   ┌─,─┐       │
                             └─▼─0...6─┴─┘              └─▼─0...23─┴──┘


►──M──(──┬─────────────┬──)───────────────────────────────────────────────────────┤
         │   ┌─,─┐      │
         └─▼─0...59─┴──┘
```
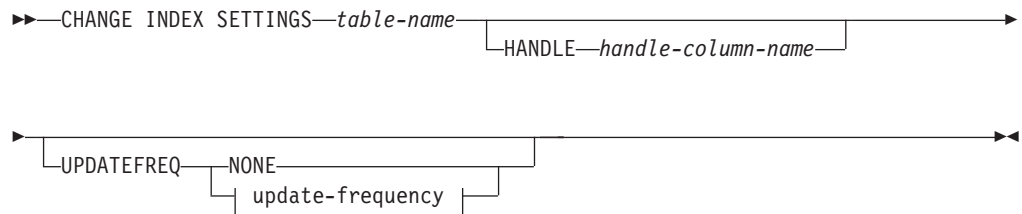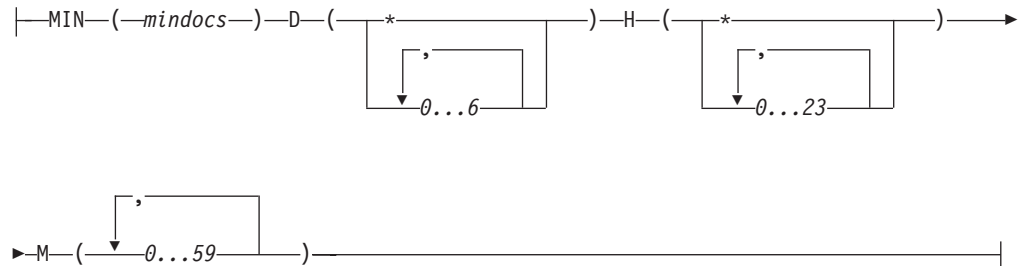
Command parameters

table-name

The name of the text table in the connected database that contains the text column whose index update frequency is to be changed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose index update frequency is to be changed. This is required if the text column has its own index, that is, if the index was created using the command ENABLE TEXT COLUMN.

If, however, the index was created using the command ENABLE TEXT TABLE, that is, the table has one text index for all text columns, then this keyword is ignored.

UPDATEFREQ update-frequency

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

The syntax is described in "Updating an index" on page 36.

NONE

No further index updates are made. This is intended for a text column in which there will be no further changes.

If you do not specify the UPDATEFREQ keyword, the frequency settings are left unchanged.

CHANGE TEXT CONFIGURATION

This command changes the default settings of the text configuration that is used when a database is enabled. These are the *text configuration* settings. The initial text configuration settings when DB2 Text Extender is installed are described in "Text configuration settings" on page 14.

Authorization

You must have:
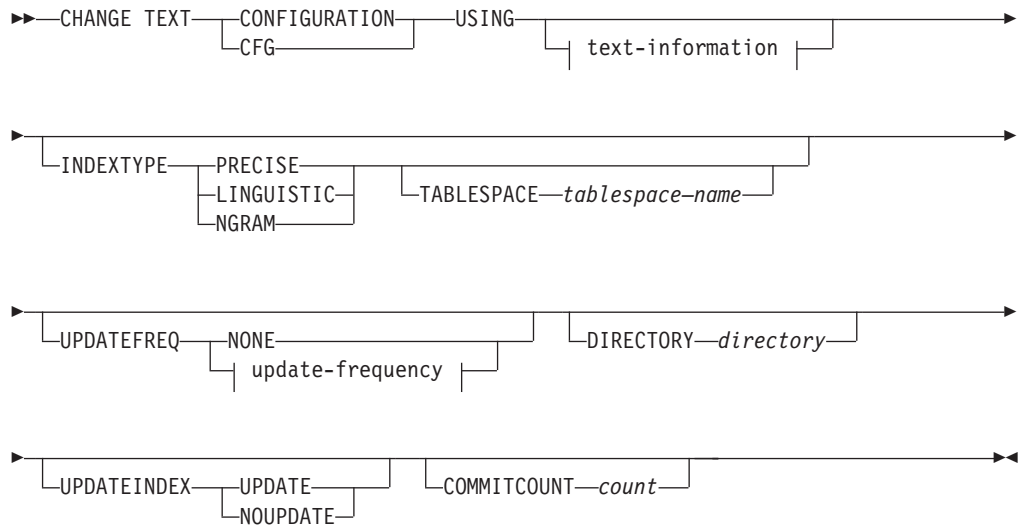
SYSADM authority

DBADM authority.

Command syntax

```
►►─CHANGE TEXT──┬─CONFIGURATION─┬──USING───────────────────────────────►
                └─CFG───────────┘        ┌─────────────────┐
                                         └─ text-information ─┘


►──┬──────────────────────────────────────────────────────────────────►
   └─INDEXTYPE──┬─PRECISE────┬──────────────────────────────┐
               ├─LINGUISTIC─┤  └─TABLESPACE──tablespace–name─┘
               └─NGRAM──────┘


►──┬─────────────────────────────────────────┬──┬──────────────────────►
   └─UPDATEFREQ──┬─NONE──────────────┬─┘       └─DIRECTORY──directory─┘
                 └─ update-frequency ─┘


►──┬──────────────────────────────────┬──┬──────────────────────┬──►◄
   └─UPDATEINDEX──┬─UPDATE───┬─┘         └─COMMITCOUNT──count─┘
                  └─NOUPDATE─┘
```

text-information:

```
├──┬──────────────┬──┬────────────────────┬──┬─────────────────┬──┤
   └─CCSID──ccsid─┘  └─LANGUAGE──language─┘  └─FORMAT──format─┘
```

update-frequency:

```
├──MIN──(──mindocs──)──D──(──┬─*──────┬──)──H──(──┬─*───────┬──)──►
                             │  ┌─,─┐ │           │  ┌─,──┐ │
                             └──▼0...6─┘           └──▼0...23┘


►──M──(──┬─────────┬──)────────────────────────────────────────────────┤
         │  ┌─,──┐ │
         └──▼0...59┘
```
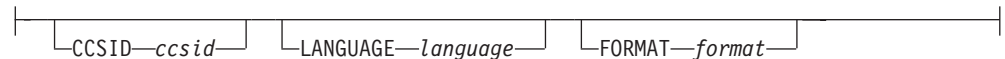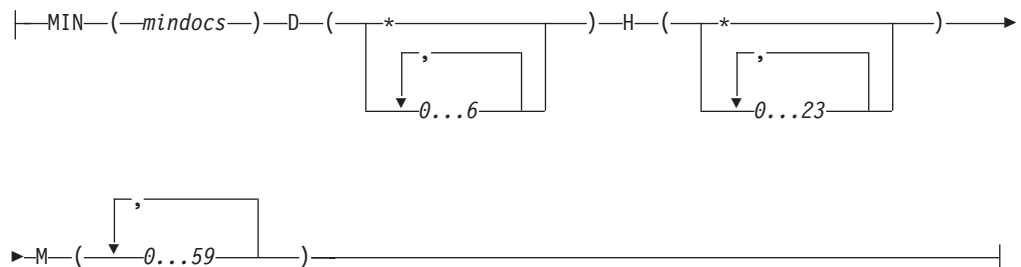
Command parameters

INDEXTYPE

To change the default index type, choose one of the following. For more information, see "Types of search" on page 32.

PRECISE

Terms are indexed and searched for exactly as they occur in the text documents.

LINGUISTIC

Terms are processed linguistically before being indexed. Search terms are also processed linguistically before the search begins.

NGRAM

Terms are indexed by parsing sets of characters rather than by using a dictionary. This dictionary type is mandatory if the documents you are indexing contain DBCS characters, although an Ngram index can also be used for SBCS documents.

TABLESPACE tablespace–name

Specify the name of an existing tablespace. The tablespace is used to hold the index-specific tables created by DB2 Text Extender (such as the log tables). For large tables, use a separate tablespace. If you do not specify a tablespace, the tables are created in the DB2 default tablespace.

UPDATEFREQ update-frequency

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

The syntax is described in "Updating an index" on page 36.

NONE

No further index updates are made. This is intended for a text column in which there will be no further changes.

DIRECTORY directory

The directory in which the text index is to be stored.

UPDATEINDEX

A keyword that determines whether the text documents are indexed immediately after the command using this option has completed, without waiting for the next periodic indexing set by UPDATEFREQ. These commands are ENABLE TEXT COLUMN, and ENABLE TEXT FILES.

UPDATE

Indexing of the text documents occurs immediately after the command has completed.

NOUPDATE

Indexing occurs at a time set by the update frequency settings specified either in this command by UPDATEFREQ, or by the text configuration setting.

COMMITCOUNT count

A value from 500 to 1 000 000 indicating the number of inserts or updates after which a DB2 UDB intermediate commit statement is issued. This can avoid a situation in which there is insufficient log space when enabling large tables, or columns, or a large number of external files.

CCSID ccsid

The Coded Character Set Identifier to be used when indexing text documents.

For information about CCSIDs that can be supported, see "CCSIDs" on page 27.

LANGUAGE language

The language in which the text is written. This determines which dictionary is to be used when indexing text documents and when searching in text documents. "Chapter 15. Linguistic processing for linguistic and precise indexes" on page 221 describes how dictionaries are used.

The supported languages are listed in Table 5 on page 43.

FORMAT format

The type of text document stored, such as WordPerfect, or ASCII. DB2 Text Extender needs this information when indexing documents. The document formats supported are listed in "Which document formats are supported" on page 24.

Usage notes

To change these settings for a particular database **after** the database has been enabled, use "CHANGE INDEX SETTINGS" on page 107.

CONNECT

This command connects DB2 Text Extender to a database.

Authorization

CONNECT authorization to the database.

Command syntax

```
►►──CONNECT──┬─────────────────────────────────────────────────────┬──►◄
             └─TO──db-alias──┬───────────────────────────────────┬──┘
                             └─USER──user-name──USING──password──┘
```
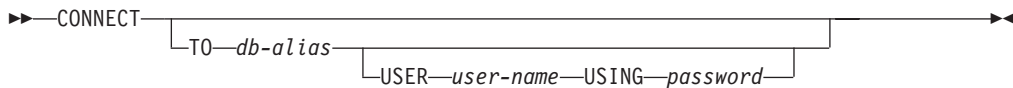
Command parameters

TO db-alias
> The database to connect to. The name must be a database alias name, not a database name.

USER user-name
> If no user name is specified, it is retrieved from the operating system.

USING password
> A password is required only if a user name is specified.

CONNECT no operand
> If you do not specify an operand and there is no connected database, CONNECT makes an implicit connection to the database specified in the environment variable DB2DBDFT. If you do not specify an operand and there is a connected database, CONNECT displays information about the current database.

Usage notes

When you issue a DB2TX command without already being connected to a database, DB2 Text Extender connects to the database specified in the environment variable DB2DBDFT. To explicitly connect to a particular database, use CONNECT TO.

You can be connected to only one database at a time; this is called the current database. In interactive mode, a connection lasts until another CONNECT TO statement changes the database, or until a QUIT command is issued. In command line mode, a CONNECT command has no effect.

If you are using an OS/2 client, and this is the first time that you are connecting to a database after you have started the system, you are prompted for a user ID and a password by User Profile Management (UPM). If you are already logged on, UPM takes the user ID and password from this session and tries to connect to the remote server. See "Starting the DB2 Text Extender command line processor" on page 48.

DELETE INDEX EVENTS

This command deletes indexing events from an index's log table for a given handle column or table.

Authorization

None.

Command syntax

```
►►──DELETE INDEX EVENTS──table-name──────────────────────────────────►◄
                              └─HANDLE──handle-column-name─┘
```

Command parameters

table-name

The name of the text table in the connected database whose error events are to be deleted from the log table. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose messages are to be deleted from the log table.

Usage notes

If a handle column name is given, the indexing events for only this column are deleted.

You clean up the log tables, you should delete index events after you have checked the reason for the event and possibly removed the source of the error.

DISABLE DATABASE

This command resets any preparation work done by DB2 Text Extender for a database and disables all text tables for use by DB2 Text Extender.

Authorization

You must have:

SYSADM authority

DBADM authority.

Command syntax

▶▶──DISABLE DATABASE───▶◀

Command parameters

None.

Usage notes

This command resets the connected database so that it can no longer be searched by DB2 Text Extender; that is, it disables all DB2 Text Extender text tables and text columns in the database. This includes all modifications that were made in the database to enable DB2 Text Extender text tables, text columns, and external files are reset: all related text indexes are deleted, the DB2 Text Extender catalog view TEXTCOLUMNS in the database is deleted, and all DB2 Text Extender triggers are deleted.
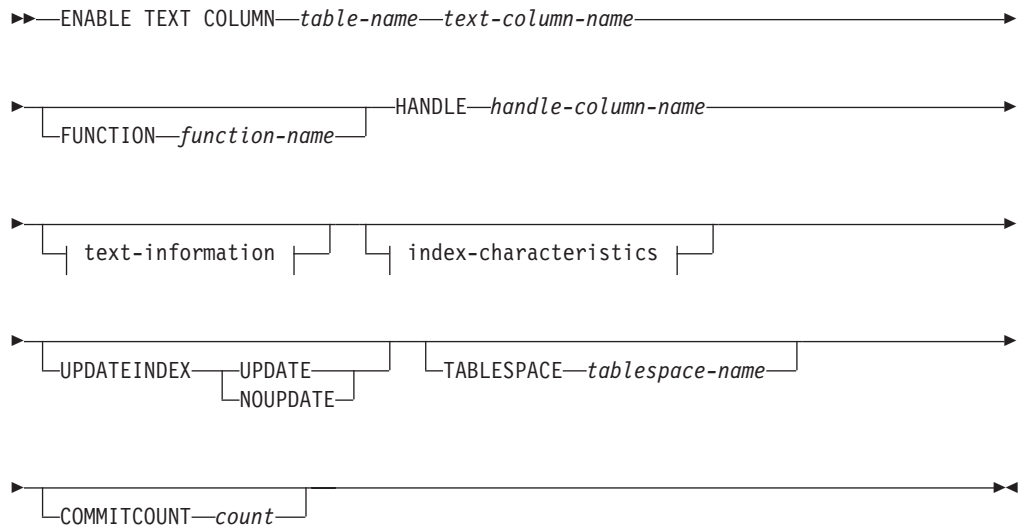
DISABLE TEXT COLUMN

This command disables a text column for use by DB2 Text Extender.

Authorization
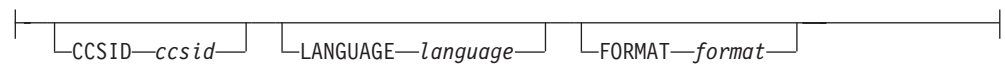
You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

►►──DISABLE TEXT COLUMN──*table-name*──HANDLE──*handle-column-name*──────────────►◄

Command parameters

table-name
The name of the text table in the connected database that contains the column to be disabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name
The name of the handle column to be disabled for use by DB2 Text Extender.

Usage notes

The index is deleted.

The log table used to record changes in the handle column (inserts, updates, and deletions) is deleted.

The triggers that write entries to the log table are deleted.

The handle column is not changed.

DISABLE TEXT FILES

This command disables a set of external text files for use by DB2 Text Extender.

Authorization

You must have at least one of the following for the table:
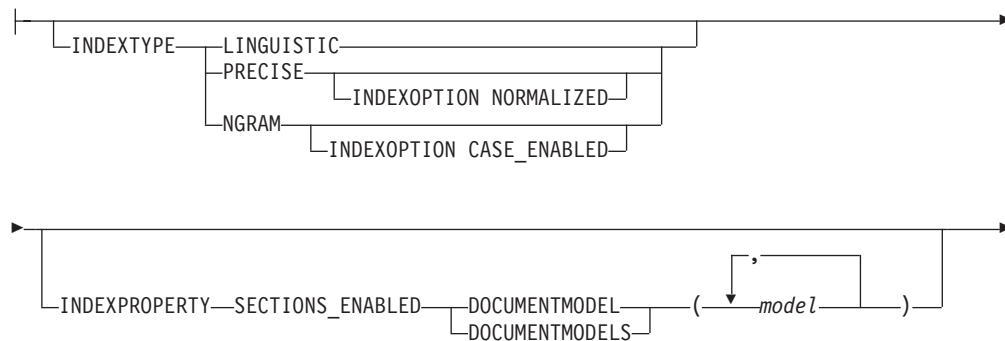
ALTER privilege
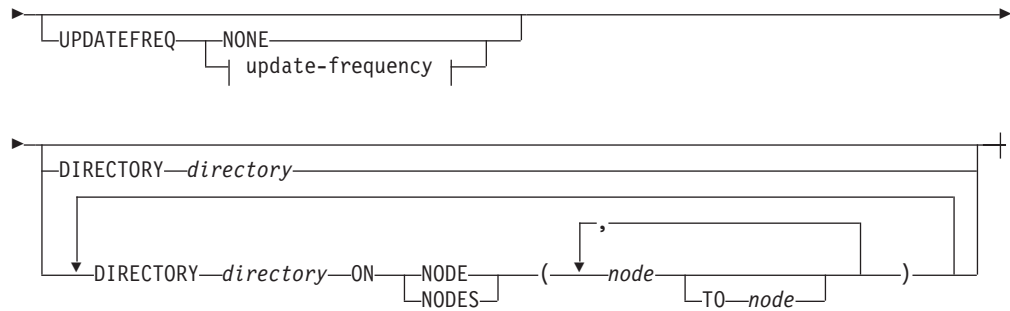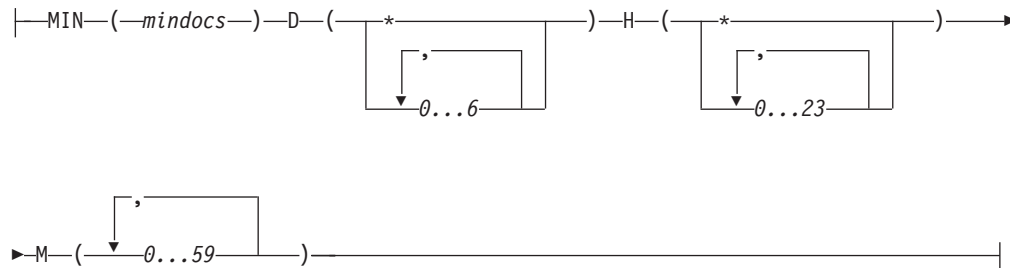SELECT privilege
UPDATE privilege.

Command syntax

►►──DISABLE TEXT FILES──*table-name*──HANDLE──*handle-column-name*──────────────►◄

Command parameters

table-name

The name of the text table in the connected database that contains the handle column for the external text files to be disabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column to be disabled for use by DB2 Text Extender.

Usage notes

The index is deleted.

The log table used to record changes in the handle column (inserts, updates, and deletions) is deleted. The triggers that write entries to the log table are also deleted.

DISABLE TEXT TABLE

This command disables all the text columns in a table for use by DB2 Text Extender.

Authorization

You must have at least one of the following for the table:

ALTER privilege

SELECT privilege

UPDATE privilege.

Command syntax

►►──DISABLE TEXT TABLE──*table-name*──────────────────────────────────────►◄

Command parameters

table-name

The name of the text table in the connected database that contains the column to be disabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

Usage notes

This command makes all the text columns in the table unusable by DB2 Text Extender.

If the text columns in this table were enabled individually by ENABLE TEXT COLUMN, it deletes all their associated text indexes. (To disable text columns and delete their associated text indexes individually, use the DISABLE TEXT COLUMN command.) If the text columns in this table were enabled together by ENABLE TEXT TABLE, there is one common index for all the text columns. This command deletes the common index.

The log tables used to record changes in the text columns (inserts, updates, and deletions) are deleted. The triggers that write entries to the log table are deleted.

ENABLE DATABASE

This command enables the current database to store text data.

Authorization

You must have:

SYSADM authority

DBADM authority.

Command syntax

►►──ENABLE DATABASE───►◄

Command parameters

None.

Usage notes

You must be connected to a database, either explicitly or implicitly, before issuing this command (see "CONNECT" on page 112).

This command prepares the connected database for use by DB2 Text Extender. It is a mandatory step before a DB2 Text Extender text table or text column can be enabled in the database.

ENABLE DATABASE creates a DB2 Text Extender catalog view called DB2TX.TEXTINDEXES, described in "Working with the DB2 Text Extender catalog view" on page 92, and a catalog view called DB2TX.TEXTCOLUMNS used for "performance" queries.

It also creates text configuration settings, described in "Text configuration settings" on page 14.

Some other work is also done, such as the declaration of DB2 Text Extender distinct types and DB2 Text Extender functions.

> **Hint**
>
> For UNIX platforms only.
>
> If the DB2 instance owner user id is not equal to the DB2 fenced user id, the fenced user id must be granted DBADM authority.

ENABLE TEXT COLUMN

This command enables a text column for use by DB2 Text Extender.

Authorization

You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

```
►►──ENABLE TEXT COLUMN──table-name──text-column-name──────────────────────────►

►─────────────────────────────────────HANDLE──handle-column-name──────────────►
      └─FUNCTION──function-name─┘

►───┬────────────────────┬──┬──────────────────────┬──────────────────────────►
    └─┤ text-information ├─┘  └─┤ index-characteristics ├─┘

►───┬─UPDATEINDEX──┬─UPDATE───┬──┬──────────────────────────────┬──────────────►
    │              └─NOUPDATE─┘  └─TABLESPACE──tablespace-name─┘

►───┬──────────────────────┬──────────────────────────────────────────────────►◄
    └─COMMITCOUNT──count─┘
```

text-information:

```
├──┬──────────────┬──┬──────────────────────┬──┬────────────────┬──┤
   └─CCSID──ccsid─┘  └─LANGUAGE──language─┘  └─FORMAT──format─┘
```

index-characteristics:

```
├──┬─INDEXTYPE──┬─LINGUISTIC──────────────────────────────────┬────┬───────────►
   │            ├─PRECISE──┬──────────────────────────────┤
   │            │          └─INDEXOPTION NORMALIZED─┘
   │            └─NGRAM────┬──────────────────────────────┘
   │                       └─INDEXOPTION CASE_ENABLED─┘

                                                                     ┌──,──┐
►───┬──────────────────────────────────────────────────────────────────────────►
    └─INDEXPROPERTY──SECTIONS_ENABLED──┬─DOCUMENTMODEL──┬──(──▼──model──┴──)─┘
                                       └─DOCUMENTMODELS─┘
```

Chapter 9. Text preparation and administration commands for the client   **119**

ENABLE TEXT COLUMN command

```
►──┬─────────────────────────────────────────────┬──────────────►
   └─UPDATEFREQ─┬─NONE────────────────┬───────────┘
               └─ update-frequency ──┘
```

```
►──┬──────────────────────────────────────────────────────────────────┬──┤
   ├─DIRECTORY──directory────────────────────────────────────────────┤
   │                                           ┌──,──┐                 │
   └─DIRECTORY──directory──ON──┬─NODE──┬──(──▼──node──┬─────────┬──)──┘
                               └─NODES─┘              └─TO──node─┘
```

update-frequency:

```
├──MIN──(──mindocs──)──D──(──┬──*───────┬──)──H──(──┬──*────────┬──)──►
                             │  ┌──,──┐  │          │  ┌──,──┐   │
                             └──▼─0...6──┘          └──▼─0...23──┘
```

```
             ┌──,──┐
►──M──(──▼──0...59──┘──)─────────────────────────────────────────┤
```

Command parameters

table-name
> The name of the text table in the connected database that contains the column to be enabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

text-column-name
> The name of the column to be enabled for use by DB2 Text Extender. This column must be of the type CHAR, VARCHAR, LONG VARCHAR, CLOB, DBCLOB, GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC. If the document type is not one of these, use FUNCTION to convert the document type.

FUNCTION function-name
> The name of a user-defined function to be used by DB2 Text Extender to access text documents that are in a column that is not of type CHAR, VARCHAR, LONG VARCHAR, CLOB, DBCLOB, GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC. See "Enabling text columns of a nonsupported data type" on page 57 for more information.

HANDLE handle-column-name
> The name of the handle column to be added to the table for use by DB2 Text Extender's functions.

CCSID ccsid
> The Coded Character Set Identifier to be used when indexing text documents.
>
> If you specify a CCSID when you enable a text column for an Ngram index, the CCSID must be the same as the CCSID of the database, and the CCSID used during search (the CCSID of the database) must match this CCSID.

To find the default CCSID, use:

```
db2tx get text cfg
```

The installation default is the database CCSID.

If this keyword is not specified, the CCSID specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

For information about other CCSIDs that can be supported, see "CCSIDs" on page 27.

LANGUAGE language

The language in which the text is written. This determines which dictionary is to be used when indexing text documents and when searching in text documents. "Chapter 15. Linguistic processing for linguistic and precise indexes" on page 221 describes how dictionaries are used.

This keyword specifies the language once for the whole column. You can override this value for individually inserted text documents using the INIT_TEXT_HANDLE function in an INSERT statement.

If this keyword is not specified, the language specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

The supported languages are listed in Table 5 on page 43.

FORMAT format

The type of text document stored, such as WordPerfect, or ASCII. DB2 Text Extender needs this information when indexing documents. The document formats supported are listed in "Which document formats are supported" on page 24.

The document formats supported for structured documents are:

- ASCII_SECTIONS

  Documents having the format ASCII_SECTIONS cannot contain nested sections. (For information about nested sections, see "Working with structured documents (section support)" on page 38.) A start tag for a section is ended by the next start tag.

- HTML

  A sample document model file is provided for HTML documents. It contains a subset of the standard HTML definitions, which you can modify. HTML documents cannot contain nested sections.

- XML

  The processing of XML documents includes Document Type Definition (DTD) evaluation. The model assigned to the document is checked against the DTD. If the tags defined in the document models file are not defined in the DTD, the document is not indexed. If no model has been defined for a recognized DTD, the document will not be indexed. XML documents can contain nested sections.

  XML is only available with OS/390 release 2.9 which includes Text Search Engine release 4 (FMID HIMN230).

For these formats, you must specify the structure information in a document model file. See "Working with structured documents (section support)" on page 38. If the format TDS and INDEXPROPERTY SECTION_ENABLED are specified, it is assumed that the document format is ASCII_SECTIONS.

Tags that are not defined in the models file are indexed in the normal way, according to the index type.

This keyword specifies the format once for the whole column. You can override this value for individually inserted text documents using the INIT_TEXT_HANDLE function in an INSERT statement.

If this keyword is not specified, the format specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

INDEXTYPE

The type of index to be created. For more information, see "Types of search" on page 32.

PRECISE

Terms are indexed and searched for exactly as they occur in the text documents.

LINGUISTIC

Terms are processed linguistically before being indexed. Search terms are also processed linguistically before the search begins.

NGRAM

Terms are indexed by parsing sets of characters rather than by using a dictionary. This index type is mandatory if the documents you are indexing contain DBCS characters, although an Ngram index can also be used for SBCS documents.

If you do not specify the INDEXTYPE keyword, the value in the text configuration settings is used.

Documents in XML format are not supported for Ngram indexes.

INDEXOPTION

Options to be used when creating the index.

CASE_ENABLED

This option is available **only for Ngram indexes**. Normally, Ngram indexes do not allow a case-sensitive search. By specifying CASE_ENABLED, you ensure that documents are indexed such that a case-sensitive search is possible. For more information see "Make a fuzzy search or search in DBCS documents" on page 34.

NORMALIZED

This option is available **only for precise indexes**. A normalized precise index differs from a precise index in that:

- It is not case-sensitive; all words except those in all uppercase are converted to lowercase.
- Words in all uppercase are not subject to stop-word filtering; the abbreviation UK, for example, is indexed.
- English language search terms may be expanded to include lemma forms using a heuristic algorithm, so that a search for house also searches for houses.

INDEXPROPERTY SECTIONS_ENABLED DOCUMENTMODEL(S) model

Properties of a selected index type.

SECTIONS_ENABLED specifies that the selected index type can contain information about the document structure.

DOCUMENTMODEL/DOCUMENTMODELS *model* specifies the model or models to be associated as default for the documents to be indexed. A model name must be specified if the index property SECTIONS_ENABLED is used. If a list of models is specified, the first model is used as the default model for the index. The default model is used during indexing if the document has no reference to a model, or if no model is specified during search.

The characters that can be used for the model name are a-z, A-Z, and 0-9.

The specified model name must correspond to a model definition in the model definition file desmodel.ini. Note that the model name is case sensitive.

To change the model or models associated with an index,

1. Use DISABLE TEXT COLUMN to disable the index
2. Use ENABLE TEXT COLUMN to reindex the documents, specifying different document model names.

UPDATEFREQ update-frequency

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

The syntax is described in "Updating an index" on page 36.

If you do not specify UPDATEFREQ, the default frequency specified in the text configuration settings is used.

> **Tip**
>
> If you have many tables, consider avoiding the use of the default values. By making individual update frequency settings for tables you can avoid indexing all the tables simultaneously and causing an unnecessarily prolonged load on your system resources.

NONE

No further index updates are made. This is intended for a text column in which there will be no further changes.

These update frequency settings are ignored if they have already been set for the whole table by ENABLE TEXT TABLE.

DIRECTORY directory

The directory path in which the text index is to be stored. The specified path is concatenated with ""txins*nnn*" where *nnn* is the node number.

This is a directory on the system where the DB2 Text Extender server is running. If the directory does not yet exist, it is created. If it already exists, ensure on UNIX platforms that the directory has read/write permissions for the instance owner, and check the 'fenced user ID' if one has been specifed that is different from the one specified when the DB2 instance was created.

If you do not specify the DIRECTORY keyword, the value of the DIRECTORY setting in the text configuration settings is used.

This setting is ignored if it has already been set for the whole table by ENABLE TEXT TABLE.

If you are using partitioned databases, you can specify one common index path name for all nodes, or for a range of nodes, or different index directory paths for each node.

ON NODE node [TO node]

The number of the node or the range of nodes to which a directory path name is being assigned.

UPDATEINDEX

A keyword that determines whether the text documents associated with this handle column are indexed immediately after this command has completed, without waiting for the next periodic indexing set by UPDATEFREQ.

UPDATE

Indexing of the text documents occurs immediately after this command has completed.

NOUPDATE

Indexing occurs at a time set by the update frequency settings specified either in this command by UPDATEFREQ, or by the text configuration setting.

If you do not specify this keyword, the value in the text configuration settings is taken.

TABLESPACE tablespace-name

The name of the table space is used to hold the log table `TLGIXnnnnnn`. The

log table size depends on the number of updates on the indexed text
column. The tablespace must have been created previously.

COMMITCOUNT count

A value from 500 to 1 000 000 indicating the number of inserts or updates
after which DB2 UDB must issue an intermediate commit statement. This
can avoid a situation in which there is insufficient log space when enabling
large tables, or columns, or a large number of external files.

Usage notes

This command adds a handle column to the specified DB2 table. Each handle
column is associated with a text column, and is used by DB2 Text Extender's
functions.

If this table has not already been enabled to create a common index, an index is
created that is associated with this text column.

Also, a log table is created in the database. The log table is used to record changes
to the text column, that is inserts, updates, and deletions. Insert, update, and delete
triggers are defined for the text column to keep the log table up to date
automatically.

If the text column that you are enabling belongs to a table that is part of a
multiple-node nodegroup, the index directory that you specify must be available
on all physical nodes. If you use the default directory specified in the text
configuration, make sure that the path is available on all nodes of the nodegroup.
If this is not convenient, you can specify a specific path for each node in the
ENABLE TEXT COLUMN command.

If you change the node configuration of a nodegroup that contains a table that is
enabled for DB2 Text Extender, you must reindex the table.

Tip

If you run out of log space in this step, see "Enabling a text column in a large
table" on page 57 for possible solutions.

ENABLE TEXT FILES

This command enables DB2 Text Extender to search in text files that are not in a DB2 UDB database.

> **Tip**
>
> This command cannot be used if the text columns in the table share a common index, as described in "Enabling a text table (optional)" on page 51.

Authorization

You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

►►──ENABLE TEXT FILES──*table-name*──HANDLE──*handle-column-name*──────────────►

►──┬─────────────────────────────┬──►
   └─┤ text-information ├──┬─┤ index-characteristics ├─┘

►──┬──┬──────►
   └─UPDATEINDEX──┬─UPDATE───┬──┬──────────────────────────────────┬─┘
                  └─NOUPDATE─┘ └─TABLESPACE──*tablespace-name*─┘

►──┬───────────────────────┬──►◄
   └─COMMITCOUNT──*count*─┘

text-information:

├──┬─────────────────┬──┬──────────────────────┬──┬──────────────────┬──┤
   └─CCSID──*ccsid*─┘ └─LANGUAGE──*language*─┘ └─FORMAT──*format*─┘

index-characteristics:

├──┬─INDEXTYPE──┬─LINGUISTIC──┬─►
                ├─PRECISE──┬───────────────────────────────┐
                │   └─INDEXOPTION NORMALIZED──┘
                └─NGRAM──┬───────────────────────────────────┐
                         └─INDEXOPTION CASE_ENABLED──┘
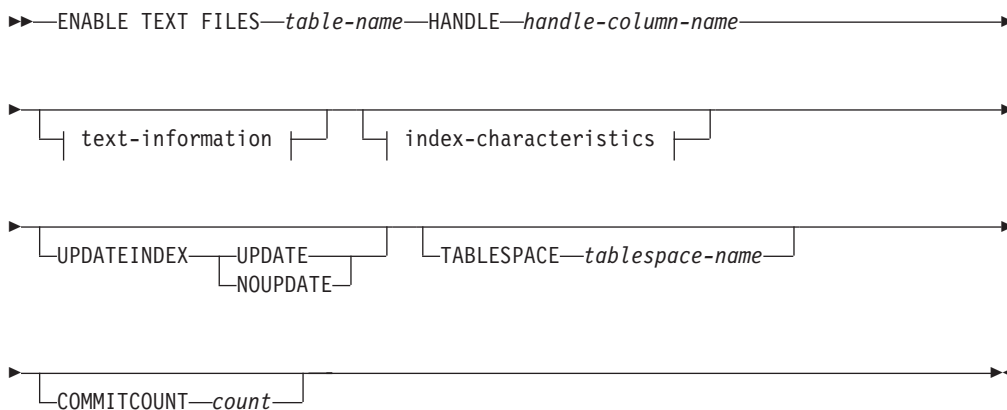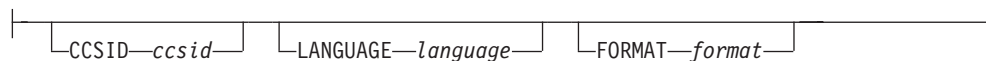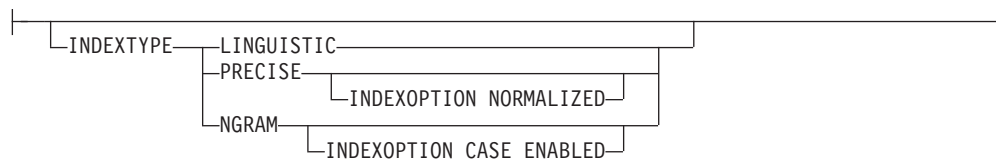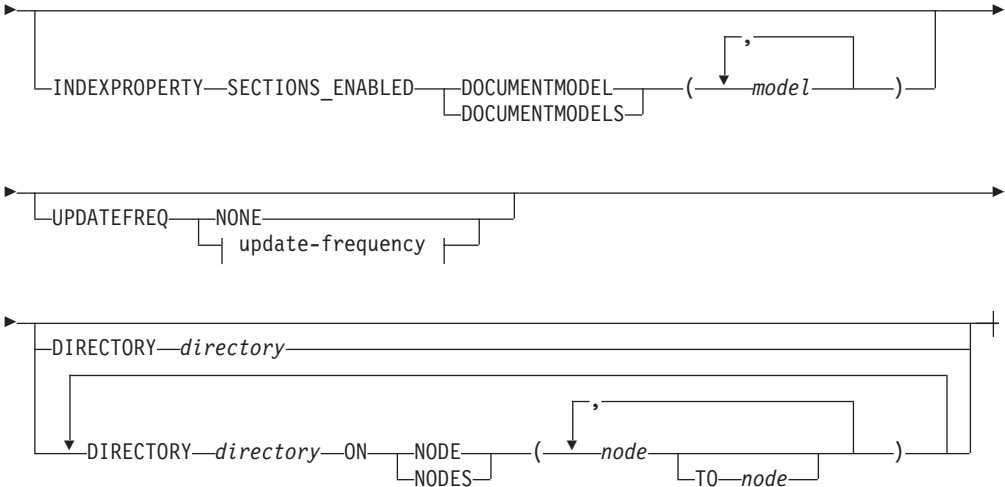
update-frequency:
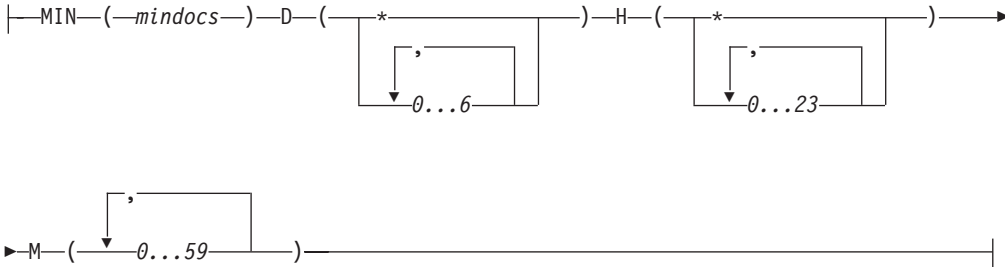
Command parameters

table-name

The name of the text table in the connected database that is to be associated with the external text files to be indexed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

For a description of the other command parameters, see "ENABLE TEXT COLUMN" on page 119.

Usage notes

This command adds a handle column to the specified DB2 table. Each handle column is associated with a collection of external text files, and is used by DB2 Text Extender's functions. An index is created that is associated with these files.

After you have enabled the text files, initialize handles in the handle column using INIT_TEXT_HANDLE. Then fill the index using UPDATE INDEX.

You cannot reuse a handle column name if that name has been used before in ENABLE TEXT FILES to identify a handle column of a text column.

A log table is created for recording changes to the files, but you must activate the triggers manually to record these changes because DB2 UDB does not have the

ENABLE TEXT FILES command

files under its control and is therefore not aware of such changes. See "Updating an index for external files" on page 83 for a description of how to do this.

If you run out of log space in this step, see "Enabling a text column in a large table" on page 57 for possible solutions.

ENABLE TEXT TABLE

Creates a common index for use by any of the table's text columns that are later enabled. The table is then a common-index table. A table that does not get enabled in this way, where the text columns that are later enabled create their own individual indexes, is a multi-index table.

Authorization

You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

```
►►──ENABLE TEXT TABLE──table-name──┬──────────────────────┬──►◄
                                   └─┤ index-characteristics ├─┘
```

index-characteristics:

```
├──┬──────────────────────────────────────────────────────────┬──►
   └─INDEXTYPE──┬─LINGUISTIC──────────────────────────────┬──┘
               ├─PRECISE─────────────────────────────────┤
               │        └─INDEXOPTION NORMALIZED─┘        │
               └─NGRAM───────────────────────────────────┤
                        └─INDEXOPTION CASE_ENABLED─┘
```

```
►──┬──────────────────────────────────────────────────────────────────────┬──►
   │                                              ┌─,──────┐                │
   └─INDEXPROPERTY──SECTIONS_ENABLED──┬─DOCUMENTMODEL──┬──(──▼─model─┴──)──┘
                                      └─DOCUMENTMODELS─┘
```

```
►──┬───────────────────────────────┬──►
   └─UPDATEFREQ──┬─NONE─────────────┬──┘
                 └─┤ update-frequency ├─┘
```

```
►──┬───────────────────────────────────────────────────────────┬──┤
   ├─DIRECTORY──directory──────────────────────────────────────┤
   │ ┌─────────────────────────────────────────────────────┐   │
   └─▼─DIRECTORY──directory──ON──┬─NODE──┬──(──▼─node─┬──────────┬──┴──)──┘
                                 └─NODES─┘            └─TO──node─┘
```

update-frequency:

```
├──MIN──(──mindocs──)──D──(──┬─*──────┬──)──H──(──┬─*───────┬──)──►
                             │ ┌─,─┐  │           │ ┌─,──┐  │
                             └─▼─0...6─┴─┘         └─▼─0...23─┴─┘
```

```
                    ┌─ , ◄──────────┐
►─M─(─────▼──── 0...59 ────┴───)──────────────────────────────┤
```
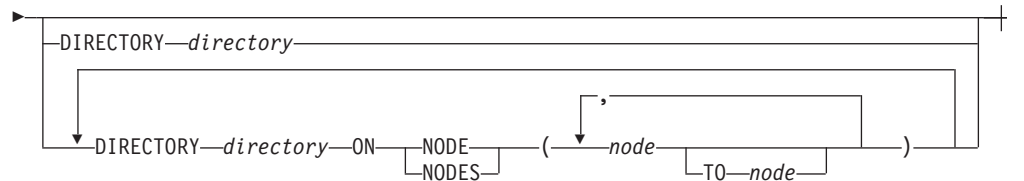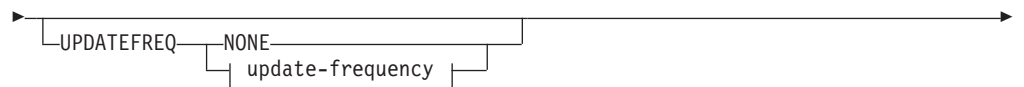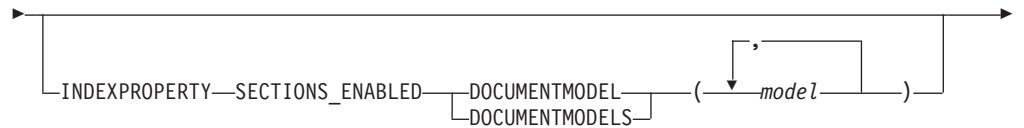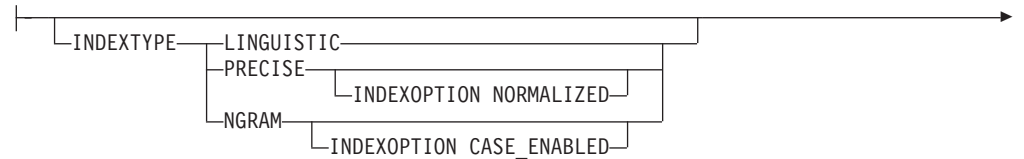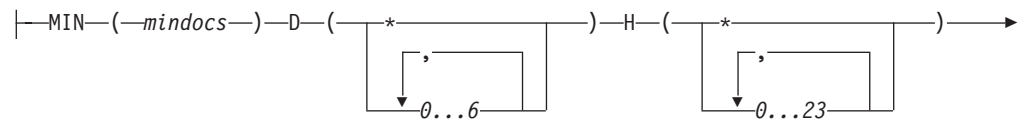
Command parameters

table-name

The name of the text table to be enabled in the connected database. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

INDEXTYPE

The type of index to be created. For more information, see "Types of search" on page 32.

PRECISE

Terms are indexed and searched for exactly as they occur in the text documents.

LINGUISTIC

Terms are processed linguistically before being indexed. Search terms are also processed linguistically before the search begins.

NGRAM

Terms are indexed by parsing sets of characters rather than by using a dictionary. This dictionary type is mandatory if the documents you are indexing contain DBCS characters, although an Ngram index can also be used for SBCS documents.

If you do not specify the INDEXTYPE keyword, the text configuration is used.

INDEXOPTION

Options to be used when creating the index.

CASE_ENABLED

This option is available **only for Ngram indexes**. Normally, Ngram indexes do not allow a case-sensitive search. By specifying CASE_ENABLED, you ensure that documents are indexed such that a case-sensitive search is possible. For more information see "Make a fuzzy search or search in DBCS documents" on page 34.

INDEXPROPERTY SECTIONS_ENABLED DOCUMENTMODEL(S) model

Properties of a selected index type.

SECTIONS_ENABLED specifies that the selected index type can contain information about the document structure.

DOCUMENTMODEL/DOCUMENTMODELS *model* specifies the model or models to be associated as default for the documents to be indexed. A model name must be specified if the index property SECTIONS_ENABLED is used. If a list of models is specified, the first model is used as the default model for the index. The default model is used during indexing if the document has no reference to a model, or if no model is specified during search.

The characters that can be used for the model name are a-z, A-Z, and 0-9.

The specified model name must correspond to a model definition in the model definition file `desmodel.ini`.

To change the model or models associated with an index,

1. Use DISABLE TEXT TABLE to disable the index

2. Use ENABLE TEXT TABLE to reindex the documents, specifying different document model names.

UPDATEFREQ update-frequency

The index update frequency in terms of *when* the update is to be made, and *how many text documents must be queued* in the log table. If there are not enough text documents in the log table at the day and time given, the index is not updated.

The syntax is described in "Updating an index" on page 36.

If you do not specify UPDATEFREQ, the default frequency specified in the text configuration settings is used.

NONE

No further index updates are made. This is intended for a text column in which there will be no further changes.

> **Tip**
>
> If you have many tables, consider avoiding the use of the default values. By making individual update frequency settings for tables you can avoid indexing all the tables simultaneously and causing an unnecessarily prolonged load on your system resources.

DIRECTORY directory

The directory path in which the text index is to be stored. The specified path is concatenated with ""txins*nnn*" where *nnn* is the node number.

This is a directory on the system where the DB2 Text Extender server is running. If the directory does not yet exist, it is created. If it already exists, ensure on UNIX platforms that the directory has read/write permissions for the instance owner, and check the 'fenced user ID' if one has been specifed that is different from the one specified when the DB2 instance was created.

If you do not specify the DIRECTORY keyword, the value of the DIRECTORY setting in the text configuration settings is used.

If you are using partitioned databases, you can specify one common index path name for all nodes, or for a range of nodes. or different index directory paths for each node.

ON NODE node [TO node]

The number of the node or the range of nodes to which a directory path name is being assigned.

Usage notes

A new text index is created that is associated with all the text columns in this table. You do this when you want to have one common index for all the text columns of a table, rather than a separate index for each text column.

When you have enabled a table, you must then run ENABLE TEXT COLUMN for each of the text columns in which you want to search.

A log table is created in the database. The table is used to record changes, that is , inserts, updates, and deletions, in the text columns that are later enabled.

ENABLE TEXT TABLE command

When a text column is enabled, triggers are created that monitor changes to the text and automatically keep a record in the log table of which documents need to be indexed.

DB2 Text Extender indexes the text documents listed in the log table periodically as specified by the UPDATEFREQ keyword.

GET ENVIRONMENT

This command displays the settings of the environment variables.

Authorization

None.

Command syntax

```
►►──GET ENVIRONMENT──────────────────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

These are the environment variables displayed:

DB2INSTANCE
DB2 UDB instance name.

DB2DBDFT
Default database name.

DB2TX_INSTOWNER
DB2 Text Extender instance name. Required only if you connect to a UNIX server.

DB2TX_INSTOWNERHOMEDIR
Instance owner's home directory. Required only if you connect to a UNIX server.

GET INDEX SETTINGS

This command displays the settings of an index, showing the following:

- Index type
- Index option (optional)
- Update index option
- Index directory
- Update frequency
- Default model.

Authorization

None.

Command syntax

```
►►──GET INDEX SETTINGS──table-name──────────────────────────────────────────►◄
                                    └─HANDLE──handle-column-name─┘
```

Command parameters

table-name
> The name of the text table in the connected database whose index settings are to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name
> The name of the handle column whose index settings are to be displayed.

Usage notes

If the table is enabled as a multi-index table, this command displays the index settings of all enabled text columns in the table. If a *handle-column-name* is provided, this command displays the index settings of the specified column.

If the table is a common-index table, the settings of the common index are displayed. If a *handle-column-name* is provided, it is ignored.

If the table or column is enabled with the index property SECTIONS_ENABLED, the command GET INDEX SETTINGS displays the default model for the index. The default model is the model name you specified during enabling or the first model name in a list of model names.

Here is an example:

```
Current index settings:

        Index type              (INDEXTYPE)   = LINGUISTIC
        Default model          (DOCUMENTMODEL) = mymodel
        Update index option     (UPDATEINDEX)  = UPDATE
        Update frequency         (UPDATEFREQ)  = NONE
        Node 1
        Index directory          (DIRECTORY)   = /home/user1/db2tx/indices
```

GET INDEX STATUS

This command displays the following index status information for a given handle column or table:

- Whether the search function is available
- Whether the index update function is available
- Whether the reorganize function is available
- The number of scheduled documents
- The number of indexed documents
- The number of indexed documents in the primary index
- The number of indexed documents in the secondary index
- Error events.

Authorization

None.

Command syntax

```
►►──GET INDEX STATUS──table-name──────────────────────────────────────►◄
                              └─HANDLE──handle-column-name─┘
```

Command parameters

table-name

The name of the text table in the connected database that contains the text columns whose status is to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose status is to be displayed.

Usage notes

For a multi-index table, you must specify the name of the handle column.

GET STATUS

This command displays information about the enabled status of databases, tables, or text columns.

Authorization

None.

Command syntax

```
►►──GET STATUS──────────────────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

This command displays whether the database is enabled, the names of the enabled text tables in the database, the names of the enabled text columns and their associated handle columns, and the names of external-file handle columns.

GET TEXT CONFIGURATION

This command displays the default settings for the text configuration for the connected database.

To change these default settings, use "CHANGE TEXT CONFIGURATION" on page 109.

Authorization

You must have at least one of the following for the table:

SELECT privilege

Command syntax

```
►►──GET TEXT──┬─CONFIGURATION─┬────────────────────────────────────────────►◄
              └─CFG───────────┘
```

Command parameters

None.

Usage notes

For an example of the text configuration information, see "Displaying the text configuration settings" on page 87.

GET TEXT INFO

This command displays the text information settings for text columns:

- CCSID
- Language
- Format.

Authorization

None.

Command syntax

```
►►──GET TEXT INFO──table-name──────────────────────────────────────────►◄
                             └─HANDLE──handle-column-name─┘
```

Command parameters

table-name

The name of the text table in the connected database that contains the text columns whose text information settings are to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose text information settings are to be displayed.

Usage notes

If a handle column name is given, the text information for only this column is displayed.

If a handle column name is not given, the text information for each enabled column in this table is displayed.

QUIT

This command stops the DB2 Text Extender command line processor and returns control to the operating system.

Authorization

None.

Command syntax

```
►►──QUIT─────────────────────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

The connection to the database is terminated.

REORGANIZE INDEX

If a text column is often updated, searching the index can become inefficient. To make searching effient again, the index has to be *reorganized*. Although DB2 Text Extender recognizes when an index needs to reorganized and does so in the background automatically, there may be situations that require an index to be reorganized manually using REORGANIZE INDEX. You can use the command GET INDEX STATUS to find out if an index needs to be reorganized.

Authorization

None.

Command syntax

```
►►──REORGANIZE INDEX──table-name──────────────────────────────────────►◄
                              └─HANDLE──handle-column-name─┘
```

Command parameters

table-name
> The name of the text table in the connected database whose index is to be reorganized. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name
> The name of the handle column whose index is to be reorganized.

Usage notes

For a multi-index table, you must specify a handle column name.

Although searches can be made on the index while REORGANIZE INDEX is running, index updates cannot.

RESET INDEX STATUS

When the index status of a table or column shows `Search not available` or `Update not available`, an error has occurred during indexing that prevents you working with the index.

This command resets the index status so that you can continue to work with it. Before resetting the index status, check for any errors that may be logged in the index's log table (see "Displaying error events" on page 89).

Authorization

None.

Command syntax

```
►►──RESET INDEX STATUS──table-name───────────────────────────────────────►◄
                          └─HANDLE──handle-column-name─┘
```

Command parameters

table-name
The name of the text table in the connected database that contains the text columns whose status is to be reset. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name
The name of the handle column whose status is to be reset.

Usage notes

For a multi-index table, you must specify a handle column name.

For a common-index table, each enabled column in this table is reset.

UPDATE INDEX

This command starts indexing immediately. It brings the index up to date to reflect the current contents of the text column(s) with which the index is associated.

To have updates on external files reflected in the index, you must force a "change" entry to be placed in the log table by issuing an update statement on the corresponding handle column. See "Updating an index for external files" on page 83 for an example.

Authorization

You must have at least one of the following for the table:

ALTER privilege
SELECT privilege
UPDATE privilege.

Command syntax

```
►►──UPDATE INDEX──table-name──────────────────────────────────────────────►
                            └─HANDLE──handle-column-name─┘

►──────────────────────────────────────────────────────────────────────►◄
   └─COMMITCOUNT──count─┘
```

Command parameters

table-name

The name of the text table in the connected database that contains the text column whose index is to be updated. This can also be the name of a common-index table. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

If this is a common-index table, the *handle-column-name* is not required and is ignored. The index to be updated is associated with the whole table and not with an individual text column.

If this is a multi-index table, then *handle-column-name* is the name of the handle column whose index is to be updated.

COMMITCOUNTcount

A value from 500 to 1 000 000 indicating the number of inserts or updates after which DB2 UDB must issue an intermediate commit statement. This can avoid a situation in which there is insufficient log space when enabling large tables, or columns, or a large number of external files.

UPDATE INDEX command

Chapter 10. Administration commands for the server

This chapter describes the syntax of the administration commands for the server. Server administration consists of tasks you can do to start, stop, and check the status of the DB2 Text Extender server, and to create a sample database and sample a table. "Setting up and maintaining a DB2 Text Extender server" on page 16 describes how to use these commands.

Command	Purpose	Page
TXICRT	Creates the DB2 Text Extender instance	146
TXIDROP	Drops the DB2 Text Extender instance	148
TXILIST (UNIX only)	Lists DB2 Text Extender instances (UNIX only)	149
TXIPCLEAN (UNIX only)	Cleans up system resources after a system failure (UNIX only)	150
TXNADD	Adds DB2 Text Extender servers	151
TXNCHECK	Checks DB2 Text Extender database integrity	152
TXNDROP	Drops DB2 Text Extender servers	153
TXSAMPLE	Creates and enable a sample table	154
TXSERVICE	Copies DB2 and Text Extender setup information to a file	155
TXSTART	Starts the DB2 Text Extender services	157
TXSTATUS	Displays the status of the search service	158
TXSTOP	Stops the DB2 Text Extender services	159
TXTHESC	Compiles a thesaurus definition file	160
TXTHESN	Compiles an Ngram thesaurus definition file	162
TXTRACE	Produces trace information	164
TXVERIFY	Verifies the correct installation.	168
TXWIZARD	An example program for indexing and search (AIX, Windows NT, and Windows 2000 only	169

TXICRT

This command creates the DB2 Text Extender instance.

Authorization

(UNIX only) Root

Command syntax

For UNIX:

```
►►─txicrt─instance-name─db2-instance-name──────────────────────────────────►
                                          └─port-start──┐
                                                   └─port-end─┘

►─┬──────────────────────┬──┬────────┬──────────────────────────────────►◄
  └─-nodes─identifiers─┘  └─-client─┘
```

For Windows NT and Windows 2000:

```
►►─txicrt─db2-instance-name──────────────────────────────────────────────►
                          └─port-start──┐
                                   └─port-end─┘

►─┬──────────────────────┬──┬────────┬──────────────────────────────────►◄
  └─-nodes─identifiers─┘  └─-client─┘
```

Command parameters

instance-name
> (UNIX only) The user ID of the instance owner.

db2-instance-name
> The name of the DB2 instance to be associated with the instance being created.

port-start/port-end
> The TCP/IP port range to be used if you are working with more than one physical node. Not required if you are using only logical nodes.
>
> The port range must be equal to the number of nodes you want to work on. If you are working with one node, specify only a `port-start` parameter. To support a subset of nodes, use the `port-last` parameter.
>
> Do not specify a port range if the instance is created on a single-partition database.
>
> Use only port numbers that are not already listed in the TCP/IP `etc/services` file. Do not specify port numbers for client-only installations.

-nodes *identifiers*
> A list of node identifiers. Example: `-nodes 0 1 2`.
>
> Use this parameter if your table belongs to specific nodes in a nodegroup.

If this parameter is not specified, all nodes are used, that is, all the nodes defined in db2nodes.cfg.

Do not specify node identifiers for client-only installations.

-client

Specify this parameter if you have a DB2 Text Extender client-only installation. If you do not specify this parameter, TXICRT configures a client/server instance of DB2 Text Extender.

Usage notes

Enabling text tables or columns is possible only on nodes wherewhen you have created the DB2 Text Extender instance.

A DB2 instance must already have been created for the user ID.

In UNIX installations, this instance-specific command is in the /usr/lpp/db2tx_07_01/instance directory.

On Windows NT or Windows 2000 systems, a default instance is created at installation time in a non-partitioned database environment.

TXIDROP

This command drops the DB2 Text Extender instance together with all its indexes.

Authorization

(UNIX only) Root

Command syntax

For UNIX:

►►──txidrop──*instance-name*──*db2-instance-name*─────────────────────────►◄

For Windows NT and Windows 2000:

►►──txidrop──*instance-name*──►◄

Command parameters

instance-name
> The name of the instance to be dropped.

db2-instance-name
> The name of the DB2 instance associated with the instance to be dropped.

Usage notes

Before dropping an instance, disable any databases that are enabled for it.

In UNIX installations, this instance-specific command is in the
`/usr/lpp/db2tx_07_01/instance` directory.

TXILIST (UNIX only)

This command lists the instances for the current DB2 Text Extender version.

Authorization

SYSADM, DBADM

Command syntax

```
►►──txilist─────────────────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

If there are more than one version of DB2 Text Extender installed, this command lists only the instances for the DB2 Text Extender version you are currently working with, that is, the version determined by the current directory.

TXIPCLEAN (UNIX only)

This command cleans up system resources for the logged on user, that is, it kills the `desssdemprocess`, cleans up shared memory, segements message queues and semaphores.

Authorization

Member of the smadmin group

Command syntax

```
►►──txipclean───────────────────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

Use this command with care; it cleans up not only DB2 Text Extender resources, but all the shared resources of the logged-on user. Always log on as the instance owner to limit the clean up to the instance resources.

There may be situations when you cannot start or stop DB2 Text Extender properly. In this case, try to stop DB2 and DB2 Text Extender first before running TXIPCLEAN.

TXNADD

This command creates an additional DB2 Text Extender server on the specified node. The TXSTART command subsequently starts the server.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txnadd──node-number──────────────────────────────────────►◄
                       └─port─┘
```

Command parameters

node-number
> The number of the node to be added. Specify the node number exactly as it is specified in the db2nodes.cfg file.

port The port number to be used for the added node. You must specify a port number if you are adding a physical node, or if your node configuration contains physical nodes.

Usage notes

Use TXNCHECK to check which indexes you must recreate to remain synchronous with your database.

> **Tip**
> If you add a physical node to a node configuration that has logical nodes, you must first drop the instance and then recreate the instance, specifying port numbers for all the nodes.

TXNCHECK

This command checks the DB2 Text Extender integrity for a given database.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txncheck─────────────────────────────────────────────────►◄
              └─database-name─┘
```

Command parameters

database-name
> The name of the database to be checked. If a database name is not
> specified, the name of the DB2 default database is used.

Usage notes

A DB2 Text Extender-enabled table must be reindexed if the nodegroup it is in is
changed. Use TXNCHECK to display which DB2 Text Extender-enabled tables
need to be reindexed due to a redistributed nodegroup.

TXNDROP

This command removes a DB2 Text Extender server for a given node.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txndrop──node-number──────────────────────────────────────────────────►◄
```

Command parameters

node-number
> The number of the node to be dropped. Specify the node number exactly as it is specified in the `db2nodes.cfg` file.

Usage notes

Use TXNCHECK to check which indexes you must recreate to remain synchronous with your database.

TXSAMPLE

This command creates a sample table, loads sample English documents into column COMMENT, and enables the text column. See "Preparing a sample database for installation verification" on page 12.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
▶▶──txsample──database-name─────────────────────────────────────────────◀◀
                            └─ccsid─┘  └─user-id──password─┘
```

Command parameters

database-name
The name of the database, already enabled by DB2 Text Extender, in which the sample table is to be created.

user-id
This is only required if you are working from a client workstation. It is only supported on Microsoft Windows.

password
This is only required if you are working from a client workstation. It is only supported on Microsoft Windows.

ccsid The CCSID of the database is only supported on Microsoft Windows. See the output of the DB2 command GET DATABASE CONFIGURATION. For information on this command, see the *DB2 Command Reference* guide.

Usage notes

If your database CCSID is not 850, take a look at the example enabling step for the Ngram index type, and there, change the CCSID to match your database CCSID.

> **Tip**
> This command can also be used on a client workstation supported on Microsoft Windows.

TXSERVICE

This command is intended for service issues only. It copies information about current DB2 and Text Extender setup to a file, which in the case of Text Extender problems, can be provided to IBM support.

Authorization

DB2 instance owner

Command syntax

```
►►──txservice──output-file──database-name──────────────────────────────────►◄
                                        └──table-name──────────┘
                                                  └──handle-column──┘
```

Command parameters

output-file
> The path and file name of the output file to be provided to IBM support.

database-name
> The name of the database in which the problems occur.

table-name
> The name of the database table that is related to, or causes the problem. If the problem occurs during the creation of the Text Extender instance, or when enabling your database for Text Extender, you do not need this parameter as the problem is not related to the database table.

handle-column
> The name of the handle column of the text index where the problem occurs. This parameter should be included with the table-name, if the problems are related to one or more text indexes. If the command returns an error message stating that the handle-column does not exist, run the command again without the handle-column parameter.

Examples

When performing a search on the text-index with the handle H_DOC, on table DATA.DOCUMENTS in database CUST_INFO you get the error message DES0332N, "An unexpected error occurred at the DB2 Text Extender server". Issue the following command:

```
txservice service.out CUST_INFO DATA.DOCUMENTS H_DOC
```

and provide the file service.out, the command that caused the error, and the error message to IBM support.

Usage notes

The command will not apply any changes to your system. The following information will be stored in the output file:

- information about the installed levels of DB2 and Text Extender
- details on the installed Text Extender files
- contents of the Text Extender error log file, desdiag.log
- contents of the internal Text Extender administration tables

TXSERVICE command

- a list of all text indexes created for the current database
- database and database manager configurations
- the session settings for all environment variables
- the structure of the database table provided with the table parameter
- the status and settings of the text index provided with the handle column parameter

The command will not save any information stored in the tables you created text indexes for. The output file will be a simple text file.

TXSTART

This command starts the DB2 Text Extender search services.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txstart──┬──────────────────────┬──────────────────────────►◄
             └─NODENUM──nodenum──┘
```

Command parameters

NODENUM nodenum

The node number of the server to be started. If a node number is not specified, all servers are started.

Usage notes

Run this command:

- While logged on with a user ID in the SM administration group
- Whenever you stop and restart your server system

Before you can index data or search on your data, you have to start the DB2 Text Extender search services.

If you have problems starting the DB2 Text Extender services, stop DB2 and run the command TXIPCLEAN. See "TXIPCLEAN (UNIX only)" on page 150.

TXSTATUS

This command displays whether DB2 Text Extender is up and running.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txstatus──────────────────────────────────────────────►◄
            └─NODENUM──nodenum─┘
```

Command parameters

NODENUM nodenum

The node number of the server whose status is to be checked. If a node number is not specified, the status of all servers is displayed.

TXSTOP

This command stops the DB2 Text Extender services.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txstop───────────────────────────────────────────────────────►◄
              └─NODENUM──nodenum─┘
```

Command parameters

NODENUM nodenum

The node number of the server whose status is to be checked. If a node number is not specified, the all servers are stopped.

Usage notes

This command does not stop DB2.

TXTHESC

This command compiles a thesaurus definition file. This thesaurus can be used only for searches on precise or linguistic indexes.

For compiling an Ngram thesaurus definition file, see "TXTHESN" on page 162.

Authorization

(UNIX only) None.

Command syntax

```
>>──txthesc──┬──-ccsid──code-page──── -f──definition-file-name──┬─────────────><
             ├──-quiet─────────────────────────────────────────┤
             │   ├──-h───┤                                       │
             │   ├──-H───┤                                       │
             │   ├──-?───┤                                       │
             └──-copyright─┘
```
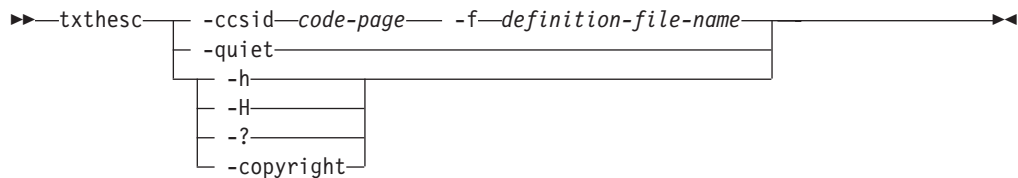
Command parameters

-f *definition-file-name*
> The name of the SGML file containing the thesaurus definition. The file name must contain either the absolute path of the relative path to the file.
>
> The thesaurus dictionary is generated in the same directory as the definition file. It has the same name as the definition file but with the extensions th1 through th6.

> ---
> **Tip**
> Because thesaurus files are overwritten when they have the same names, use a separate directory for each thesaurus.
> ---

-ccsid *code-page*
> The code page in which the thesarus definition file is written. Currently, only code page 850 is supported.

-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 notes

Use this command to compile a standard thesaurus definition file into a binary thesaurus definition format. The definition file must be in SGML format.

To use a compiled thesaurus file, move it to the dictionary directory of the server instance, then specify the location of the files during searching.

The dictionary directory on OS/2 and Windows systems is:

drive`:\dmb\db2tx\dict`

On AIX, HP-UX, and SUN-Solaris systems, the dictionary directory is:
DB2TX_INSTOWNER `/db2tx/dicts`

TXTHESN

This command compiles an Ngram thesaurus definition file. This thesaurus can be used only for searches on Ngram indexes.

Authorization

(UNIX only) None.

Command syntax

```
►►──txthesn──┬─ -ccsid──code-page── -f──definition-file-name──┬──────────────►◄
             ├─ -quiet───────────────────────────────────────┤
             │    ├─ -h─┤                                      │
             │    ├─ -H─┤                                      │
             │    ├─ -?─┤                                      │
             └─ -copyright─┘
```
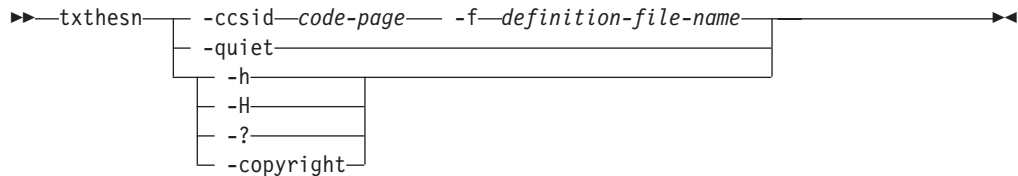
Command parameters

-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 is optional.
>
> The thesaurus dictionary is generated in the same directory as the definition file. It has the same name as the definition file but with the extensions wdf, wdv, grf, grv, MEY, ROS, NEY, SOS, and lk*n*, where *n* is a digit.

> ┌─ **Tip** ──
> │ Because thesaurus files are overwritten when they have the same
> │ names, you should use a separate directory for each thesaurus.
> └───

-ccsid *code-page*
> The code page in which the thesarus definition file is written. For a list of the supported code pages, see "CCSIDs" on page 27.

-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 notes

Use this command to compile a thesaurus definition file into a binary thesaurus definition format. The definition file must be in the format described in "Creating an Ngram thesaurus" on page 238.

To use a compiled thesaurus file, move it to the dictionary directory of the server instance, then specify the location of the files during searching.

The dictionary directory on OS/2 and Windows systems is:

`drive:\dmb\db2tx\dict`

On AIX, HP-UX, and SUN-Solaris systems, the dictionary directory is:

`DB2TX_INSTOWNER /db2tx/dicts`

TXTRACE

This command writes processing information to a trace buffer in shared memory. It can be written in binary from the trace buffer to a file for later formatting when tracing has been switched off, or can be formatted and written to a file while tracing is still on.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txtrace──┤ parameters ├──────────────────────────────────────►◄
```

parameters:

```
├──┬─on────────────────────────┬──────────────────────────────────┤
   │     ┌─ options ─┐          │
   ├─dump─┤ dump-file ├─────────┤
   ├─off────────────────────────┤
   ├─format──┬──────────┬──┬───────────┬──>──┬ format-file ├──┤
   │         └─ options ─┘  └─ dump-file ─┘                    │
   ├─change─────────────────────┤
   ├─clear──────────────────────┤
   ├─flow───────────────────────┤
   └─information────────────────┘
```

options:

```
├──┬─────────────────────────────────────────────────────────────┬──┤
   ▼                                                               │
   ├─ -m──products──.──events──.──components──.──functions──┤
   ├─ -rc──return-code──────────────────────────────────────┤
   ├─ -e──max-system-errors─────────────────────────────────┤
   ├─ -r──max-record-size───────────────────────────────────┤
   └─ -i──initial-buffer-size──┬── -l──last-buffer-size──────┘
```

dump-file:

```
├──┬───────┬──dump-filename──────────────────────────────────────┤
   └─path─┘
```

format-file:

```
├──┬───────┬──format-filename────────────────────────────────────┤
   └─path─┘
```
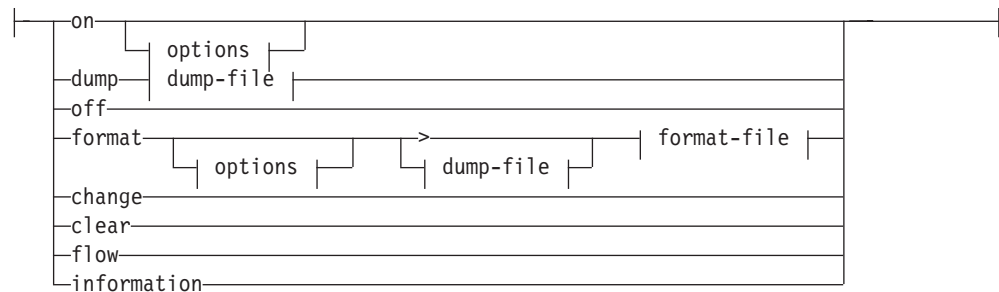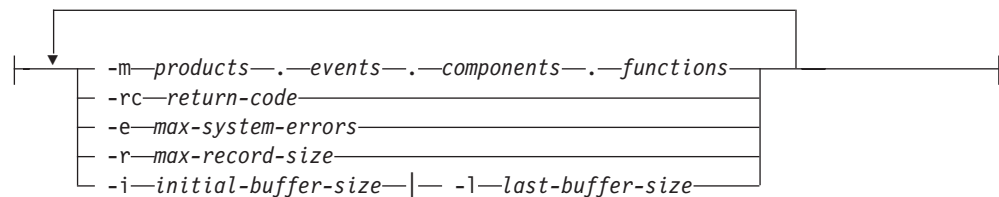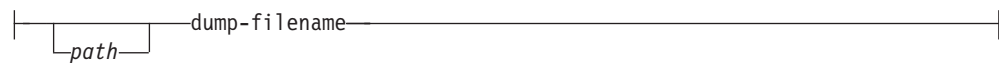
Command parameters

Note:

A -u option is also available with all of the TXTRACE parameters to display information about the parameter.

on   To start the trace facility.

dump | dmp
To write the trace information in binary to a file.

off   To stop the trace facility.

format | fmt
To format the binary trace information. You can format the dump file, when tracing is switched off, by specifying the name of the dump file and the name of the file to hold the formatted trace information. To format the trace information directly from the trace buffer while tracing is still switched on, enter: destrc format > filename.tmp

change | chg
To change the trace mask, maxSysErrors, or maxRecordSize.

clear | clr
To clear the trace.

flow | flw
To show control flow of the trace.

information | info | inf
To get information about the trace.

options
To filter the trace information either when turning tracing on (this reduces the shared memory usage), or when formatting the trace information. Unless the trace is very large, it is usually easier to write the full trace information and then filter it during formatting.

-m   To add a "mask" to specify which events, components, and functions are to be included in the trace. The default is to trace everything. The mask is in four parts, separated by periods; for example: 2.2-6.1,3.* You can specify a range using "-" as a separator character, or a list using "," as a separator character. For example: 2-6 includes only the events whose IDs are in the range from 2 to 6. To include only components 2 *and* 6, specify 2,6

products
Product ID. The product ID for DB2 Text Extender is "2". The product ID for TextMiner is "3".

events The set of event types to be included in the trace:
0	system_error
1	system_error
2	system_error
3	non-fatal_error
4	non-fatal_error
5	api_errcode
7	fnc_errcode
8	trap error
10	api_entry
11	api_exit

13	api_retcode
15	api_data
30	fnc_entry
31	fnc_exit
33	fnc_retcode
35	fnc_data

components

The components to trace.

The component IDs for DB2 Text Extender are:

1	COMMAND_LINE_INTERFACE
2	UDF
3	STORED_PROCEDURES
4	ADMINISTRATION
5	INDEX_CONTROL
6	LIBRARY_SERVICES
7	DES_PARSER
8	DES_DEMON
9	DES_API
10	SERVICES

The component IDs for TextMiner are:

1	automachine
2	bgproc (background processing)
3	cluster
4	common
5	commsrvc (common services)
6	communic (communication)
7	daemon
8	dsclient
9	environ (environment)
10	glue
11	idxcomm (index build, common part)
12	libsrv (library services)
13	search
14	trace
15	guru
16	indexbld (index build, tm only)
17	indexeng (index engine, tm only)
18	smsearch
19	search engine, tm only)
20	tmsearch
21	gtrcm (gtr, common part)
22	gtrsrch (search, gtr only)
23	gtridx (index build, gtr only)

functions

Asterisk (*). The set of functions to trace. Use an asterisk (*) to trace all functions unless directed to do otherwise by the IBM Support Center.

-rc *return-code*

Treat *return-code* as a system error.

-e *max-system-errors*

Integer. To stop the trace after this number of errors. The default is 1 which specifies that when the first system error occurs, all

subsequent tracing of lower severity events is suppressed. This is acceptable if you are interested only in the first major error, but you should specify a higher number (such as -e 50) if you want to see the full trace after the initial system error. The trace destination is shared memory.

-r *max-record-size*
Integer. To stop the trace after this number of records have been written to the trace file. The default is 16 KB.

-i *initial-buffer-size*
Integer. To keep this number of records from the beginning of the trace. If -i is specified, the default is 16 KB. On a UNIX system, a recommended buffer size is 2 MB.

If neither -i nor -l are specified, -l is the default.

If you specify -i, no wrapping occurs; no further trace entries are written if the volume of records exceeds *max-record-size*, even if you clear all trace entries. To get new trace entries written, increase the buffer size, turn the trace off and then on again.

-l *last-buffer-size*
Integer. To keep this number of records from the end of the trace. The default is 16 KB. On a UNIX system, a recommended buffer size is 2 MB.

path The directory where the corresponding file is stored.

dump-filename
The name of the file that contains the binary trace information.

formatted-filename
The name of the file that contains the formatted trace information.

Examples

See "Tracing faults" on page 18.

TXVERIFY

This command creates a sample database and enables it.

Authorization

(UNIX only) SYSADM, DBADM

Command syntax

```
►►──txverify──database-name─────────────────────────────────────────►◄
                            └─user-id──password─┘
```

Command parameters

database-name
>   The name of the database in which the sample table is to be created.

user-id
>   This is required only if you are working from a client workstation.

password
>   This is required only if you are working from a client workstation.

Usage notes

This command can also be used on a client workstation.

TXWIZARD

This command is available only in AIX, Windows NT, and Windows 2000 systems. It is provided as an example of how to create indexes on selected tables or columns, and to search on these indexes.

Command syntax

```
►►──txwizard──────────────────────────────────────────►◄
```

Command parameters

None.

Usage notes

This sample wizard leads you through the steps required to be able to search on selected columns of a table using the DB2 Text Extender search functions. You can use the Java source code that is provided with the sample wizard to either build your own text preparation and search application or to learn how the DB2 Text Extender functions can be used and called within a Java environment.

In AIX, the Java source code of the wizard is stored in directory
`/usr/lpp/db2tx_07_01/samples/wizard`

In Windows NT and Windows 2000, it is stored in directory
`%DMBMMPATH%\samples\wizard`

TXWIZARD command

Chapter 11. Search functions

DB2 Text Extender provides SQL functions to search in text documents stored in DB2 UDB , and to work with the results of a search. Some of the functions' parameters are data types known as *distinct types* that are provided with DB2 Text Extender.

This chapter describes the DB2 Text Extender SQL functions and distinct types.

DB2 Text Extender provides a DB2 UDB command line processor input file called `txsample.udf`. It contains examples of DB2 Text Extender functions that run against the sample table described in "Preparing a sample database for installation verification" on page 12. Use this file to see examples of the syntax of the text preparation and search functions.

The DB2 Text Extender distinct types

Distinct type	Source data type	Comments
DB2TEXTH	VARCHAR(60) FOR BIT DATA	**Text handle.** A variable-length string containing information needed for indexing a text document stored in a text column. The information in a handle includes a document ID, the name of the server where the text is to be indexed, the name of the index, and information about the text document. Handles are stored in columns that DB2 Text Extender creates and associates with each text column. The functions HANDLE and INIT_TEXT_HANDLE return this data type.
DB2TEXTFH	VARCHAR(210) FOR BIT DATA	**File handle.** A variable-length string containing information needed for indexing an external text file – a file stored outside of the control of DB2 UDB . The information in a text handle includes a document ID, the name of the server where the text is to be indexed, the name of the index, information about the text file, and information about the location of the file. File handles are stored in columns that DB2 Text Extender creates and associates with each group of external files. The functions FILE and INIT_TEXT_HANDLE return this data type.
DB2TEXTHLISTP	VARCHAR(16) FOR BIT DATA	**Handle list pointer.** A pointer to a list of handles associated with text documents found by a search. The function HANDLE_LIST returns this data type.

　　　　　　　　　　　　　　　　　　　　　　　　　　　　　171

DB2 Text Extender distinct types

Distinct type	Source data type	Comments
DB2TEXTFHLISTP	VARCHAR(16) FOR BIT DATA	**Handle list pointer.** A pointer to a list of handles associated with external files found by a search. The function HANDLE_LIST returns this data type.

A summary of DB2 Text Extender functions

Search function	Purpose	Page
CCSID	Returns the CCSID from a handle	173
CONTAINS	Makes a search for text in a particular document	174
FILE	Returns or changes the path and name of a file in an existing handle	175
FORMAT	Returns or changes the document format setting in a handle	176
HANDLE[1]	Returns a handle from a list of handles	177
HANDLE_LIST[1]	Searches and returns a list of handles	178
INIT_TEXT_HANDLE	Returns a partially initialized handle containing information such as format and language settings	179
LANGUAGE	Returns or changes the language setting in a handle	180
NO_OF_DOCUMENTS[1]	Returns the number of documents listed in a handle list	181
NO_OF_MATCHES	Searches and returns the number of matches found	182
RANK	Searches and returns the rank value of a found text document	183
REFINE	Takes a search argument and a refining search argument and returns a combined search argument	184
SEARCH_RESULT	Returns an intermediate table with the search result of the specified search string	185

Examples of the use of DB2 Text Extender functions are given in "Chapter 6. How to search" on page 63.

---

1. These search functions are features of an earlier release of DB2 Text Extender. For compatibility reasons they continue to be supported, Their functionality has been superceded by the SEARCH_RESULT search function.

CCSID

The CCSID function returns the CCSID (data type SMALLINT) from a handle. This is the CCSID parameter used for indexing the corresponding text document. This is described in "CCSIDs" on page 27. It is set for each text column by the ENABLE TEXT COLUMN command.

Function syntax

```
►►──CCSID──(──handle──)──────────────────────────────────────────►◄
```

Function parameters

handle

An expression whose result is a value of type DB2TEXTFH orDB2TEXTH. It is usually the name of a handle column from which the CCSID setting is to be returned.

CONTAINS

The CONTAINS function searches for text in a particular text document. It returns the INTEGER value 1 if the document contains the text. Otherwise, it returns 0.

Function syntax

►►──CONTAINS──(──*handle*──,──*search-argument*──)────────────────────◄

Function parameters

handle

An expression whose result is a value of type DB2TEXTFH or DB2TEXTH. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.

FILE

The FILE function does one of the following:

- Returns the path and file name in a handle
- Changes the path and file name in a handle, and returns the path and file name.

The returned handle is a value of type DB2TEXTFH.

Function syntax

```
►►──FILE──(──handle──)─────────────────────────────────────────────────►◄



►►──FILE──(──handle──,──file-name──)────────────────────────────────────►◄
```

Function parameters

handle
   An expression whose result is a value of type DB2TEXTFH. It is usually the name of a handle column from which the file name is to be returned.

file-name
   A string of type VARCHAR(150) specifying the new absolute path and file name of the external file that is to be associated with the handle. The path could be, for example, a LAN drive or an NFS-mounted drive. The file access permissions must permit access to the file by the DB2 UDB instance owner.

FORMAT

The FORMAT function does one of the following:

- Returns the document format specified in a handle
- Changes the format specification in a document's handle, and returns the changed handle.

The returned document format is a string of type VARCHAR(30). The returned handle is of type DB2TEXTFH or DB2TEXTH.

This is the format parameter used for indexing the corresponding text document. The document formats supported are listed in "Which document formats are supported" on page 24.

Function syntax

```
            (1)
►►—FORMAT———(—handle—)————————————————————————◄
```

Notes:

1   Returns a format value, type VARCHAR(30).

```
            (1)
►►—FORMAT———(—handle—,—format——)——————————————————◄
```

Notes:

1   Returns a handle, type DB2TEXTFH or DB2TEXTH.

Function parameters

handle
> An expression whose result is a value of type DB2TEXTFH or DB2TEXTH. It is usually the name of a handle column from which the format setting is to be returned or set.

format
> The new document format setting of data type VARCHAR(30).
>
> If *format* is specified, this document format is set in the handle; in this case, the handle is returned instead of the format setting.

HANDLE

The HANDLE function returns a handle of type DB2TEXTFH or DB2TEXTH, selected by sequence number from a list of handles.

Use the HANDLE function only in an INSERT statement.

> **Tip**
>
> This function is a feature of an earlier release of DB2 Text Extender. For compatibility reasons it continues to be supported, Its functionality has been superceded by the SEARCH_RESULT search function.

Function syntax

```
►►──HANDLE──(──handle-list──,──integer──)──────────────────────────────►◄
```

Function parameters

handle-list
An expression whose result is a value of type DB2TEXTHLISTP or DB2TEXTFHLISTP. It is returned by the function HANDLE_LIST.

This is a pointer to a list of handles of documents found by a search.

integer
An INTEGER value indicating which handle in the list is to be returned.

HANDLE_LIST

The HANDLE_LIST function searches for text documents using a search argument and returns a value of type DB2TEXTHLISTP or DB2TEXTFHLISTP that points to a list of handles for the found documents.

Note: The pointer to this list is accessible only within the scope of the SQL statement using this function.

If no text documents are found containing the search term(s), the list of handles is empty. Use the NO_OF_DOCUMENTS function to find whether the list is empty.

If you are searching for text stored in a column of a base table, use this function only once per statement. If you are searching for text stored in a column of a view, and the view is made from more than one base table (that is, it is a union view), use one handle for each handle column as the *handle* input argument to the HANDLE_LIST function.

Tip

This function is a feature of an earlier release of DB2 Text Extender. For compatibility reasons it continues to be supported, Its functionality has been superceded by the SEARCH_RESULT search function.

Function syntax

```
►►──HANDLE_LIST──(──handle──,──search-argument────)──────────────────────►◄
```

Function parameters

handle

An expression whose result is a value of type DB2TEXTH or DB2TEXTFH. It is usually the handle column containing the handles of the text documents to be searched. It can also be a prototype handle extracted from the TEXTCOLUMNS catalog view.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.

INIT_TEXT_HANDLE

The INIT_TEXT_HANDLE function returns a partially initialized handle that contains preset values for the text's format or language. It can be inserted into a handle column. This is useful when you add a row containing text whose format and language are not the same as those specified in the text configuration settings.

The returned handle is a value of type DB2TEXTH.

If you intend to search in text that is stored in *external files* rather than in a DB2 UDB table, you can use the INIT_TEXT_HANDLE function to return a completely initialized handle that contains preset values for the text's CCSID, format, language, and the location of the file.

The returned handle is a value of type DB2TEXTFH.

Use the INIT_TEXT_HANDLE function to insert or update handle values.

Function syntax

```
►►──INIT_TEXT_HANDLE──(──format──,──language──)──────────────────────►◄
```

```
►►──INIT_TEXT_HANDLE──(──CCSID──,──format──,──language──,──file-name──)───────►◄
```

Function parameters

format
> A string of type VARCHAR(30) specifying the new document format setting. The formats supported are listed in "Which document formats are supported" on page 24.

language
> A string of type VARCHAR(30) specifying the new document language setting. The supported languages are listed in Table 5 on page 43.

file-name
> A string of type VARCHAR(150) specifying the absolute path and file name of the external file that is to be associated with the handle. To have access to the file in UNIX, the DB2 UDB instance owner must be included in the file access permissions. For OS/2 and Windows users, the file access permissions must include the logon user IDs.

LANGUAGE

The LANGUAGE function does one of the following:

- Returns the document language specified in a handle
- Changes the language specification in a document's handle, and returns the changed handle.

The returned document language is a string of type VARCHAR(30). The returned handle is of type DB2TEXTFH or DB2TEXTH.

This is the language parameter used for indexing the corresponding text document. The supported languages are listed in Table 5 on page 43.

Function syntax

```
                     (1)
►►──LANGUAGE─────────(─handle─)──────────────────────────────────────►◄
```

Notes:

1   Returns a language value, type VARCHAR(30).

```
                     (1)
►►──LANGUAGE─────────(─handle─,─language──)──────────────────────────►◄
```

Notes:

1   Returns a handle, type DB2TEXTFH or DB2TEXTH.

Function parameters

handle

An expression whose result is a value of type DB2TEXTFH or DB2TEXTH. It is usually the name of a handle column from which the language setting is to be returned or set.

language

The new document language setting of data type VARCHAR(30).

If *language* is specified, this document language is set in the handle; the handle is returned instead of the language setting.

NO_OF_DOCUMENTS

The NO_OF_DOCUMENTS function returns an INTEGER value indicating the number of items in a list of text documents found by a search. The returned value is the number of entries found in a list of handles.

> **Tip**
> This function is a feature of an earlier release of DB2 Text Extender. For compatibility reasons it continues to be supported, Its functionality has been superceded by the SEARCH_RESULT search function.

Function syntax

►►──NO_OF_DOCUMENTS──(──*handle-list*──)──────────────────────────────►◄

Function parameters

handle-list

An expression whose result is a value of type DB2TEXTHLISTP or DB2TEXTFHLISTP. It is returned by the function HANDLE_LIST.

This is a pointer to a list of handles of documents found by a search.

The HANDLE_LIST and NO_OF_DOCUMENTS functions must be in the same SQL statement because the list exists only within the scope of the statement.

NO_OF_MATCHES

NO_OF_MATCHES can search in text documents and return an INTEGER value indicating how many matches resulted per document.

Function syntax

►►──NO_OF_MATCHES──(──*handle*──,──*search-argument*──)──────────────────────────►◄

Function parameters

handle

An expression whose result is a value of type DB2TEXTFH or DB2TEXTH. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.

RANK

RANK can search in text documents and return a rank value for each document found, indicating how well the found document is described by the search argument.

RANK returns an DOUBLE value between 0 and 1. The rank value is absolute, indicating how well the found document satisfies the search criteria in relation to other found documents. The value indicates the number of matches found in the document in relation to the document's size.

Function syntax

```
►►──RANK──(──handle──,──search-argument──)──────────────────────────►◄
```

Function parameters

handle

An expression whose result is a value of type DB2TEXTFH or DB2TEXTH. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.

REFINE

The REFINE function takes two search arguments and returns a combined search argument of type LONG VARCHAR, consisting of the two original search arguments connected by the Boolean operator AND.

Function syntax

►►──REFINE──(──*search-argument*──,──*search-argument*──)──────────────────────►◄

Function parameters

search-argument
> A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.
>
> The search argument must not contain the search parameters IS ABOUT, THESAURUS, or EXPAND.

SEARCH_RESULT

The SEARCH_RESULT function returns the result of a search in an intermediate table. This function can be used in a FROM clause of an SQL statement.

The returned table has the following structure:

```
Column Name         Data Type

HANDLE              DB2TX.DB2TEXTH, DB2TX.DB2TEXTFH
NUMBER_OF_MATCHES   INTEGER
RANK                DOUBLE
```

Values are generated only for the selected columns of the intermediate table. Select count(*) generates the HANDLE column only. Because the calculation of the rank values consumes a lot of system resources, you should not select the rank value from the intermediate table if the rank value is not required.

This UDF is faster than CONTAINS or RANK when processing large tables.

Function syntax

►►—SEARCH_RESULT—(—*schema*—,—*table*—,—*handle*—,—*search-argument*—)—————►◄

Function parameters

schema
　　The schema name of the table you are searching in.

table　The name of the table you are searching in.

handle
　　The name of a handle column that corresponds to the column containing the documents to be searched.

search-argument
　　A string of type LONG VARCHAR containing the terms to be searched for. See "Chapter 12. Syntax of search arguments" on page 187.

Examples

For an example, refer to "Improving search performance" on page 81, or look at the sample DB2 Text Extender functions provided in the file described in "Where to find syntax examples of search functions" on page 63.

SEARCH_RESULT function

Chapter 12. Syntax of search arguments

A search argument is the condition that you specify when searching for terms in text documents. It consists of one or several search terms and search parameters.

Examples of search arguments are given in "Specifying search arguments" on page 69, and in a file called `txsample.udf`. It contains examples of DB2 Text Extender functions that run against the sample table described in "Preparing a sample database for installation verification" on page 12.

The DB2 Text Extender 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. Otherwise, it returns 0.
- NO_OF_MATCHES. This function uses a search argument to search in text documents. It returns an INTEGER value indicating how many matches resulted per document.
- RANK. 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.
- REFINE. This function takes two search arguments and returns a combined search argument of type LONG VARCHAR, consisting of the two original search arguments connected by the Boolean operator AND.
- HANDLE_LIST. This function uses a search argument to search in text documents. It returns a value of type DB2TEXTHLISTP or DB2TEXTFHLISTP that points to a list of handles for the found documents.
- SEARCH_RESULT. This function returns a table containing the requested information, that is, the rank, number of matches, and the handle.

The API functions that use search arguments are:
- DesGetBrowseInfo. This function uses a search argument for searching through text identified by a handle. It returns a pointer to browse information needed by DesStartBrowseSession for highlighting terms.
- DesGetSearchResultTable. This function uses a search argument for searching through text documents identified by a text column. The handle data of the found text items is written to a result table. Browse information about rank and the number of matches can also be written to the result table.

Search argument
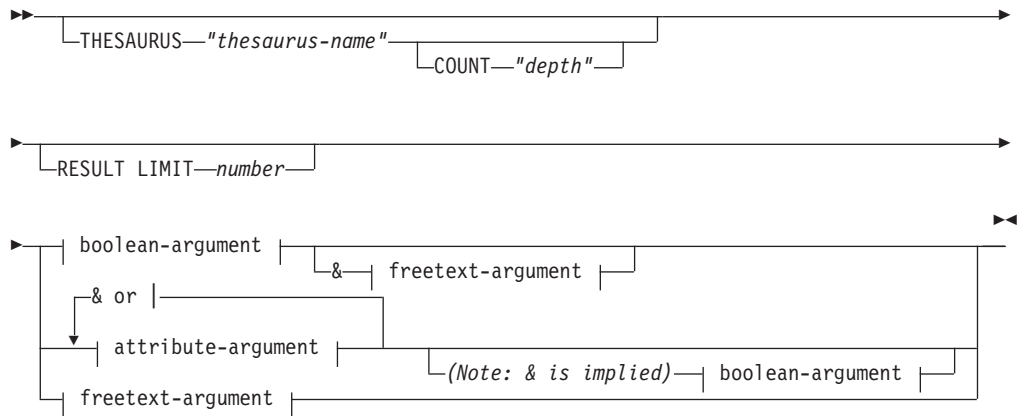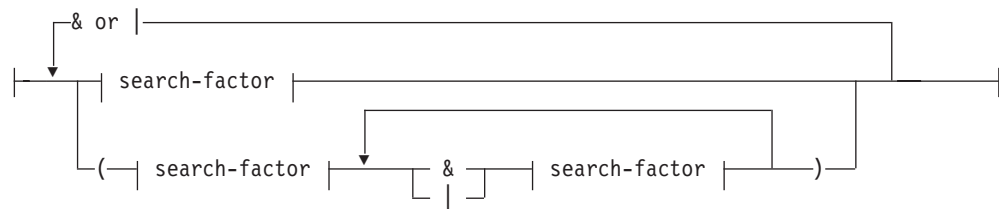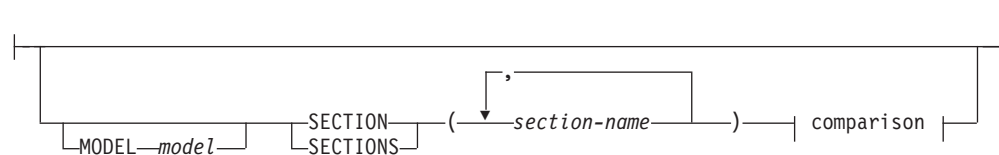
Search argument syntax

```
►►──┬─────────────────────────────────────────────────┬──────────►
    └─THESAURUS─"thesaurus-name"─┬──────────────────┬──┘
                                 └─COUNT─"depth"────┘


►──┬──────────────────────────┬────────────────────────────────►
   └─RESULT LIMIT─number───────┘


►──┬─ boolean-argument ─┬──────────────────────────────────►◄
   │        └─&─┬ freetext-argument ┤─┘
   │─& or │─────────────────────────┤
   │ ├─ attribute-argument ┤─────────┤
   │              └─(Note: & is implied)─ boolean-argument ┤─┘
   └─ freetext-argument ┤
```

boolean-argument:

```
        ┌─& or │────────────────────────────────────┐
├──┬──┬─ search-factor ┤──────────────────────────┬──┤
   │  └─(─┤ search-factor ├─┬─┬─&─┬─┤ search-factor ├─┬─)─┘
   │                       └─┤ │ ├─┘
```

attribute-argument:

```
                                                          │
├──┬───────────────────────────────────────────────────┬──┤
   └─┬──────────────┬─┬─SECTION──┬─(─┬─section-name─┬─)─ comparison ┤
     └─MODEL─model──┘ └─SECTIONS─┘   └───────,──────┘
```

Comparison:

```
├──┬────────────────────────────────┬──────────────────────┤
   └─=─≥─≤─>─<─attribute-value───────┘
```

freetext-argument:

```
├─IS ABOUT─┬──────────────────┬─┬──────────┬─"phrase-or-sentence"──►
           ├─SYNONYM FORM OF──┤ └─language─┘
           ├─feature──────────┤
           └─ thesaurus ──────┘
```

```
►──┬──────────────────────────────┬──────────────────────────────┤
   └─ESCAPE──"escape-char"──┘
```

search-factor:

```
├──┬──────────────────────────────────────────────────────┬──►
   │                        ┌─────,──────┐                  │
   └─┬─────────────┬─┬─SECTION──┬─(─▼─section-name─┴─)──┘
     └─MODEL─model─┘ └─SECTIONS─┘
```

```
►─┤ search-element ├───────────────────────────────────────┤
```
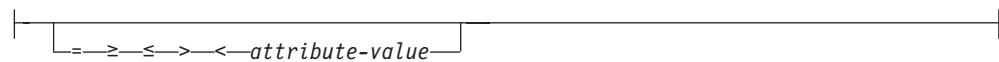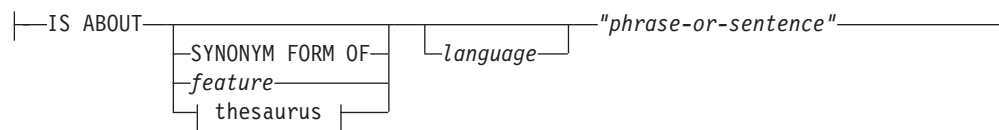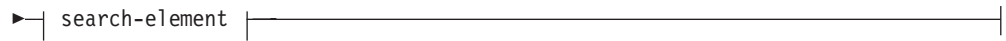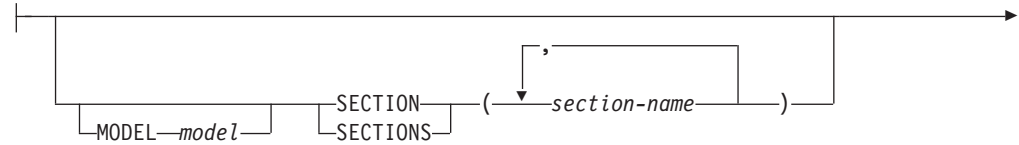
search-element:

```
├─┬──────┬─┤ search-primary ├──────────────────────────────────────┤
│ └─NOT─┘                                        ┌─AND─┐
│                                                │     │
└─┤ s.-primary ├─┬─IN SAME PARAGRAPH AS─┬─▼─┤ s.-primary ├─┘
                 └─IN SAME SENTENCE  AS─┘
```
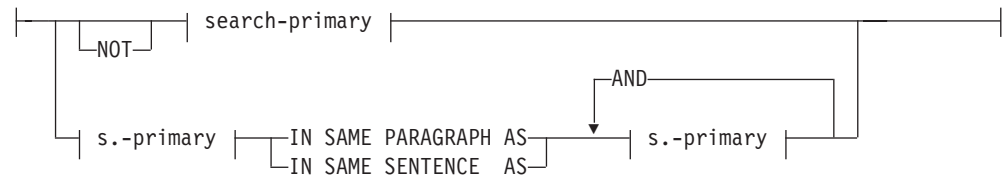
search-primary:

```
├─┬─┤ search-atom ├──────────────────────────┬──┤
  │        ┌────,─────┐                       │
  └─(─▼─┤ search-atom ├─┴─)──────┘
```

search-atom:

```
├─┬─────────────────────────┬─┬───────────┬─"word-or-phrase"──►
  ├─PRECISE FORM OF──────────┤ └─language─┘
  ├─STEMMED FORM OF──────────┤
  ├─FUZZY FORM OF─match-level─┤
  ├─SYNONYM FORM OF──────────┤
  ├─BOUND────────────────────┤
  ├─SOUNDS LIKE──────────────┤
  ├─feature──────────────────┤
  └─┤ thesaurus ├────────────┘
```

```
►─┬──────────────────────────────────────┬──────────────┤
  └─ESCAPE──"escape-character"──┘
```
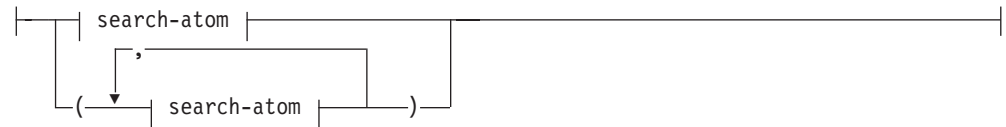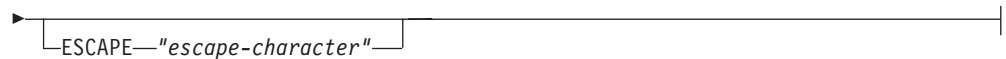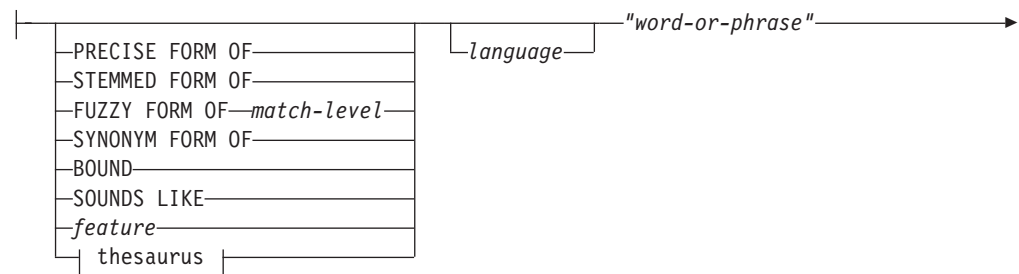
thesaurus (if THESAURUS is specified):

```
├──────────────────────────┬─TERM OF─────────────────────────────────────────┤
        └─EXPAND──"relation"─┘
```
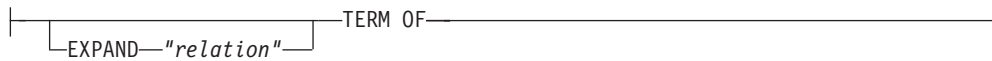
Examples

Examples are given in "Specifying search arguments" on page 69.

Search parameters

IS ABOUT

An option that lets you specify a free-text search argument, that is, a natural-language phrase or sentence that describes the concept to be found. See "Free-text and hybrid search" on page 75.

MODEL *model*

A keyword used to specify the name of the document model to be used in the search term. The document model describes the structure of documents that contain identifyable sections so that the content of these sections can be searched individually.

The model name must be specified in a document models file described in "Working with structured documents (section support)" on page 38. The model name can be masked using wildcard characters.

If you do not specify a model, the default model specified during index creation is used.

SECTION(S) *section-name*

A keyword used to specify one or more sections that the search is to be restricted to. The section name must be specified in a model in a document models file, described in "Working with structured documents (section support)" on page 38. A section name can be masked using wildcard characters % and _. Note that for Ngram indexes, the wild card search is only allowed for section names that are not nested.

Sections can be nested within other sections, for example:

```
play/Act/Title=play/act/title
```

Restrictions: Searching in nested sections is possible only for documents stored in columns enabled with format XML. For Ngram indexes, only one section name can be searched. Note that the default Boolean operation for multiple sections is 1.

attribute-value

A value used together with a preceding comparison operator for the attributes listed in the preceding section list. A query requesting an attribute's value to be within a certain range can use two comparison operators within one attribute condition.

A combination of operators using the same kind of comparison, like ≥ in the first and > in the second) of the same condition is invalid. Specification of two comparisons with = operator is also invalid.

=   Requests an equality comparison of the attribute in the indexed document with the following attribute value.

≥   Requests a "greater than or equal to" comparison of the attribute in the indexed document with the following attribute value.

> Requests a "greater than" comparison of the attribute in the indexed document with the following attribute value.

≤ Requests a "less than or equal to" comparison of the attribute in the indexed document with the following attribute value.

< Requests a "less than" comparison of the attribute in the indexed document with the following attribute value.

Sections can be nested within other sections, for example:

`play/Act/Title=play/act/title`

Restrictions: Searching in nested sections is possible only for documents stored in columns enabled with format XML. For Ngram indexes, only one section name can be searched and XML format is not supported. For date operations, the year "1" is treated less than the year 2001.

THESAURUS *thesaurus-name*

A keyword used to specify the name of the thesaurus to be used to expand the search term. The thesaurus name is the file name (without its extension) of a thesaurus that has been compiled using the thesaurus compiler `TXTHESC` or `TXTHESN`. There are default thesauri `desthes` and `desnthes`, stored in the sample directory, where `desnthes` is an Ngram thesaurus. You can also specify the file's path name. The default path name is the dictionary path.

The maximum length of the thesaurus name (including the path) is 120 characters.

COUNT *depth*

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.

RESULT LIMIT *number*

A keyword used to specify the maximum number of entries to be returned in the result list. *number* is a value from 1 to 32767. If a free-text search is used, the search result list is ranked only with respect to the complete search result list. Otherwise, the limited search result is ranked only from the entries of the list.

EXPAND *relation*

A keyword used to specify the relation, such as INSTANCE, between the search term specified in TERM OF and the thesaurus terms to be used to expand the search term. The relation name must correspond to a relation used in the thesaurus. See "Thesaurus concepts" on page 231.

For an Ngram thesaurus, use the member-relation name described in "Creating an Ngram thesaurus" on page 238. For user-defined member relations, use `:RELATION` *n* where *n* is the member relation number specified in `:RELATED (`*number*`)`.

TERM OF *"word-or-phrase"*

The search term, or multi-word search term, to which other search terms are to be added from the thesaurus.

search-factor

An operand that can be combined with other operands to form a search

argument. The evaluation order is from left to right. The logical AND (&)
operator binds stronger than the logical OR (|) operator. Example:

```
"passenger" & "vehicle" | "transport" & "public"
```

is evaluated as:

```
("passenger" & "vehicle") | ("transport" & "public")
```

To search for:

```
"passenger" & ("vehicle" | "transport") & "public"
```

you must include the parentheses as shown.

NOT search-primary

An operator that lets you exclude text documents from your search that
contain a particular term.

When NOT is used in a search factor, you cannot use the SYNONYM
FORM OF keyword.

search-primary IN SAME PARAGRAPH AS search-primary

A keyword that lets you search for a combination of terms occurring in the
same paragraph.

The following search argument finds text documents containing the term
"traffic" only if the term "air" is in the same paragraph.

```
"traffic" IN SAME PARAGRAPH AS "air"
```

You cannot use the IN SAME PARAGRAPH AS keyword when NOT is
used in a search factor.

search-primary IN SAME SENTENCE AS search-primary

A keyword that lets you search for a combination of terms occurring in the
same sentence. Similar to IN SAME PARAGRAPH AS.

AND search-primary

A keyword that lets you combine several search-primaries to be searched
for in the same sentence or the same paragraph.

The following search argument searches for "forest", "rain", "erosion", and
"land" in the same sentence.

```
"forest" IN SAME SENTENCE AS "rain" AND "erosion" AND "land"
```

search-atom

If you connect a series of search atoms by commas, then a search is
successful if a term in any one of the search atoms is found. Each search
atom must contain at least a word or a phrase.

The following statement is true if one or more of the search arguments is
found.

```
CONTAINS (mytexthandle, '( "text",
                          "graphic",
                          "audio",
                          "video")') = 1
```

**PRECISE FORM OF, STEMMED FORM OF, FUZZY FORM OF, SYNONYM
FORM OF, BOUND**

Table 8 on page 194 shows the options that correspond to the various types
of index. For example, for a linguistic index, any of the options are suitable
except for PRECISE FORM OF. If you specify PRECISE FORM OF, it is
ignored and the default value is taken.

The search term processing is described in more detail in Table 9 on
page 194 .

Syntax of search arguments

Table 8. Linguistic options

Search atom keyword	Index type				
	Linguistic	Precise	Precise Normalized	Ngram	Ngram case-enabled
PRECISE FORM OF		X	X		O
STEMMED FORM OF	X			O	O
FUZZY FORM OF				O	O
IS ABOUT	O	O	O		
SYNONYM FORM OF	O	O	O		
EXPAND	O	O	O		
SOUNDS LIKE	O	O	O		
IN SAME SENTENCE AS	O	O	O	O	O
IN SAME PARAGRAPH AS	O	O	O	O	O
BOUND				O	O
X=default setting O=function available					

Table 9. Search term options for Ngram indexes

Search atom keyword	Search term processing				
	Case		Stemming	Match	
	Sensitive	Insensitive		Exact	Fuzzy
PRECISE FORM OF	when case-enabled	X		X	
STEMMED FORM OF		X	X		
FUZZY FORM OF		X			X
X=default setting					

If you use a keyword that is not available for that index type, it is ignored and either the default keyword is used instead, or a message is returned.

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, rather than being first reduced to its stem form. For precise indexes, 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".

This is the default option for precise indexes. For a precise normalized index, the default form of search is not case-sensitive. If you specify this keyword for a linguistic index, it is ignored and STEMMED FORM OF is assumed.

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.

Example: `programming computer systems` is replaced by `program compute system` when you use the US-English dictionary, and by `programme compute system` when you use the UK-English dictionary.

This search phrase can find "programmer computes system", "program computing systems", "programming computer system", and so on.

This is the default option for linguistic indexes. If you specify this keyword for a precise index, it is ignored and PRECISE FORM OF is assumed instead.

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`. *match-level*: An integer from 1 to 5 specifying the degree of similarity, where 5 is more similar than 1.

SYNONYM FORM OF

A keyword that causes the word or phrase following SYNONYM FORM OF to be searched for together with its synonyms. The synonyms are provided by the dictionary specified by *language* or else by the default dictionary.

Synonyms for a phrase are alternative phrases containing all the possible combinations of synonyms that can be obtained by replacing each word of the original phrase by one of its synonyms. The word sequence remains as in the original phrase.

If you specify this keyword for a precise index, it is ignored and PRECISE FORM OF is assumed instead.

You cannot specify this keyword when NOT is used in the search factor, or when the word or phrase to be searched for contains masking characters.

BOUND

A keyword for searching in documents that use the Korean CCSID. It causes the search to respect word phrase boundaries. If *language* is specified, it is ignored; Korean is assumed.

language

A variable that determines which dictionary is used in linguistic processing of text documents during indexing and retrieval. This applies not only to linguistic indexes, but also to precise indexes because these use a dictionary to process stop words.

Linguistic processing includes synonym processing and word-stem processing.

The supported languages are listed in Table 5 on page 43.

Syntax of search arguments

Note: When searching in documents that are not in U.S. English, you must specify the language in the search argument *regardless of the default language*.

"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.

Precise or linguistic search. DB2 Text Extender can search using either the precise form of the word or phrase, or a variation of it. If you do not specify one of the options in Table 8 on page 194, the default linguistic options are used according to which type of index is being used.

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"
```

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 SYNONYM FORM OF, *feature*, or THESAURUS.

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.

> Example: If *escape-character* is $, then $%, $_, and $$ represent %, _, and $ respectively. Any % and _ characters not preceded by $ represent masking characters. Also note that the escape character can only be a single character.

Summary of rules and restrictions:

Boolean operations
> NOT is not allowed after OR.

FUZZY FORM OF
> The first 3 characters must match. Cannot be used if a word in the search atom contains a masking character. Cannot be used in combination with NOT. Can be used only with an Ngram index.

IN SAME PARAGRAPH AS
> Cannot be used if NOT is used in a search factor.

IN SAME SENTENCE AS
> Cannot be used if NOT is used in a search factor.

Linguistic index
> Prevents the use of PRECISE FORM OF. Takes as default STEMMED FORM OF. Masking characters can be used. Searches are case-insensitive.

Masking character
> Prevents the use of SYNONYM FORM OF, and THESAURUS.

Ngram index
> Masking characters can be used, although not following a non-alphanumeric character. Searches are case-insensitive unless the index is case-enabled and PRECISE FORM OF is used.

NOT Prevents the use of SYNONYM FORM OF, IN SAME PARAGRAPH AS, and IN SAME SENTENCE AS.

PRECISE FORM OF
> Ignored for a linguistic index.

Precise index
> Prevents the use of STEMMED FORM OF, and SYNONYM FORM OF. Takes as default PRECISE FORM OF. Masking characters can be used. Searches are case-sensitive.

Syntax of search arguments

STEMMED FORM OF

Ignored for a precise index, but available for a normalized precise index containing English documents.

SYNONYM FORM OF

Cannot be used if a word in the search atom contains a masking character. Cannot be used in combination with NOT. Cannot be used with a precise index.

Chapter 13. API functions for searching and browsing

DB2 Text Extender provides C functions for searching for text documents, and for browsing (displaying) the found documents. These functions constitute the DB2 Text Extender application programming interface (API). This chapter describes the API functions in alphabetical order.

"Chapter 8. Using the API functions for searching and browsing" on page 97 provides an introduction to the functions, and describes how they can be used together.

Function	Purpose	Page
DesCloseDocument	Releases the storage allocated by DesOpenDocument.	200
DesEndBrowseSession	Closes a browse session and releases the storage allocated by DesStartBrowseSession.	201
DesFreeBrowseInfo	Releases the storage allocated by DesGetBrowseInfo.	202
DesGetBrowseInfo	Searches in the document for text using a search argument, and creates browse information.	203
DesGetMatches	Returns a pointer to highlighting information for the text document described by a document handle. The highlighting information is a data stream. It comprises the text context (at least one paragraph) and information for highlighting text in that context.	206
DesGetSearchResultTable	Takes a search argument to search for text documents in a given text column, and stores the result in a user-provided table. It can also return browse information.	211
DesOpenDocument	Receives a browse session pointer, a handle, and an option DES_FAST or DES_EXTENDED indicating the type of linguistic processing to be used for highlighting found terms. Prepares the text document that corresponds to the handle to get the document text and highlighting information, and returns a document handle that is used for iteratively calling DesGetMatches.	215
DesStartBrowseSession	Opens a browse session using the browse information from DesGetBrowseInfo, and returns a browse session handle for use by the other browse functions.	217

> **Tip**
>
> Many of the API functions need a connection handle (hdbc). You must provide this handle using the SQLConnect function, but this does not prevent you from calling DB2 Text Extender from embedded SQL programs. The *DB2 Call Level Interface Guide and Reference* describes how to mix CLI statements with embedded SQL statements.

DesCloseDocument

Purpose

Closes a text document opened by DesOpenDocument, and releases the storage allocated during the return of document text and highlighting information.

Syntax

```
DESRETURN
  DesCloseDocument
      (DESBROWSESESSION      BrowseSession,
       DESHANDLE             DocumentHandle);
```

Function arguments

Table 10. DesCloseDocument arguments

Data Type	Argument	Use	Description
DESBROWSESESSION	*BrowseSession*	input	Browse session handle.
DESHANDLE	*DocumentHandle*	input	Handle returned by DesOpenDocument identifying an opened text document.

Return codes

RC_SUCCESS

RC_INVALID_PARAMETER
RC_INVALID_SESSION
RC_SE_INCORRECT_HANDLE
RC_SE_IO_PROBLEM
RC_SE_LS_FUNCTION_FAILED
RC_SE_NOT_ENOUGH_MEMORY
RC_SE_REQUEST_IN_PROGRESS
RC_SE_WRITE_TO_DISK_ERROR

Restrictions

This function can be called only after you have opened a text document by calling DesOpenDocument.

DesEndBrowseSession

Purpose

Ends a browse session started by DesStartBrowseSession and releases the storage allocated for the browse session.

Syntax

```
DESRETURN
  DesEndBrowseSession
     (DESBROWSESESSION      BrowseSession);
```

Function arguments

Table 11. DesEndBrowseSession arguments

Data Type	Argument	Use	Description
DESBROWSESESSION	*BrowseSession*	input	Browse session handle.

Usage

This function does not release the storage allocated for the browse session by DesGetBrowseInfo. This storage contains browse information that can still be used for another browse session. To release this storage, call DesFreeBrowseInfo.

Return codes

RC_SUCCESS

RC_INVALID_SESSION
RC_INVALID_PARAMETER
RC_SE_UNEXPECTED_ERROR

Restrictions

This function can be called only after you have started a browse session by calling DesStartBrowseSession.

DesFreeBrowseInfo

Purpose

Frees the storage allocated for the browse information by DesGetBrowseInfo.

Syntax

```
DESRETURN
  DesFreeBrowseInfo
     (DESBROWSEINFO        BrowseInfo);
```

Function arguments

Table 12. DesFreeBrowseInfo arguments

Data Type	Argument	Use	Description
DESBROWSEINFO	*BrowseInfo*	input	Browse information

Return codes

RC_SUCCESS

RC_INVALID_PARAMETER

Restrictions

This function can be called only after you have allocated storage for browsing information by calling DesGetBrowseInfo.

DesGetBrowseInfo

Purpose

Receives a search argument for searching through text identified by a handle. It returns a pointer to browse information needed by DesStartBrowseSession for highlighting the found terms.

Syntax

```
DESRETURN
  DesGetBrowseInfo
    (SQLHDBC            hdbc,
     SQLCHAR            *pHandle,
     DESUSHORT          HandleLength,
     char               *pSearchArgument,
     DESSMALLINT        ArgumentLength,
     DESBROWSEINFO      *pBrowseInfo,
     DESMESSAGE         *pErrorMessage);
```

Function arguments

Table 13. DesGetBrowseInfo arguments

Data Type	Argument	Use	Description
SQLHDBC	*hdbc*	input	A database connection handle.
SQLCHAR *	*pHandle*	input	Pointer to a handle that has been extracted from the database.
DESUSHORT	*HandleLength*	input	Length of pHandle. DES_NTS cannot be used here.
char *	*pSearchArgument*	input	Pointer to the text search argument that specifies the information that you want to find.
DESSMALLINT	*ArgumentLength*	input	Either the length of pSearchArgument (not including a null byte terminator), or DES_NTS.
DESBROWSEINFO *	*pBrowseInfo*	output	Pointer to browse information containing the data needed to browse a text document. This pointer is passed to DesStartBrowseSession.
DESMESSAGE *	*pErrorMessage*	output	Implementation-defined message text. If an error occurs, DB2 Text Extender returns an error code and an error message. The application program allocates the buffer of size DES_MAX_MESSAGE_LENGTH. If *pErrorMessage* is the null pointer, no error message is returned.

Usage

Your application program must establish a connection to the database before it calls DesGetBrowseInfo.

For the pointer to the search argument, *char** is used rather than *SQLCHAR**. This is because it is possible that the parameter value may not come from the database.

DesGetBrowseInfo API function

For the mapping between the SQL data types and C data types, you must use the SQL symbolic name SQL_VARBINARY for a handle. The type of host variables pointing to the C representation of *Handle* values is SQLCHAR*.

DB2 Text Extender allocates storage for the browse information. The application program must free the storage and related resources by calling DesFreeBrowseInfo.

Because *Handle* values are bit data and contain several '\0' characters, you must specify the length of *pHandle*.

The search argument in pSearchArgument is described in "Chapter 12. Syntax of search arguments" on page 187.

Return codes

RC_SUCCESS
RC_NO_BROWSE_INFO

RC_ALLOCATION_ERROR
RC_FILE_IO_PROBLEM
RC_INTERNAL_ERROR
RC_INVALID_PARAMETER
RC_PARSER_INVALID_ESCAPE_CHARACTER
RC_PARSER_INVALID_USE_OF_ESCAPE_CHAR
RC_PARSER_SYNTAX_ERROR
RC_SE_COMMUNICATION_PROBLEM
RC_SE_EMPTY_INDEX
RC_SE_EMPTY_QUERY
RC_SE_FUNCTION_DISABLED
RC_SE_FUNCTION_IN_ERROR
RC_SE_INCORRECT_HANDLE
RC_SE_INDEX_DELETED
RC_SE_INDEX_NOT_ACCESSIBLE
RC_SE_INDEX_SUSPENDED
RC_SE_INSTALLATION_PROBLEM
RC_SE_IO_PROBLEM
RC_SE_MAX_NUMBER_OF_BUSY_INDEXES
RC_SE_MAX_OUTPUT_SIZE_EXCEEDED
RC_SE_NOT_ENOUGH_MEMORY
RC_SE_PROCESSING_LIMIT_EXCEEDED
RC_SE_QUERY_TOO_COMPLEX
RC_SE_SERVER_BUSY
RC_SE_SERVER_CONNECTION_LOST
RC_SE_SERVER_NOT_AVAILABLE
RC_SE_UNEXPECTED_ERROR
RC_SE_UNKNOWN_INDEX_NAME
RC_SE_UNKNOWN_SERVER_NAME
RC_SE_WRITE_TO_DISK_ERROR

Warnings: The following return codes indicate that the function has returned a result, but it may not be as expected.

RC_SE_CONFLICT_WITH_INDEX_TYPE
RC_SE_DICTIONARY_NOT_FOUND
RC_SE_STOPWORD_IGNORED
RC_SE_UNKNOWN_SECTION_NAME
RC_SE_DOCMOD_READ_PROBLEM

Restrictions

This function can be called only after you have made a connection to a database and used a DB2 Text Extender function to extract a handle from the database.

DesGetMatches

Purpose

Returns a data stream containing highlighting information for the text document described by a document handle. See "Data stream syntax". The highlight information comprises the text context (at least one paragraph) and information for highlighting text in that context.

DesGetMatches returns only a portion of the data stream, indicating the length of the portion in the output structure.

A sequence of calls to DesGetMatches gets the entire text document content. When the end of the text document is reached, RC_SE_END_OF_INFORMATION is returned.

Syntax

```
DESRETURN
  DesGetMatches
    (DESBROWSESESSION     BrowseSession,
     DESHANDLE            DocumentHandle,
     DESMATCHINFO         *pMatchInfo,
     DESULONG             *pMatchInfoLength,
     DESMESSAGE           *pErrorMessage);
```

Function arguments

Table 14. DesGetMatches arguments

Data Type	Argument	Use	Description
DESBROWSESESSION	*BrowseSession*	input	Browse session handle.
DESHANDLE	*DocumentHandle*	input	Document handle returned by DesOpenDocument.
DESMATCHINFO *	*pMatchInfo*	output	Pointer to a buffer containing the data stream portion received. DesGetMatches allocates that buffer.
DESULONG *	*pMatchInfoLength*	output	The length of the data stream portion pointed to by pMatchInfo.
DESMESSAGE *	*pErrorMessage*	output	Implementation-defined message text. If an error occurs, DB2 Text Extender returns an error code and an error message. The application program allocates the buffer of size DES_MAX_MESSAGE_LENGTH. If *pErrorMessage* is the null pointer, no error message is returned.

Data stream syntax

```
►►── ──05── ──DB2TX_DOC── ──DB2TX_START── ──────────────────────────►
```

```
      ►─┬────────────────────────────────────────────────────┬─►
        └─ ─ll── ─DB2TX_DNAM── ─DB2TX_ATOMIC── ─document_name─┘
```

```
         ┌─◄──────────────────┐
      ►─┬─┤ Section ├──────────┴───05── ─DB2TX_DOC── ─DB2TX_END── ──────────►◄
```

Section:

```
      ├── ─05── ─DB2TX_DEL── ─DB2TX_START── ─────────────────────────────────►
```

```
      ►─┬──────────────────────────────────────────────────┬─►
        └─ ─ll── ─DB2TX_SNAM── ─DB2TX_ATOMIC── ─section_name─┘
```

```
         ┌─◄───────────────────────────────────┐
         │                  ┌─◄─────────────┐   │
      ►─┬─┤ Text encoding ├──┴─┤ Paragraph ├─┴──┴──────────────────────────────►
```

```
      ►── ─05── ─DB2TX_DEL── ─DB2TX_END── ──────────────────────────────────┤
```

Text encoding:

```
      ├────07── ─DB2TX_CCSID── ─DB2TX_ATOMIC── ─coded_character_set_identifier── ──►
```

```
      ►── ─07── ─DB2TX_LANG── ─DB2TX_ATOMIC── ─language_identifier── ───────────┤
```

Paragraph:

```
                                              ┌─◄─────────────────┐
      ├── ─05── ─DB2TX_PAR── ─DB2TX_START──────┴─┤ Paragraph text ├─┴────────────►
```

```
      ►── ─05── ─DB2TX_PAR── ─DB2TX_END── ──────────────────────────────────────┤
```

Paragraph text:

```
      ├──┬─ ─ll── ─DB2TX_TEXT── ─DB2TX_ATOMIC── ─text_unit─ ─┬─────────────────────►
         └─ ─ll── ─DB2TX_LINK── ─DB2TX_ATOMIC── ─media_ref─ ─┘
```
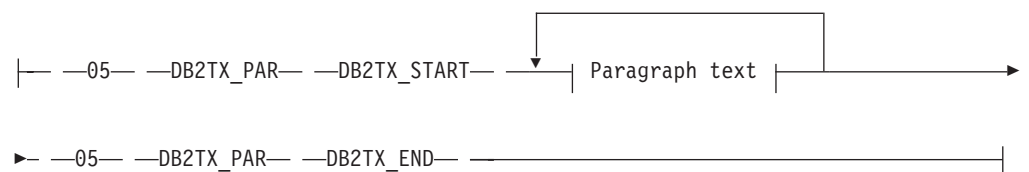
DesGetMatches API function

```
         ┌──────────────────────────────────────┐
         │                                      │
▶──────────┴──┬──05──┬──DB2TX_NL──┬──DB2TX_ATOMIC──┬──┴──────────────────▶
```

```
         ┌────────────────────────────────────────────────┐
         │                                                │
▶──────────┴──┬──ll──┬──DB2TX_MATCH──┬──DB2TX_ATOMIC──┬──match_information──┬──┴──┤
```
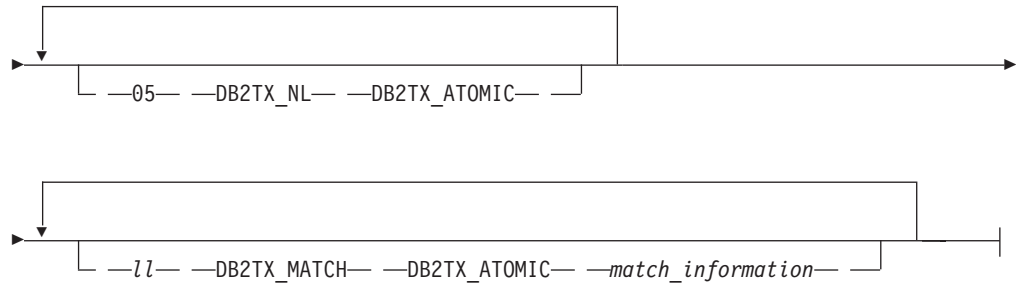
Each segment in the syntax diagram, such as 05 DB2TX_DOC DB2TX_START begins with a length field of type integer, which in the diagram is either an explicit number, such as 05, or a variable *ll*. The length of the segment includes the 2-byte length field.

Note: The length is in big-endian format.

Each segment includes one of the following 1-byte type identifiers:

DB2TX_START
Indicates the start of a segment, such as a document or a paragraph.

DB2TX_END
Indicates the end of a segment.

DB2TX_ATOMIC
Indicates that the item that follows is atomic, such as a document name or a language identifier.

The data stream items are each two bytes long. They are:

DB2TX_DOC
Indicates the start and end of a document.

DB2TX_DNAM
A document name. If no name is specified, the identifier of the document is used.

DB2TX_DEL
Indicates the start and end of a document element. The only type of document element currently supported is a *text section*.

DB2TX_SNAM
Specifies the name of a text section. Currently DB2 Text Extender supports only one text section and automatically supplies a default name. If you specify a section name, it is ignored.

DB2TX_PAR
Indicates the start and end of a text paragraph within the current section.

DB2TX_TEXT
Specifies one text portion within the current paragraph. Usually, *text unit* contains one line of text, and the TEXT item is followed by a DB2TX_NL item; but text lines may also be split into several parts, each part specified in its own DB2TX_TEXT item.

The text uses the CCSID and language associated with the current paragraph.

DB2TX_LINK

Specifies a DB2 Text Extender hypermedia reference. It uses the CCSID of the current paragraph.

DB2TX_NL

Indicates the start of a new line in the current paragraph.

DB2TX_MATCH

Contains occurrence information for matches in the current text portion. The information is supplied as a sequence of binary number pairs. The first number in each pair is the offset of a match within the current text portion, the second number is the length, in characters, of that match. The given length could exceed the given text portion. Both offset and length are two-byte values specified in big-endian format.

DB2TX_CCSID

The CCSID for text in subsequent paragraphs until a paragraph is preceded by a new DB2TX_CCSID item. The following CCSIDs are returned:

`DB2TX_CCSID_00500`

for text in the Latin-1 EBCDIC code page 500.

`DB2TX_CCSID_04946`

for text in the Latin-1 ASCII code page 850.

`DB2TX_CCSID_00819`

for text in the ASCII code page 819.

These symbolic names for CCSIDs are defined in the file DES_EXT.H provided with the DB2 Text Extender. The two-byte binary values are specified in big-endian format.

DB2TX_LANG

The language identifier for text in subsequent paragraphs until a paragraph is preceded by a new DB2TX_LANG item. File DES_EXT.H provided with DB2 Text Extender defines symbolic names for all language identifiers supported by DB2 Text Extender. The two-byte binary values are specified in big-endian format.

Usage

DesGetMatches returns RC_SE_END_OF_INFORMATION when the end of the text document is reached.

Return codes

RC_SUCCESS
RC_SE_END_OF_INFORMATION

RC_INVALID_PARAMETER
RC_INVALID_SESSION
RC_SE_CAPACITY_LIMIT_EXCEEDED
RC_SE_INCORRECT_HANDLE
RC_SE_IO_PROBLEM
RC_SE_NOT_ENOUGH_MEMORY
RC_SE_REQUEST_IN_PROGRESS
RC_SE_LS_FUNCTION_FAILED
RC_SE_UNEXPECTED_ERROR

Warnings: The following return codes indicate that the function has returned a result, but it may not be as expected.

RC_SE_DICTIONARY_NOT_FOUND

Restrictions

This function can be called only after you have opened a text document by calling DesOpenDocument.

DesGetSearchResultTable

Purpose

Uses a search argument for searching through text documents identified by a text column. The handle data of the found text items is written to a result table. Browse information about rank and the number of matches can also be written to the result table.

Syntax

```
DESRETURN
  DesGetSearchResultTable
    (SQLHDBC          hdbc,
     char             *pTableSchema,
     DESSMALLINT      TableSchemaLength,
     char             *pTableName,
     DESSMALLINT      TableNameLength,
     char             *pColumnName,
     DESSMALLINT      ColumnNameLength,
     char             *pSearchArgument,
     DESSMALLINT      ArgumentLength,
     char             *pResultSchema,
     DESSMALLINT      ResultSchemaLength,
     char             *pResultTableName,
     DESSMALLINT      ResultTableNameLength,
     DESSEARCHOPTION  SearchOption,
     DESBROWSEOPTION  BrowseOption,
     DESBROWSEINFO    *pBrowseInfo,
     DESMESSAGE       *pErrorMessage);
```

Function arguments

Table 15. DesGetSearchResultTable arguments

Data Type	Argument	Use	Description
SQLHDBC	hdbc	input	A database connection handle.
char *	pTableSchema	input	The schema of the base table to be searched.
DESSMALLINT	TableSchemaLength	input	Either the length of pTableSchema (not including a null byte terminator) or DES_NTS.
char *	pTableName	input	Pointer to the name of the base table to be searched.
DESSMALLINT	TableNameLength	input	Either the length of pTableName (not including a null byte terminator) or DES_NTS.
char *	pColumnName	input	Pointer to the name of the column to be addressed by the intended text search. The column must be of type DESTEXTH.
DESSMALLINT	ColumnNameLength	input	Either the length of pColumnName (not including a null byte terminator) or DES_NTS.
char *	pSearchArgument	input	Pointer to the text search argument.
DESSMALLINT	ArgumentLength	input	Either the length of pSearchArgument (not including a null byte terminator) or DES_NTS.

DesGetSearchResultTable API function

Table 15. DesGetSearchResultTable arguments (continued)

Data Type	Argument	Use	Description
char *	pResultSchema	input	Pointer to the schema containing the result table.
DESSMALLINT	ResultSchemaLength	input	Either the length of pSchemaName (not including a null byte terminator) or DES_NTS.
char *	pResultTableName	input	Pointer to the name of the result table that you have previously created in which the result of the search is to be stored. See Figure 17 on page 213 for the structure of this table.
DESSMALLINT	ResultTableNameLength	input	Either the length of pResultTableName (not including a null byte terminator) or DES_NTS.
DESSEARCHOPTION	SearchOption	input	An option that determines whether you are asking for ranking information, for the number of matches, or only for the handles of the matching text documents. 　　DES_RANK 　　DES_MATCH 　　DES_RANKANDMATCH 　　DES_TEXTHANDLEONLY This option determines the content of result table as described in "Usage".
DESBROWSEOPTION	BrowseOption	input	Reserved.
DESBROWSEINFO *	pBrowseInfo	output	Pointer to browse information or a pointer to null, depending on the value of BrowseOption.
DESMESSAGE *	*pErrorMessage*	output	Implementation-defined message text. If an error occurs, DB2 Text Extender returns an error code and an error message. The application program allocates the buffer of size DES_MAX_MESSAGE_LENGTH. If *pErrorMessage* is the null pointer, no error message is returned.

Usage

The connection to the database must be established by the application program calling DesGetSearchResultTable.

The name *pResultTableName* refers to a result table that you have created in advance. The utility DESRESTB in the sample directory creates a result table for text handles. After the call of this function, the result table contains information identifying text values matching the search argument. This is the structure of the result table:

DesGetSearchResultTable API function

RC_SQL_ERROR_NO_INFO
RC_SQL_ERROR_WITH_INFO
RC_TEXT_COLUMN_NOT_ENABLED

Warnings: The following return codes indicate that the function has returned a result, but it may not be as expected.

RC_SE_CONFLICT_WITH_INDEX_TYPE
RC_SE_DICTIONARY_NOT_FOUND
RC_SE_STOPWORD_IGNORED
RC_SE_UNKNOWN_SECTION_NAME
RC_SE_DOCMOD_READ_PROBLEM

DesOpenDocument

Purpose

Receives a browse session pointer, a handle, and an option (DES_EXTENDED or DES_FAST) indicating whether the text document should be analyzed with or without the use of a dictionary. It prepares the text document that corresponds to the handle to get the document text and highlighting information, and it returns a document handle that is used for iteratively calling DesGetMatches.

Syntax

```
DESRETURN
  DesOpenDocument
    (DESBROWSESESSION    BrowseSession,
     SQLCHAR             *pHandle,
     DESUSHORT           HandleLength,
     DESMATCHMODE        MatchMode,
     DESHANDLE           *pDocumentHandle,
     DESMESSAGE          *pErrorMessage);
```

Function arguments

Table 16. DesOpenDocument arguments

Data Type	Argument	Use	Description
DESBROWSESESSION	*BrowseSession*	input	Browse session handle.
SQLCHAR *	*pHandle*	input	Pointer to a handle extracted from the database.
DESUSHORT	HandleLength	input	Length of pHandle (DES_NTS cannot be used).
DESMATCHMODE	MatchMode	input	Mode to determine whether a dictionary is used for finding the highlighting information. **DES_FAST** 　　Do not use a dictionary **DES_EXTENDED** 　　Use a dictionary
DESHANDLE *	pDocumentHandle	output	A document handle for iteratively calling DesGetMatches.
DESMESSAGE *	*pErrorMessage*	output	Implementation-defined message text. If an error occurs, DB2 Text Extender returns an error code and an error message. The application program allocates the buffer of size DES_MAX_MESSAGE_LENGTH. If *pErrorMessage* is the null pointer, no error message is returned.

Usage

DES_FAST and DES_EXTENDED refer to the use of linguistic processing for finding which terms to highlight in the browsed text. See "Linguistic processing for browsing" on page 230 for more information. Specify DES_FAST to use basic text analysis, and DES_EXTENDED to use extended matching.

For the mapping between the SQL data types and C data types, you must use the SQL symbolic name SQL_VARBINARY for a handle. The type of host variables pointing to the C representation of *TextHandle* values is SQLCHAR*.

DB2 Text Extender allocates storage for the browse information. The application program must free this storage and related resources by calling DesFreeBrowseInfo.

Because *TextHandle* values are bit data and contain several '\0' characters, you must specify the length of *pHandle*.

The caller must have read access to the table containing the text document referred to by pHandle.

Return codes

RC_SUCCESS

RC_ALLOCATION_ERROR
RC_INTERNAL_ERROR
RC_INVALID_MATCH_OPTION
RC_INVALID_PARAMETER
RC_INVALID_SESSION
RC_SE_DOCUMENT_NOT_ACCESSIBLE
RC_SE_DOCUMENT_NOT_FOUND
RC_SE_INCORRECT_HANDLE
RC_SE_IO_PROBLEM
RC_SE_LS_FUNCTION_FAILED
RC_SE_LS_NOT_EXECUTABLE
RC_SE_MAX_NUMBER_OF_BUSY_INDEXES
RC_SE_NOT_ENOUGH_MEMORY
RC_SE_REQUEST_IN_PROGRESS
RC_SE_UNKNOWN_INDEX_NAME
RC_SE_UNEXPECTED_ERROR

Restrictions

This function can be called only after you have started a browse session by calling DesStartBrowseSession.

DesStartBrowseSession

Purpose

Starts a browse session, establishing the environment needed for browsing a text document and highlighting its matches. It receives a pointer to browse information, either from DesGetBrowseInfo or from DesGetSearchResultTable, and returns a browse session handle for use by the other browse functions.

Syntax

```
DESRETURN
  DesStartBrowseSession
     (DESBROWSEINFO        BrowseInfo,
      char                *pUserId,
      DESSMALLINT          UserIdLength,
      char                *pPassword,
      DESSMALLINT          PasswordLength,
      DESBROWSESESSION    *pBrowseSession,
      DESMESSAGE          *pErrorMessage);
```

Function arguments

Table 17. DesStartBrowseSession arguments

Data Type	Argument	Use	Description
DESBROWSEINFO	*BrowseInfo*	input	Pointer to information needed for browsing and highlighting matches in a text document. The pointer is returned by DesGetSearchResultTable or DesGetBrowseInfo.
char *	*pUserId*	input	User ID for the database
DESSMALLINT	*UserIdLength*	input	Length of the user ID for the database
char *	*pPassword*	input	Password for the database
DESSMALLINT	*PasswordLength*	input	Length of the password for the database
DESBROWSESESSION *	*pBrowseSession*	output	A handle for a browse session for use by other browse functions.
DESMESSAGE *	*pErrorMessage*	output	Implementation-defined message text. If an error occurs, DB2 Text Extender returns an error code and an error message. The application program allocates the buffer of size DES_MAX_MESSAGE_LENGTH. If *pErrorMessage* is the null pointer, no error message is returned.

Usage

This function opens a browse session for browsing text documents. You are prompted for your user ID and password to check your authorization to access the database.

You close the browse session by calling DesEndBrowseSession.

BrowseInfo depends on the search argument and on the base text column used for building the browse information.

Return codes

RC_SUCCESS

RC_ALLOCATION_ERROR
RC_INVALID_BROWSE_INFO
RC_INVALID_PARAMETER
RC_INTERNAL_ERROR
RC_SE_NOT_ENOUGH_MEMORY
RC_SE_UNEXPECTED_ERROR
RC_SQL_ERROR_NO_INFO
RC_SQL_ERROR_WITH_INFO

Restrictions

You must call DesGetBrowseInfo or DesGetSearchResultTable with the appropriate
Browse Option before calling this function.

Chapter 14. Sample API program

DB2 Text Extender provides a sample program, DESSAMP1.C, located in the SAMPLES directory.

DESSAMP1.C is an example of a program that uses the DesGetSearchResultTable function and your own browser program. It follows the sequence of API function calls shown in Figure 16 on page 98.

You need access to an enabled database and an enabled text column. To run the sample program, do the following:

1. Optional. Copy the source file DESSAMP1.C to a local directory on your client machine.

2. Use the supplied makefile (`dessamp.mak` for OS/2, Windows NT, and Windows 2000; `dessamp` for UNIX systems) to compile and link the sample files.

3. Run the utility DESRESTB to create a result table in the database that you intend to use with the sample code:

   ```
   DESRESTB database-name
   ```

This table is used to store information such as the search results; it has the following structure:

Column	Data type
HANDLE	DB2TX.DB2TEXTH or DB2TX.DB2TEXTFH
RANK	DOUBLE
MATCHES	INTEGER

Sample API program

Chapter 15. Linguistic processing for linguistic and precise indexes

DB2 Text Extender offers linguistic processing in these areas of retrieval:

- **Indexing**. When DB2 Text Extender analyzes documents to extract the terms to be stored in the text index, the text is processed linguistically to extract the right terms for the index. This is done to make retrieval as simple and as fast as possible.
- **Retrieval**. When DB2 Text Extender searches through the document index to find the documents that contain occurrences of the search terms you have specified, the search terms are also processed linguistically to match them with the indexed terms.
- **Browsing**. When you browse a document that has been found after a search, linguistic processing is used to highlight the terms found in the document.

Linguistic processing when indexing

When DB2 Text Extender indexes and retrieves documents, it makes a linguistic analysis of the text. As you can see from the following table, the amount of linguistic processing depends on the index type. For Ngram indexes, no linguistic processing is applied.

The linguistic processing used for indexing documents consists of:

- Basic text analysis

    Recognizing terms (tokenization)

    Normalizing terms to a standard form

    Recognizing sentences

- Reducing terms to their base form
- Stop-word filtering
- Decomposition (splitting compound terms).

Table 18 shows a summary of how terms are indexed when the index type is **linguistic** and no additional index properties have been requested.

Table 18. Term extraction for a linguistic index

Document text	Term in index	Linguistic processing
Mouse Käfer	mouse kaefer	Basic text analysis (normalization)
mice swum	mouse swim	Reduction to base form
system-based Wetterbericht	system-based, system base wetterbericht, wetter bericht	Decomposition
a report on animals	report animal	Stop-word filtering. Stop words are: a, on

Linguistic processing when indexing

By comparison, Table 19 shows a summary of how terms are indexed when the index type is **precise.**

Table 19. Term extraction for a precise index

Document text	Term in index	Linguistic processing
Mouse Käfer	Mouse Käfer	No normalization
mice swum	mice swum	No reduction to base form
a report on animals	report animals	Stop-word filtering. Stop words are: a, on
system-based Wetterbericht	system-based Wetterbericht	No decomposition

Basic text analysis

DB2 Text Extender processes basic text analysis without using an electronic dictionary.

Recognizing terms that contain nonalphanumeric characters

When documents are indexed, terms are recognized even when they contain nonalphanumeric characters, for example: "$14,225.23", "mother-in-law", and "10/22/90".

The following are regarded as part of a term:

Accents

Currency signs

Number separator characters (like "/" or "·")

The "@" character in e-mail addresses (English only)

The "+" sign.

Language-specific rules are also used to recognize terms containing:

* Accented prefixes in Roman languages, such as `l'aventure` in French.
* National formats for dates, time, and numbers.
* Alternatives, such as `mission/responsibility`, indicated in English using the "/" character.
* Trailing apostrophes in Italian words like `securita'`. It is usual in typed Italian text, when the character set does not include characters with accents, to type the accent *after* the character; for example, "à" is typed "a'".

Normalizing terms to a standard form

Normalizing reduces mixed-case terms, and terms containing accented or special characters, to a standard form. This is done by default when the index type is linguistic. (In a precise index the case of letters is left unchanged—searches are case-sensitive.)

For example, the term `Computer` is indexed as `computer`, the uppercase letter is changed to lowercase. A search for the term `computer` finds occurrences not only of `computer`, but also of `Computer`. The effect of normalization during indexing is that terms are indexed in the same way, regardless of how they are capitalized in the document.

Normalization is applied not only during indexing, but also during retrieval. Uppercase characters in a search term are changed to lowercase before the search is made. When your search term is, for example, `Computer`, the term used in the search is `computer`.

Accented and special characters are normalized in a similar way. Any variation of `école`, such as `École`, finds `école`, `Ecole`, and so on. `Bürger` finds `buerger`, `Maße` finds `masse`.

If the search term includes masking (wildcard) characters, normalization is done before the masking characters are processed. Example: `Bür_er` becomes `buer_er`.

Recognizing sentences

You can search for terms that occur in the same sentence. To make this possible, each document is analyzed during indexing to find out where each sentence ends.

DB2 Text Extender offers two types of sentence-end recognition:
- Universal Unicode Tokenizer for languages other than Arabic and Hebrew.

  This is the simpler, but faster method. The tokenizer looks for a period, exclamation or question mark, preceded by a token character, such as a letter, and followed by a blank, tab, or new-line character. To check that this is really the end of a sentence and not just an abbreviation ending in a period, a language-specific list of abbreviations is checked.
- POE-Based tokenizer for Arabic and Hebrew

  This tokenizer is linguistically more advanced, but requires more processing power. The tokenizer finds the end of sentences primarily through punctuation matching, but also by taking cues from special input types and the number of words.

Tokenizer for recognizing sentences

The POE-based tokenizer determines sentence (or sentence fragment) boundaries using punctuation rules and language-specific processing involving abbreviation processing, although the level of function varies widely by language. Most languages that use single-byte code pages have an associated Abbreviation Addenda Dictionary which is provided with POE. Because double-byte languages typically do not employ abbreviations with periods, Abbreviation Addenda Dictionaries are not available for these languages.

The determination of the end of a sentence is done primarily through punctuation matching. The following table lists the terminating punctuation characters and their Graphic Character Global Identifieries (GCGIDs).

GCGID of SBCS characters	GCGID of DBCS characters	Description
SP110000	SP110080	Period
SP020000	SP020080	Exclamation mark
SP150000	SP150080	Question mark
SP140000	N/A	Semi-colon (Greek Question mark)
N/A	JQ730080	Double-byte circle period

A terminating punctuation character, such as a period, an exclamation mark, or a question mark is assumed to mark the end of a sentence unless one of the following occurs:

Linguistic processing when indexing

- The terminating punctuation character is followed by a closing punctuation character listed in the following table, such as a closing parenthesis.

GCGID of SBCS characters	GCGID of DBCS characters	Description
SP070000	SP070080	Closing parenthesis
SP040000	SP040080	Double quote
SP050000	SP050080	Single quote
SP180000	SP070083	Double angled quote
N/A	SM140080	Closing brace
N/A	SM080080	Closing bracket
N/A	JQ720080	Single square quote
N/A	JQ720081	Double square quote
N/A	SP200080	Closing single hook quote
N/A	SP220080	Closing double hook quote
N/A	SP070081	Closing carapace bracket
N/A	SP070082	Closing single angle quote
N/A	SP070084	Closing cornered parenthesis
N/A	SP370080	Vertical closing single square quote
N/A	SP370081	Vertical closing double square quote
N/A	SP250084	Vertical closing squared parenthesis
N/A	SP250080	Vertical closing parenthesis
N/A	SP350080	Vertical closing brace
N/A	SP250081	Vertical closing carapace bracket
N/A	SP250083	Vertical closing double angled quote
N/A	SP250082	Vertical closing single angled quote

Note: The items marked N/A are not considered closing punctuation characters by POE, and the vertical closing punctuation characters are supported in Chinese only.

Example:

```
...this sentence ends with two parentheses.))
```

In the example, the second parenthesis is detected as the end of the sentence. However, in National German, a closing quotation mark is not considered to mark the end of a sentence if it is followed by a comma.
- The terminating punctuation character is followed by another terminating character. Example:

```
This is a strong exclamation!!!
```

The final exclamation mark is detected as the end of the sentence.
- The terminating punctuation character is preceded either by a numeric or a punctuation character, and followed by a numeric character. This prevents strings such as '1.25' and '.314' from ending a sentence.

Linguistic processing when indexing

- The terminating punctuation character is a period and is part of an abbreviation that is not allowed at the end of a sentence. Limited abbreviation processing is performed for every language.
- The terminating punctuation character is a period and is not followed by a white-space character, such as a blank or a new-line character. This is to avoid headings like 'III.IV' being detected as ending a sentence.

The POE-based tokenizer also does abbreviation processing to determine if a period is part of an abbreviation or if it marks the end of a sentence. You can add abbreviations to an abbreviation addenda dictionary. If no dictionary is passed to the POE-based tokenizer, all single letters followed by periods are marked as abbreviations; no other abbreviation processing takes place.

Whether or not a piece of text is an abbreviation is often ambiguous, because an abbreviations can be mistaken for a normal word followed by a period. For example, consider the characters "no." in the following sentences:

```
Enter the no. of exemptions you are claiming.
Answer each question with yes or no.
```

But even when a piece of text is known to be an abbreviation, there is still ambiguity as to whether it ends a sentence. Some abbreviations never end a sentence, while others sometimes do. For example, consider the use of the abbreviation "Hwy." in the following sentences:

```
The drive along Hwy. 1 to Santa Cruz was beautiful.
Many people speak highly of the Pacific Coast Hwy.
```

Because abbreviations can be ambiguous and because some abbreviations may not occur at the end of a sentence, POE attempts to classify the found abbreviations. If a period is found to be part of an abbreviation that sometimes ends a sentence, further processing is performed. If POE determines that the abbreviation is not at the end of the sentence, the token representing the period is joined with the token for the abbreviation text. Otherwise, the token representing the period remains a separate token.

POE-based abbreviation processing uses three sets of criteria to determine whether a period is part of an abbreviation:
- All single letters followed by a period are considered to be an abbreviation. Single-letter abbreviations are classified as possible sentence ends.
- Words contained in the input Abbreviation Addenda Dictionary are always considered to be abbreviations. Whether an abbreviation found in the addenda can end a sentence is determined by the information associated with that word in the addenda. For example,"Mr." is marked in the U.S. English Abbreviation Addenda Dictionary as an abbreviation that cannot end a sentence, while "etc." is an abbreviation that can sometimes end a sentence.
- Any word from two to six characters in length followed by a period, and not found in any of the input dictionaries or Abbreviation Addenda Dictionaries, are also considered to be abbreviations. This is to handle cases like "Jrnl. Comp. Ling.". Abbreviations determined by dictionary lookup are always treated as a possible end of sentence.

If an abbreviation is identified as a possible end of sentence, POE examines the text that follows the abbreviation to determine whether the abbreviation is at the end of the current sentence by checking whether the next word begins with an uppercase letter.

If an abbreviation is followed by two or more new-line, new-sentence, or new-paragraph data elements, POE assumes that an end-of-sentence has been reached. Also, if the subsequent text is an inverted question mark or an inverted exclamation mark, an end-of-sentence marker is inserted in the output.

If POE determines that the period is part of an abbreviation that does not end a sentence, it continues its search for a sentence delimiter. Otherwise it checks other terminating punctuation character exception conditions (following terminating punctuation or closing punctuation) before marking an end of sentence.

Reducing terms to their base form (lemmatization)

In a linguistic index, you can search for `mouse`, for example, and find `mice`. Terms are reduced to their base form for indexing; the term `mice` is indexed as `mouse`. Later, when you use the search term `mouse`, the document is found. The document is found also if you search for `mice`.

The effect is that you find documents containing information about mice, regardless of which variation of the term `mouse` occurs in the document, or is used as a search term.

In the same way, conjugated verbs are reduced to their infinitive; `bought`, for example, becomes `buy`.

Stop-word filtering

Stop words are words such as prepositions and pronouns that occur very frequently in documents, and are therefore not suitable as search terms. Such words are in a stop-word list associated with each dictionary, and are excluded from the indexing process.

Stop word processing is case–insensitive. So a stop word `about` also excludes the first word in a sentence `About`. This is The stop word lists, supplied in various languages, can be modified.

An Ngram index does not have a stop word list.

Decomposition (splitting compound terms)

Germanic languages, such as German or Dutch, are rich in compound terms, like `Versandetiketten`, which means `mail (Versand) labels (Etiketten)`. Such compound terms can be split into their components.

For a precise index, compound terms are indexed unchanged as one word. For a linguistic index, compound terms are split during indexing. When you search, compound terms are split if you have a linguistic index.

The components are found if they occur in any sequence in a document as long as they are contained within one sentence. For example, when searching for the German word `Wetterbericht` (weather report), a document containing the phrase `Bericht über das Wetter` (report about the weather) would also be found.

An attempt is made to split a term if:
- The term's language uses compound terms
- The term has a certain minimum length

Linguistic processing when indexing

- The term is not itself an entry in the electronic dictionary—compounds that are commonly used like the German word `Geschäftsbericht` (business report) *are* in the German dictionary.

If a split is found to be possible, the term's component parts are then reduced to their base form. Here are some examples from Danish, German, and Dutch:

Compound term	Component parts
børsmæglerselskab	børsmæglerselskab børs mægler selskab
Kindersprachen	kindersprache kind sprache
probleemkinderen	probleemkinderen probleemkind kind probleem

Linguistic processing for retrieval

Query processing aims at making search terms weaker so that the recall rate of searches is increased, that is, more relevant documents are found. There are two basic operations on query terms to achieve that goal; they are expansions and reductions. In addition, some search term operations involve both expansion and reduction.

- Expansions take a word or a multi-word term from within a search term and associate it with a set of alternative search terms, each of which may be a multi-word term itself. The source expression and the set of target expressions form a Boolean OR-expression in DB2 Text Extender's query language. As expansions leave the source term unchanged, they are to some extent independent of the index type. The following are expansion operations:

  Synonym expansion

  Thesaurus expansion

- Reductions change the search term to a form that is more general than the one specified by the user. Because it changes the search term, reductions are dependent on the index type to ensure that the changed term matches. Therefore, DB2 Text Extender derives reduction information from the type of those indices or index that the query is directed against. The following are reductions:

  Lemmatization (see "Reducing terms to their base form (lemmatization)" on page 227)

  Normalization (see "Normalizing terms to a standard form" on page 222).

  Stop words (see "Stop-word filtering" on page 227).

- Some operations both change the search term and expand it with a set of alternative terms. Due to the inherent reduction, these again depend on information contained in the index. The following operations fall into this class:

  Character and word masking

  Sound expansion.

Synonyms

Synonyms are semantically related words. Usually, these words have the same word class or classes (such as noun, verb, and so on) as the source term. Synonyms are obtained from a separate file for each language. They are always returned in base form and, up to a few exceptions, are not multi-word terms. Search term words are always reduced to their base form when looking up synonyms. Here are some examples of a word's synonyms in three languages:

- English

  word:
  ```
  comment remark statement utterance term expression
  communication message assurance guarantee warrant bidding command
  charge commandment dictate direction directive injunction instruction
  mandate order news advice intelligence tidings gossip buzz cry
  hearsay murmur report rumor scuttlebutt tattle tittle-tattle
  whispering
  ```

- French

  mot:
  ```
  expression parole terme vocable lettre billet missive epître
  plaisanterie
  ```

- German

  Wort:
  ```
  Vokabel Bezeichnung Benennung Ausdruck Begriff Terminus
  Ehrenwort Brocken Bekräftigung Versprechen Zusicherung Gelöbnis
  Beteuerung Manneswort Schwur Eid Ausspruch
  ```

Thesaurus expansion

A search term can be expanded using thesaurus terms that can be reached through a specific relation. These relations may be hierarchical (such as the "Narrower term" relation), associative (such as a "Related term" relationship), or it may be a synonym relationship. A thesaurus term may be, and often is, a multi-word term.

"Thesaurus concepts" on page 231 describes thesaurus expansion in more detail.

The search term (start term) is not normalized when the thesaurus lookup is done. The words that result from the thesaurus lookup are reduced to their base form according to the index type.

Sound expansion

Sound expansion expands single words through a set of similarly sounding words. It is particularly useful whenever the exact spelling of a term to be searched is not known.

Character and word masking

Masking is a non-linguistic expansion technique, where a regular expression is replaced with the disjunction of all indexed words that satisfy it. Neither a masked expression nor any of its expansions is subject to lemmatization, stop-word extraction, or any of the other expansion techniques. This may have the effect that, for example, an irregular verb form like swum, when searched with the masked term swu*, is matched on a precise index, but not on a linguistic index, where this form has been lemmatized to become swim.

If you use word masking, performance can be slow, especially when searching in large indexes.

Linguistic processing for browsing

Linguistic processing is also used when you browse documents that have been found after a search. It is done in two stages:

1. Basic text analysis: normalization and term expansion
2. Extended matching.

Stage 1: Normalization and term expansion

The first stage is done without using an electronic dictionary.

Normalization
Normalization is described in "Basic text analysis" on page 222.

Term expansion
Term expansion is the inverse of reducing a term to its base form. If the index is linguistic, then search terms are reduced to their base form before the search begins.

Similarly, if you have a linguistic index, a document's terms are reduced to their base form before being added to the index. Documents are therefore found on the basis of a term's base form.

When you browse a found document, however, you expect to see all variants of the base form highlighted. To highlight these variants, the found base term is expanded.

All variants (inflections) for each term found in the dictionaries can be produced. These are the inflections produced for the German word gehen (to go):

```
gegangen  geh     gehe    gehen    gehend  gehest  gehet  gehst
ging             ginge   gingen   gingest  ginget  gingst  gingt  geht
```

Stage 2: Extended matching

The second stage is extended matching, which can be used on the rare occasions when basic text analysis and normalization cannot highlight a found term. Extended matching finds the more obscure matches.

You choose extended matching by specifying DES_EXTENDED as a parameter in the DesOpenDocument API function.

Extended matching uses the same linguistic processing that is done while linguistically indexing.

These are the occasions when extended matching can find additional matches:

• The search term includes masking characters and is an inflection.

  Masking characters are processed and stem reduction is done for the search term and the corresponding documents are found. Without extended matching, text that matches the specified search criteria would not be highlighted.

  Example: A document contains the inflected term swam.

  – During indexing this term is reduced to swim.

  – If the search term is swi%, the above document is found, because the stem reduction is swim.

- Without extended matching, only those words that match the term `swi%` are highlighted. With extended matching, the inflected term `swam` is also highlighted.

- If compound words have been indexed.

  When a document in a Germanic language contains a compound word and is indexed using a linguistic index, the document index retains the parts of the compound word and the compound word itself. When you search for a part of a compound word, the documents containing the compound word are found, but without extended matching the word is not highlighted.

  Example: A document contains the German word `Apfelbaum` (apple tree).

  - During linguistic indexing, this word is reduced to `apfel` and `baum`.
  - When the index is searched for the term `baum`, the term `Baum` and the document that contains it is found through the index.
  - Without extended matching, no terms are highlighted because the document contains `Apfelbaum`, but not `Baum`. With extended matching, the `Apfelbaum` compound is split and the `Baum` part is found for highlighting.

- If words are hyphenated at the end of a line.

  If the hyphen is inserted automatically by a word processor, the hyphenated word can be found and highlighted. If, however, the hyphen is typed by the user, the documents containing the word are found, but without extended matching the word is not highlighted.

  Example: A document contains the hyphenated word `container`, broken at the end of a line like this:

  ```
  Another name for a folder is a con-
  tainer.
  ```

  - During indexing the word is normalized to `container`.
  - When the index is searched for the term `container`, the term and the document that contains it is found.
  - An attempt is made to highlight any words in the document that match `container`. Without extended matching, a match is found only if the hyphen in `con-tainer` was inserted by the text processor, and not typed by a user.

Thesaurus concepts

A thesaurus is a controlled vocabulary of semantically related terms that usually covers a specific subject area. It can be visualized as a semantic network where each term is represented by a node. If two terms are related to each other, their nodes are connected by a link labeled with the relation name. All terms that are directly related to a given term can be reached by following all connections that leave its node. Further related terms can be reached by iteratively following all connections leaving the nodes reached in the previous step. Figure 18 on page 232 shows an example of the structure of a very small thesaurus.

Thesaurus concepts
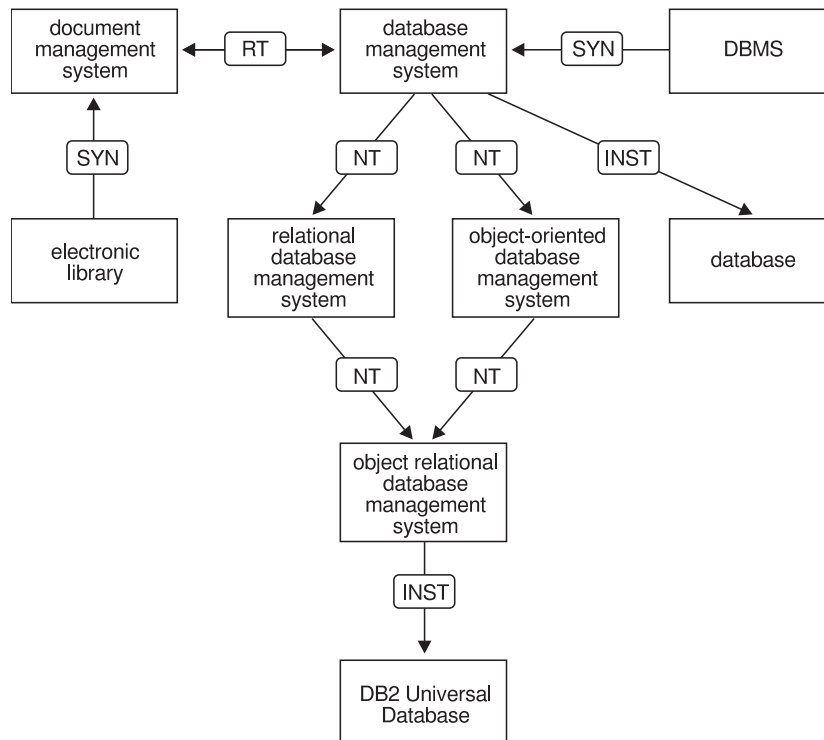
Figure 18. A thesaurus displayed as a network

DB2 Text Extender lets you expand a search term by adding additional terms from a thesaurus that you have previously created. Refer to "Chapter 12. Syntax of search arguments" on page 187 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.

The dictionary format used by a linguistic and a precise index differs from the one used by an Ngram index. Thus two different thesaurus compilers are provided with the product. They are not only slightly different in the concepts they are based on, but require different source formats. So you should first decide which index type you will use before you start defining the thesauri for your search application.

The basic components of a thesaurus are "terms" and "relations".

Terms

A term is a word or expression denoting a concept within the subject domain of the thesaurus. For example, the following could be terms in one or more thesauri:

```
data processing
helicopter
gross national product
```

In a DB2 Text Extender thesarus, terms are classified as either descriptors or nondescriptors. A *descriptor* is a term in a class of synonyms that is the preferred term for indexing and searching. The other terms in the class are called

nondescriptors. For example, `outline` and `shape` are synonymous, where `shape` could be the descriptor and `outline` a nondescriptor.

An Ngram thesaurus does not distinguish between descriptors and nondescriptors.

Relations

A relation is an expression of an association between two terms. Relations have the following properties:

- The *depth* of a relation is the number of levels over which the relation extends. This is specified in the search syntax using the THESDEPTH keyword. Refer to "Chapter 12. Syntax of search arguments" on page 187 to find out how to use thesaurus expansion in a query.
- The *directionality* of a relation specifies whether the relation is true equally from one term to the other (bidirectional), or in one direction only (unidirectional).

Thesaurus expansion can use every relation defined in the thesaurus. You can also specify the depth of the expansion. This is the maximum number of transitions from a source term to a target term. Note however that the term set may increase exponentially as the depth is incremented.

The following example shows those terms that are newly added as the depth increases.

```
health

health service, paramedical, medicine, illness

allergology, virology, veterinary medicine, toxicology, surgery,
stomatology, rhumatology, radiotherapy, psychiatry, preventive
medicine, pathology, odontology, nutrition, nuclear medicine,
neurology, nephrology, medical check up, industrial medicine,
hematology, general medicine, epidemiology, clinical trial,
cardiology, cancerology
```

DB2 Text Extender thesaurus relations

These are the relation types provided by a DB2 Text Extender thesaurus:

- Associative
- Synonymous
- Hierarchical
- Other

In a DB2 Text Extender thesaurus there are no predefined relations. You can give each relation a name, such as BROADER TERM, which can be a mnemonic abbreviation, such as BT. The common relations used in thesaurus design are:

- BT or BROADER TERM
- NT or NARROWER TERM
- RT or RELATED TERM
- SYN or SYNONYM
- USE
- UF or USE FOR

Associative: An associative relation is a bidirectional relation between descriptors, extending to any depth. It binds two terms that are neither equivalent nor hierarchical, yet are semantically associated to such an extent that the link between them may suggest additional terms for use in indexing or retrieval.

Associative relations are commonly designated as RT (related term). Examples are:
    dog RT security
    pet RT veterinarian

Synonymous: When a distinction is made between descriptors and nondescriptors, as it is in a DB2 Text Extender thesaurus, the synonymous relation is unidirectional between two terms that have the same or similar meaning. In a class of synonyms, one of the terms is designated as the descriptor. The other terms are then called nondescriptors. Refer to "Ngram thesaurus relations" for a definition of the synonymous relation when no distinction is made between descriptors and nondescriptors.

The common designation USE leads from a given nondescriptor to its descriptor. The common designation USE FOR leads from the descriptor to each nondescriptor. For example:
    feline USE cat
    lawyer UF advocate

Hierarchical: A hierarchical relation is a unidirectional relation between descriptors that states that one of the terms is more specific, or less general, than the other. This difference leads to representation of the terms as a hierarchy, where one term represents a class, and subordinate terms refer to its member parts. For example, the term "mouse" belongs to the class "rodent".

BROADER TERM and NARROWER TERM are hierarchical relations. For example:
    car NT limousine
    equine BT horse

Other: A relation of type *other* is the most general. It represents an association that does not easily fall into one of the other categories. A relation of type *other* can be bidirectional or unidirectional, there is no depth restriction, and relations can exist between descriptors and nondescriptors.

This relation is often used for new terms in a thesaurus until the proper relation with other terms can be determined.

Of course you can define your own bidirectional synonymous relation by using the relation type *associative* for a synonymous relation between descriptors or even with the relation type *other* for a synonymous relation between arbitrary terms.

Ngram thesaurus relations

An Ngram thesaurus supports the following two types:
- Associative
- Synonymous

There are two predefined relations, each of them based on one of these two types. You can define your own relations based on the type *associative*. For details, see "Creating an Ngram thesaurus" on page 238.

Associative
An associative relation is a bidirectional relation between two terms that do not express the same concept but relate to each other. The predefined relation *RELATED_TO* and all user-defined relations are based on this relation type.

Examples are:

    tennis RELATED_TO racket
    German RELATED_TO sausage

Synonymous

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 for a term and its abbreviation. The predefined relation SYNONYM_OF is the only relation based on this type.

Examples are:
    spot SYNONYM_OF stain
    US SYNONYM_OF United States

Creating a thesaurus

See also "Creating an Ngram thesaurus" on page 238.

There is a sample English thesaurus compiler input file `desthes.sgm` stored in the samples directory of the installation path. The dictionary directory on OS/2 and Windows systems is *drive*`:\dmb\db2tx\samples`. On AIX, HP-UX, and SUN-Solaris systems, the directory is `DB2TX_INSTOWNER`/db2tx/samples.

A compiled version of this thesaurus and its SGML input file is stored in the dictionary directory. *drive*`:\dmb\db2tx\dict` or `DB2TX_INSTOWNER`/db2tx/dicts

The files belonging to this thesaurus are called `desthes.th1,` `desthes.th2,` ..., and `desthes.th6.`.

To create a thesaurus, first define its content in a file. It is recommended that you use a plain directory for each thesaurus that you define. The file can have any extension except `th1` to `th6`, which are used for the thesaurus dictionary. If you use the same directory for an Ngram thesaurus, see "Creating an Ngram thesaurus" on page 238 for more excluded file extensions.

Then compile the file by running:

`txthesc -f` *filename* `-c` *ccsid*

where *filename* can contain only the characters a-z, A-Z, and 0-9.

Currently, only CCSID 850 is supported.

`txthesc` produces thesaurus files having the name *filename* without extension and the extension `th1` to `th6`, in the same directory where the definition file is located. If there is already a thesaurus with the same name, it is overwritten without warning.

Refer to "Chapter 12. Syntax of search arguments" on page 187 to find out how to use a thesaurus in a query.

Specify the content of a thesaurus using the Standard Generalized Markup Language (SGML). The following diagram shows the syntax rules to follow when creating a thesaurus.

►►──`<thesaurus>`──`<header>`──`<thname>`──*thesaurus-name*──`</thname>`─────────────►

Thesaurus concepts
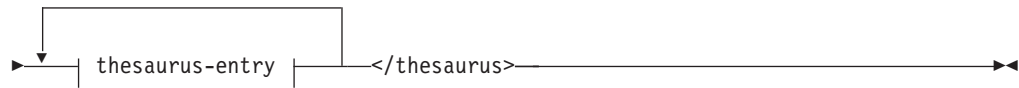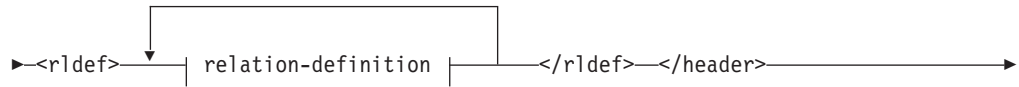
```
►─<rldef>─┬─┤ relation-definition ├─┬──</rldef>──</header>──────────────────►
          └◄────────────────────────┘
```

```
►─┬─┤ thesaurus-entry ├─┬──</thesaurus>─────────────────────────────────────►◄
  └◄───────────────────┘
```

relation-definition:

```
├──<rls>──<rlname>─relation-name──</rlname>─────────────────────────────────►
```

```
►─<rltype>─┬─ASSOCIATIVE──┬──</rltype>──</rls>──────────────────────────────┤
           ├─SYNONYMOUS───┤
           ├─HIERARCHICAL─┤
           └─OTHER────────┘
```

thesaurus-entry:

```
├──<en>─unique-number──,─┬─1─┬──<t>─term──</t>──────────────────────────────►
                         └─2─┘              └─┤ related-terms ├─┘
```

```
►─</en>─────────────────────────────────────────────────────────────────────┤
```
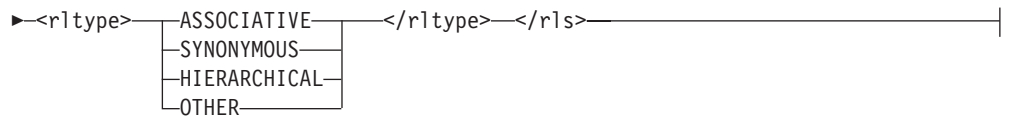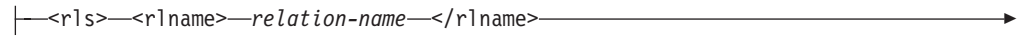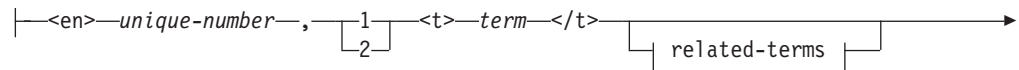
related-terms:

```
├──<r>─┬──<l>─relation-name─┬──<t>─term──</t>─┬──</l>─┬──</r>───────────────┤
       └◄──────────────────┘◄────────────────┘
```
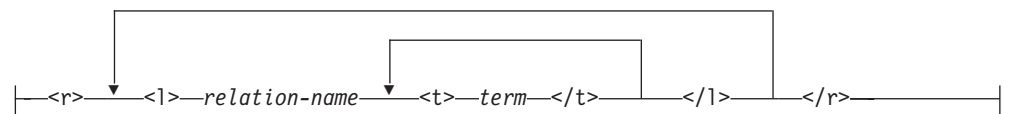
relation-name can contain only the characters a-z, A-Z, and 0-9.

Figure 19 on page 237 shows the SGML definition of the thesaurus shown in Figure 18 on page 232.

```
<thesaurus>
<header>
<thname>thesc example thesaurus</thname>
<rldef>

<rls>
<rlname>Related Term</rlname>
<rltype>associative</rltype>
</rls>

<rls>
<rlname>Narrower Term</rlname>
<rltype>hierarchical</rltype>
</rls>

<rls>
<rlname>Instance</rlname>
<rltype>hierarchical</rltype>
</rls>

<rls>
<rlname>Synonym</rlname>
<rltype>synonymous</rltype>
</rls>
</rldef>
</header>

<en> 2, 1
<t>database management system</t>
<r>
  <l>Narrower Term
  <t>oo database management system</t>
  <t>relational database management system</t>
  </l>

  <l>Synonym
  <t>DBMS</t>
  </l>

  <l>Related Term
  <t>document management system</t>
  </l>

  <l>Instance
  <t>database</t>
  </l>
</r>
</en>
```

Figure 19. The definition of a simple thesaurus (Part 1 of 2)

```
<en> 5, 1
<t> relational database management system </t>
<r>
  <l>Narrower Term
  <t>object relational database management system</t>
  </l>
</r>
</en>

<en> 3, 1
<t>object relational database management system</t>
<r>
  <l>Instance
  <t>DB2 Universal Database</t>
  </l>
</r>
</en>

<en> 6, 1
<t>object oriented database management system</t>
<r>
  <l>Narrower Term
  <t>object relational database management system</t>
  </l>
</r>
</en>

<en> 4, 1
<t>document management system</t>
<r>
  <l>Synonym
  <t>library</t>
  </l>
</r>
</en>

<en> 9, 1
<t>library</t>
</en>

<en> 10, 1
<t>DB2 Unversal Database</t>
</en>

<en> 11, 1
<t>database</t>
</en>
</thesaurus>
```

Figure 19. The definition of a simple thesaurus (Part 2 of 2)

Creating an Ngram thesaurus

There is a sample English Ngram thesaurus compiler input file `desnthes.def` stored in the dictionary directory of the installation path. The dictionary directory on OS/2 and Windows systems is:

drive`:\dmb\db2tx\dict`

On AIX, HP-UX, and SUN-Solaris systems, the dictionary directory is:

`DB2TX_INSTOWNER /db2tx/dicts`

A compiled version of this sample thesaurus is also stored there. The files belonging to this thesaurus are called `desnthes.<extension>` with the following extension where n is a digit:

- For dictionary files: wdf, wdv, grf, grv, MEY, ROS, NEY, SOS, lk*n*
- For temporary files: w*n*f, w*n*v, g*n*f, g*n*v, M!1, M!2, N!1, N!2, R!1, R!2, S!1, S!2, M*nn*, N*nn*, R*nn*, S*nn*, $00, $01, $10, $11, $20, and $21

To create an Ngram thesaurus, first define its content in a definition file. You can have several thesauri in the same directory, but it is recommended that you have a separate directory for each thesaurus. The length of the file name without extension must not exceed 8 characters. The extension is optional but is restricted to 3 characters and should be different from any of the above listed extensions.

If you use the same directory for other DB2 Text Extender thesauri, do not use the extensions listed under "Creating a thesaurus" on page 235.

Then compile the file by running:

```
txthesn -f definition-file-name -ccsid  code-page
```

Here is a list of the code pages supported by an Ngram thesaurus:

932	AIX, OS/2	Japanese
942	OS/2	Japanese
943	OS/2, Windows	Japanese
949	OS/2	Korean
950	AIX, HP-UX, OS/2, SUN-Solaris, Windows	Traditional Chinese
970	AIX, HP-UX, SUN-Solaris	Korean
1381	OS/2, Windows	Simplified Chinese
1363	Windows	Korean
1383	AIX, HP-UX, SUN-Solaris	Simplified Chinese
850	AIX, OS/2	Latin-1

`txthesn` produces thesaurus files having the same name as *definition-file-name* with the extensions mentioned above. The files are created in the same directory as the definition file. If there already exists a thesaurus with the same name in this directory it is overwritten without warning.

Specify the content of the thesaurus using the following syntax diagram:
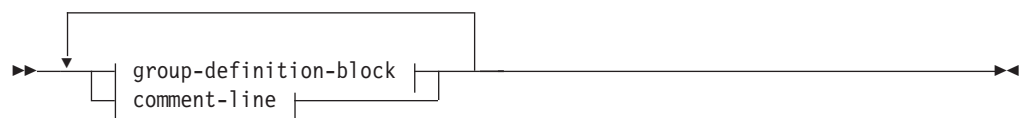
```
>>─┬─┬─ group-definition-block ─┬─┬──────────────────────><
   │ └─ comment-line ───────────┘ │
   └──────────────────────────────┘
```

group-definition-block:

```
├──┤ block-starting-line ├──\n─────────────────────────────>
```

Thesaurus concepts
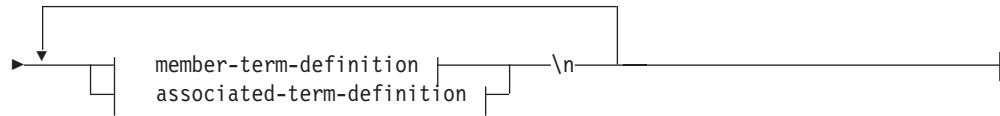
```
    ┌─────────────────────────────────┐
►──┬─▼─┬─ member-term-definition ─────┬──── \n ──────────────────────────┤
   │   └─ associated-term-definition ─┘
   └─┘
```

block-starting-line:

```
├─── :WORDS ──┬────────────────────┬──────────────────────────────────┤
              └─ member-relation ──┘
```

member-relation:

```
├──┬─ :SYNONYM ───────────────────┬────────────────────────────────────┤
   ├─ :RELATED ──────────────────┤
   └─ :RELATED ──( ─ number ─ )───┘
```

member-term-definition:

```
├─── member-term ───────────────────────────────────────────────────────┤
```

associated-term-definition:

```
├──┬─ .RELATED_TO ──────────────────┬─── associated-term ─────────────────┤
   ├─ .SYNONYM_OF ─────────────────┤
   └─ .RELATED_TO ──( ─ number ─ )──┘
```

comment-line:

```
├─── # ─ any-comment ────────────────────────────────────────────────────┤
```
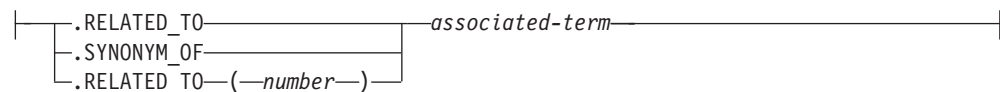
Each member term must be written to a single line. Each associated term must be preceded by the relation name. If the member terms are related to each other, specify a member relation.

The length of member terms and associated terms is restricted to 164 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 but either the 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.

If an application wants to use symbolic names for their thesaurus relations instead of the relation name and number, it must administrate the mapping itself. For example, if the relation OPPOSITE_OF was defined as RELATED_TO(1), the

application has to map this name to the internal relation name RELATED_TO(1). Refer to "Chapter 12. Syntax of search arguments" on page 187 to find out how to use thesaurus expansion in a query.

Chapter 16. Configuration files

This chapter describes the configuration files. These files are automatically generated when a DB2 Text Extender instance is created with the `txicrt` command, or when a client profile is created with the `descrcl` command. These files are generated in code page 819.

You can edit these files to tune your system, however, ensure that you use the correct code page when editing the files. The section names and the option names are case independent. A semicolon is used as a comment delimiter.

Where the option is a Boolean value, the values `TRUE`, `YES`, `ON`, and `1` are considered true regardless of their case; all other values are considered false.

Client configuration file

File name   DESCL.INI

Location   UNIX: DB2TX_INSTOWNERHOMEDIR/db2tx

Windows NT and Windows 2000:
%DMBMMPATH%*instance*\\%DB2INSTANCE%\\db2tx

The updated options become active at the next StartSession function.

Section	Option	Default value	Description
[INSTANCE]	DESWORKCL		Points to a working directory that is used for temporary files.
	DESNLPSCL		Points to the resource directory.
	DESDTDPATHSRV		Points to the directory where the DTDs for the XML documents are stored.
	DESDTDPATHCL		Points to the path where DTD files must be stored. Used for XML files.
[BUFFER]	BUFFERSEGMENTSIZE	32 000	The size of the block segments, in bytes, used for buffering.
	BUFFERSEGMENTCOUNT	3	The maximum number of segments used before buffers are swapped to temporary files. A buffer segment is defined by BUFFERSEGMENTSIZE.

Client configuration file

Section	Option	Default value	Description
[DOCUMENTFORMAT]	USEREXIT		The name of the user exit used to work with unsupported document formats. Specify either a file name if the user exit is stored in a directory which is part of the PATH statement, or a fully-qualified file name. For further information on the user exit for format conversion, see "Using unsupported document formats" on page 26.
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats. TRUE: format recognition is on FALSE: format recognition is off
	UseExitForAllFormats	FALSE	Determines when the user exit for working with document formats not listed in deslsdef.h is called. You must set a value for USEREXIT. TRUE: the user exit is always called. If this value is set, FORMATRECOGNITION is ignored. FALSE: call the user exit for all document formats above the value of EHW_USER_FORMATS.

Server configuration file

File name   DESSRV.INI

Location   UNIX: DB2TX_INSTOWNERHOMEDIR/db2tx/TXINSnnn

Windows NT and Windows 2000:
%DMBMMPATH%*instance*\%DB2INSTANCE%\db2tx\txinsnnn
This file exists only once per server instance.

The updated options become active the next time the server instance is started.

Table 20. Server configuration file options

Section	Option	Default value	Description
[INSTANCE]	DESWORKSRV		Points to a working directory that is used for temporary files.
	DESNLPSSRV		Points to the resource directory.

Table 20. Server configuration file options (continued)

Section	Option	Default value	Description
[DAEMON]	MaxMtEntries	30 on AIX, Windows, OS/390 10 on Solaris	The maximum number of indexes used in parallel at any one time. Decrease this number if you are short of resources, such as semaphores or shared memories. Available resources are platform dependent and therefore the default values are also platform dependent.
	MaxIndexEntries	1000	The maximum number of indexes used. Decrease this number if you are short of resources, such as shared memories.
[BUFFER]	BUFFERSEGMENTSIZE	32 000	The size of the block segments, in bytes, used for buffering. This is used by EhwUpdate.
	BUFFERSEGMENTCOUNT	3	The maximum number of segments used during the index update process before buffers are swapped to temporary files. Increase this number if your document collections contain large documents.
	BUFFERSORTSIZE	4 000 000	The size of the buffer, in bytes, used for sorting temporary work files.
[DOCUMENTFORMAT]	USEREXIT		The name of the user exit used to work with document formats not listed in deslsdef.h. Specify either a file name if the user exit is stored in a directory which is part of the PATH statement, or a fully-qualified file name. For further information on the user exit for format conversion, see "Using unsupported document formats" on page 26.
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats listed in deslsdef.h. TRUE: format recognition is on FALSE: format recognition is off

Server configuration file

Table 20. Server configuration file options (continued)

Section	Option	Default value	Description
	UseExitForAllFormats	FALSE	Determines when the user exit for working with document formats not listed in deslsdef.h is called. You must set a value for USEREXIT. TRUE: the user exit is always called. If this value is set, FORMATRECOGNITION is ignored. FALSE: call the user exit for all document formats above the value of EHW_USER_FORMATS.
[LINGPREC] For all indexes with linguistic or precise as their base type.	UPDATETHRESHOLD	4 000 000	An index update process is split internally into several update and reorganization runs. This value specifies the number of words to be collected in one update step.
	UPDATESLICE	1	The number of update runs that take place before an internal reorganization process starts. An update run is defined by the UPDATETHRESHOLD.
[NGRAM] For all indexes with DBCS support.	UPDATETHRESHOLD	10 000 000	Total size, in bytes, of documents added to an index during one update run. If the threshold is exceeded, a reorganization process is automatically started.
	UPDATESLICE	10 000	The maximum number of documents in a secondary index. This number is checked after each update run. If the number of documents is greater than this value, a reorganization process is automatically started.

Text analysis configuration file

File name   DESTAF.TCF

Location   UNIX: DB2TX_INSTOWNERHOMEDIR/db2tx/TXINSnnn

Windows NT and Windows 2000:
%DMBMMPATH%*instance*\\%DB2INSTANCE%\\db2tx\\txinsnnn
This file exists only once per server instance.

The text analysis configuration file contains sections for update and query processing for each index type except ngram indexes. You should change only the options described here. If you change any of the other entries, this might lead to errors during update or query processing.

Common options

The following options apply to:

- [TAF_Application:UPDATE:PREC)] for the PRECISE index type
- [TAF_Application:UPDATE:LING)] for the LINGUISTIC index type
- [TAF_Application:UPDATE:NORM)] for the PRECISE NORMALIZED index type
- [TAF_Application:QUERY:PREC)]
- [TAF_Application:QUERY:LING)]
- [TAF_Application:QUERY:NORM)]

Option	Default value	Description
FilterSet	None, required	Name of the section in the configuration file containing the rules for stopword filtering, for example, StwUseAllUpCs, StwIgnAllUpCs.
Normalize	No	Specifies whether normalization is carried out on the index. If this is set to yes, normalization converts all characters to lowercase and replaces diacritical characters with base characters. If this is set to No, the entries for NormalizeUmlauts and ConvertAllUpperToLower are ignored.
NormalizeUmlauts	Yes	Specifies whether umlauts are to be replaced, for example, ä with ae. This applies only to documents in German, Danish, Swedish, or Norwegian.
ConvertAllUpperToLower	No	Specifies whether words in all uppercase are converted to lowercase.

[TAF_Filter:STWUSEALLUPCS] [TAF_Filter:STWIGNALLUPCS]

The options in these sections describe the stopword filter sets. The section [TAF_Filter:STWUSEALLUPCS] treats all uppercase words as mixed-case words and the section [TAF_Filter:STWIGNALLUPCS] defines special treatment for all uppercase words.

Option	Default value	Description
FilterType	TAF_FILTER_STW	Section in the configuration file containing the rules for stopword filtering.
MinTokenLength	3	Minimum number of characters required for a token. If the length of the token is smaller than this value, it will not be indexed. This is an integer value ≥ 1.
MinUpperCaseTokenLength	2	The minimum number of characters required for a token in all uppercase. If the length of the token is smaller than this value, it will not be indexed. This is an integer value ≥ 1.

Text analysis configuration file

Option	Default value	Description
UseAlternateTerritories	No	Specifies whether an alternative dictionary for stopword processing is to be used if the language-specific dictionary cannot be found. If this is set to yes, the first stopword dictionary matching the language is used. For example, if the current language is en-gb but there is no en-gb.tsw stopword file, the en-us.tsw stopword file is used. If this is set to no, stopword filtering is not done.

[TAF_AnnotationType:TAF_ThesaurusEntry]

This section sets the defaults for queries requiring thesaurus expansion.

Keyword	Default value	Description
ThesBaseName	desthes	Sets the default name for the thesaurus used for queries requiring query expansion. The name is a character string.
Depth	1	Sets the default path for traversing the thesaurus used for queries requiring thesaurus expansion. This is an integer value ≥ 1.

[TAF_PlugIn:TAF_POE]

To change the default for the language environment that is preloaded when you start the Text Search Engine, you can define a language in the PreloadLanguages option.

Option	Default value	Description
PreloadLanguages	EN-US	Specifies a set of language-specific resources that are to be preloaded at startup. For example, if an application works on documents in Italian, the resources for Italian are loaded. The value of the field takes the form XX-YY, where XX is the language and YY is the territory.

Text analysis configuration file

Chapter 17. Return codes

This chapter lists the codes that are returned by the DB2 Text Extender API in response to a function call. They are listed in alphabetic order.

All DB2 Text Extender API calls return a numeric return code as the C function value. The return codes are defined in the include file DES_EXT.H provided with DB2 Text Extender.

The DB2 Text Extender API intercepts error situations and reports error conditions with a return code.

Applications that call DB2 Text Extender API functions should always check the return code before trying to process any other output parameters. The return codes possible with each call are listed with their parameters in "Chapter 13. API functions for searching and browsing" on page 199.

In some cases, incorrect input such as an obsolete session pointer can cause an abnormal end condition in the API services that cannot be intercepted by DB2 Text Extender.

RC_ALLOCATION_ERROR

Explanation: Cannot allocate storage for internal use.

What to do: Ensure that there is sufficient memory available.

RC_FILE_IO_PROBLEM

Explanation: DB2 Text Extender could not read or write a file.

What to do: Check that there is sufficient disk space and memory available at the server. Check that the environment variables and the text configuration settings are set correct.

RC_INVALID_BROWSE_INFO

Explanation: The browse information returned by DesGetSearchResultTable or by DesGetBrowseInfo and used as input for DesStartBrowseSession is not valid.

What to do: Check whether a programming error overrides the browse information.

RC_INVALID_BROWSE_OPTION

Explanation: The browse option in DesGetSearchResultTable is not valid.

What to do: Ensure that the option is BROWSE or NO_BROWSE.

RC_INVALID_MATCH_OPTION

Explanation: The match options used in DesOpenDocument is not valid.

What to do: Check that the option is FAST or EXTENDED.

RC_INVALID_PARAMETER

Explanation: One of the input parameters is incorrect.

What to do: Read the error message returned by DB2 Text Extender to determine the cause.

RC_INVALID_SEARCH_OPTION

Explanation: The search option in DesGetSearchResultTable is not valid.

What to do: Ensure that the option is DES_TEXTHANDLEONLY, DES_RANK, DES_MATCH, or DES_RANKANDMATCH.

RC_INVALID_SESSION

Explanation: The session pointer specified in the current service call is incorrect or obsolete.

Return codes

What to do: Save any information that can help to find the cause of the error, then end the application.

RC_NO_BROWSE_INFO

Explanation: No browse information is returned by DB2 Text Extender. This is because the search argument resulted in an empty search result. This is not an error.

What to do: No action necessary.

RC_PARSER_INVALID_ESCAPE_CHARACTER

Explanation: The search criteria contains an incorrect escape character. This error is reported if a blank is used as an escape character or if, for one word or phrase, more than one escape character is specified in the search criteria. Example: ESCAPE " " or ESCAPE "#$".

What to do: Check the syntax of the search argument, and try again.

RC_PARSER_INVALID_USE_OF_ESCAPE_CHAR

Explanation: The escape character syntax in the search criteria cannot be interpreted.

What to do: Check the escape character syntax. For example, if $ is the specified escape character, the word or phrase can contain only $$, $_ or $%, where _ and % are the two masking symbols.

RC_PARSER_SYNTAX_ERROR [: position]

Explanation: The search criteria syntax cannot be interpreted. If a position is shown, the position is the character position in the search string where the problem was detected. Sometimes, this is not the exact position, so also check the characters before specified character position.

What to do: Check the syntax of the search argument, by referring to "Chapter 12. Syntax of search arguments" on page 187.

RC_RESULT_TABLE_NOT_EXIST

Explanation: You are trying to store the result of a search in a table that does not exist.

What to do: Create a result table as shown in Figure 17 on page 213.

RC_SE_BROWSER_TIME_OUT

Explanation: The browse process was started but did not respond in an acceptable time. DB2 Text Extender then canceled the pending process.

This error can occur when your system does not have enough storage space or is overloaded.

What to do: Terminate the browse session by calling DesEndBrowseSession, free allocated storage by calling DesFreeBrowseInfo, and try again.

RC_SE_CAPACITY_LIMIT_EXCEEDED

Explanation: The requested function cannot be processed. There is insufficient memory or disk space.

What to do: End the program and check your system's resources.

RC_SE_COMMUNICATION_PROBLEM

Explanation: Communication with the DB2 Text Extender server failed. The error could be caused by a lack of storage space or by an incorrect installation of DB2 Text Extender.

What to do: Save any information that can help to find the error, then end the application.

RC_SE_CONFLICT_WITH_INDEX_TYPE

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF cannot be used with a linguistic index. The default linguistic specification is used as shown in Table 8 on page 194.

What to do: Adapt your application to prevent the specification of query options that conflict with the index type.

RC_SE_DICTIONARY_NOT_FOUND

Explanation: DB2 Text Extender linguistic services cannot find the dictionary files. The query is processed without linguistic support. The dictionary files corresponding to the specified language code(s) are not in the expected path.

What to do: You can continue to make API calls. For UNIX, check that the required dictionary is in the path {DB2TX_INSTOWNERHOMEDIR}/db2tx/dicts. For OS/2, check that the required dictionary is in the path specified in the text configuration setting DB2TX_DATA. If necessary, install the required dictionary.

RC_SE_DOCMOD_READ_PROBLEM

Explanation: When a DB2 Text Extender instance is created, a document models file called desmodel.ini is put in the instance directory. When you create an index, a desmodel.ini file is also put in the index directory IXnnnnnn. This document models file could not be read.

What to do: Check that a document models file exists and that it is in the correct directory.

RC_SE_DOCUMENT_NOT_ACCESSIBLE

Explanation: The requested text document is found, but is currently not accessible.

What to do: Check whether the document is accessed exclusively by another task or user.

RC_SE_DOCUMENT_NOT_FOUND

Explanation: The requested text document was not found. The most likely cause is that a text document has been deleted from storage, but has not yet been removed from the DB2 Text Extender index. This can also occur if you are trying to browse a document identified by a damaged handle.

What to do: In most cases, you can ignore this return code. It will no longer be displayed after the next index update.

If it is persistent, check that your application program is passing the found handle correctly for browsing.

RC_SE_EMPTY_INDEX

Explanation: The DB2 Text Extender index corresponding to the handle column addressed by the search request is empty. Either no text documents have been added to this index or all text documents have been removed from it.

This can occur when a text column has been enabled, but the documents in the column have not yet been indexed. That is, you specified in the ENABLE TEXT COLUMN command to create the index later, at a time determined by the periodic indexing settings.

This can also occur when a text table has been enabled to create an empty common index for all text columns, but none of the text columns has been enabled.

What to do: If ENABLE TEXT TABLE has been used to create an empty common index for all text columns, run ENABLE TEXT COLUMN for at least one of the text columns that contain text to be searched. In this command, you can determine whether the index is created immediately, or at a time determined by the periodic indexing settings.

Run GET INDEX STATUS to check that the index was built successfully.

RC_SE_EMPTY_QUERY

Explanation: The specified search criteria was analyzed and processed linguistically by DB2 Text Extender. Either a programming error caused a query to be made containing no search terms, or all search terms were stop words (words not indexed by DB2 Text Extender) that are removed from a query. The result was no search terms.

What to do: Reword the query. If the problem persists, check for a programming error.

RC_SE_END_OF_INFORMATION

Explanation: This is not an error. The end of the document has been reached. No further information is available for DesGetMatches.

What to do: Use this return code to end the iterative processing of the document with DesGetMatches.

RC_SE_FUNCTION_DISABLED

Explanation: The requested function called a DB2 Text Extender function that has been prevented by the administrator.

What to do: Ask your administrator for assistance. It may be necessary to stop and restart DB2 Text Extender (txstop/txstart).

RC_SE_FUNCTION_IN_ERROR

Explanation: The requested function has been locked due to an error situation on the DB2 Text Extender server. The API call cannot be processed.

What to do: Check the index status. Check the available space in the index directory. Reset the index status and retry the command.

RC_SE_INCORRECT_HANDLE

Explanation: A handle specified in an input parameter such as *browse session handle* is not valid. It must be a handle that was returned by a previous call and that is not obsolete.

What to do: Save any information that can help to find the cause of the error, then terminate the session by calling DesEndBrowseSession.

Check whether a programming error produced an incorrect handle.

Return codes

RC_SE_INDEX_DELETED

Explanation: The DB2 Text Extender index being accessed is deleted.

What to do: Contact the DB2 Text Extender administrator to recreate the index.

RC_SE_INDEX_NOT_ACCESSIBLE

Explanation: The DB2 Text Extender index cannot be accessed and the current call cannot be processed.

What to do: Ask the DB2 Text Extender administrator to check the accessibility of the index.

RC_SE_INDEX_SUSPENDED

Explanation: DB2 Text Extender received a request relating to a index that was suspended from another session or from the current session.

What to do: Ask the DB2 Text Extender administrator to check the status of the index.

RC_SE_INSTALLATION_PROBLEM

Explanation: DB2 Text Extender has encountered an installation problem.

What to do: Check the current setting of the environment variables DB2INSTANCE, DB2TX_INSTOWNER, DB2TXINSTOWNERHOMEDIR. Use `descfgcl -d` and `descfgsv -d -i txinsnnn` to check your search service configuration.

RC_SE_IO_PROBLEM

Explanation: An error occurred when the server attempted to open or read one of its index files. This can be due to one of the following:

> An unintentional action by the administrator, such as the deletion of a DB2 Text Extender index file
>
> Incorrect setting in the text configuration.

What to do: Terminate the application. Check with the administrator that:

- All files of the current DB2 Text Extender index exist
- The text configuration settings are correct.

RC_SE_LS_FUNCTION_FAILED

Explanation: A function that accessed the database to retrieve text documents for browsing failed. Either the database is no longer accessible to the user, or the user is not authorized for the text table.

What to do: Check that the input to the function, such as the user ID, is correct. Check that the database is accessible and that the user is authorized for the task.

RC_SE_LS_NOT_EXECUTABLE

Explanation: A function that is trying to access the database to retrieve text documents for browsing cannot be executed.

What to do: Check that DB2 Text Extender is installed correctly. If the problem persists, contact your IBM representative.

RC_SE_MAX_OUTPUT_SIZE_EXCEEDED

Explanation: An unusually large number of matches have been found. The size of the browse information has exceeded the maximum that can be handled. The request cannot be processed.

What to do: Either make the query more specific or ensure that more system memory is available.

RC_SE_MAX_NUMBER_OF_BUSY_INDEXES

Explanation: The requested function has been prevented by the search service, because the maximum number of indexes is currently active.

What to do: Reissue the function call after a short period of time. In general, the problem is only temporary.

RC_SE_NO_DATA

Explanation: This is not an error. No text document matches the search criteria. If you request browse information, no browse information is returned. No storage is allocated for the browse information.

What to do: No action is necessary.

RC_SE_NOT_ENOUGH_MEMORY

Explanation: There is not enough storage space on the client or on the server system. The current request cannot be processed.

What to do: Release storage space and end the application.

RC_SE_PROCESSING_LIMIT_EXCEEDED

Explanation: The current search request exceeded the maximum result size or the maximum processing time

specified for your client/server environment. The request was canceled.

What to do: Make the search request more specific. Consider increasing the maximum processing time.

RC_SE_QUERY_TOO_COMPLEX

Explanation: The specified query is too complex.

What to do: Adapt your application to prevent excessive use of masking characters and synonyms.

Excessive use of masking symbols or excessive use of the SYNONYM option can expand a query to a size that cannot be managed by DB2 Text Extender.

RC_SE_REQUEST_IN_PROGRESS

Explanation: A DB2 Text Extender browse API service was called while another browse API request was active for the same session.

What to do: End the session by calling DesEndBrowseSession and free storage by calling DesFreeBrowseInfo.

The DB2 Text Extender browse API does not support concurrent access to the same browse session.

All applications running concurrently in the same process should handle their own browse sessions.

RC_SE_SERVER_BUSY

Explanation: The DB2 Text Extender client cannot currently establish a session with the requested DB2 Text Extender server, or the DB2 Text Extender server communication link was interrupted and cannot be re-established.

The DB2 Text Extender server has been started correctly, but the maximum number of parallel server processes was reached.

What to do: If this is not a temporary problem, change the communication configuration on the server.

RC_SE_SERVER_CONNECTION_LOST

Explanation: The communication between client and server was interrupted and cannot be re-established.

The DB2 Text Extender server task may have been stopped by an administrator or the server workstation may have been shut down.

What to do: Check whether either of these conditions have occurred, and have them corrected.

RC_SE_SERVER_NOT_AVAILABLE

Explanation: The DB2 Text Extender API services could not establish a session with the requested DB2 Text Extender server.

The DB2 Text Extender server may not have been started.

What to do: Check that the DB2 Text Extender server has been started correctly. If the error persists, there is an installation problem.

RC_SE_STOPWORD_IGNORED

Explanation: This informational code is returned when the specified query contained at least one search term consisting only of stop words. The search term was ignored when processing the query.

What to do: You can continue to issue API calls. Avoid using stop words in DB2 Text Extender queries.

RC_SE_UNEXPECTED_ERROR

Explanation: An error occurred that could be caused by incorrect installation of DB2 Text Extender.

What to do: End the application, saving any information that may help to find the cause of the error.

RC_SE_UNKNOWN_ERROR

Explanation: An internal error has occurred.

What to do: For UNIX, see the latest entry in the log file, `$HOME/db2tx/desdiag.log`

For NT, see the latest log file entry in `%DMBMMPATH%\instance\db2\db2tx\desdiag.log`

Note that in the NT path, `db2` is the default instance name.

RC_SE_UNKNOWN_INDEX_NAME

Explanation: The name of the text index associated with a text column is part of the handle.

What to do: Ensure that the handle you use as input to DesGetBrowseInfo is correct.

RC_SE_UNKNOWN_SECTION_NAME

Explanation: A specified section name is not part of a model specified in a document models file, or of the default model used.

What to do: Specify a section name that is part of the specified model or the default model.

RC_SE_UNKNOWN_SERVER_NAME

Explanation: The name of DB2 Text Extender server is part of the handle.

What to do: Ensure that the handle you use as input to DesGetBrowseInfo is correct.

RC_SE_WRITE_TO_DISK_ERROR

Explanation: A write error occurred that could be caused by a full disk on the DB2 Text Extender server workstation, or by incorrect installation of DB2 Text Extender.

What to do: End the application, saving any information that may help to find the cause of the error. Check that there is enough disk space available at the server.

RC_SQL_ERROR_WITH_INFO

Explanation: An SQL error occurred. An error message is returned.

What to do: Check the error message returned by DB2 Text Extender for more information, such as the SQL error message, SQLState and native SQL error code.

RC_SQL_ERROR_NO_INFO

Explanation: An SQL error occurred. No error message is returned.

RC_TEXT_COLUMN_NOT_ENABLED

Explanation: The specified handle column is not a column in the table you specified.

What to do: Check whether the handle column name you specified is correct. Ensure that the text column in that table has been enabled.

Chapter 18. Messages

This chapter describes the following:

- **SQL states returned by DB2 Text Extender functions:** These messages can be displayed when you use DB2 Text Extender functions.
- **Messages from the DB2TX command line processor:** These messages can be displayed when you enter commands using the command line processor DB2TX. Each message number is prefixed by DES.

SQL states returned by DB2 Text Extender functions

The SQL functions provided by DB2 Text Extender can return error states. Example:

```
SQL0443N User-defined function
"DB2TX.CONTAINS" (specific name "DES5A")
has returned an error SQLSTATE with
diagnostic text "Cannot open message file".
SQLSTATE=38702
```

The messages in this section are arranged by SQLSTATE number.

01H10   **The file** *file-name* **cannot be opened.**

What to do: Ensure that the file exists, and that the DB2 instance name has the necessary permissions to open it.

01H11   **The text handle is incomplete**

Explanation: An attempt was made to use a handle that has been initialized, but not completed. A partial handle was created using INIT_TEXT_HANDLE containing preset values for the document language and format. However, the handle has not been completed by a trigger.

What to do: Use only handles that have been completed. If the handle concerned is stored in a handle column, enable or reenable its corresponding text column.

01H12   **Search arguments too long. The second argument was ignored.**

Explanation: The REFINE function was used to combine two search arguments, but the combined length of the search arguments is greater than the maximum allowed for a LONG VARCHAR. The REFINE function returns the first search argument instead of a combined one.

What to do: Reduce the length of one or both search arguments, then repeat the query.

01H13   **A search argument contains a stopword.**

Explanation: The specified query contains at least one search term consisting only of stop words. The search term was ignored when processing the query.

What to do: Avoid using stop words in DB2 Text Extender queries.

01H14   **A language dictionary for linguistic processing is missing.**

Explanation: DB2 Text Extender linguistic services cannot find the dictionary files. The query is processed without linguistic support. The dictionary files corresponding to the specified language code(s) are not in the expected path.

What to do: For UNIX, check that the required dictionary is in the path `{DB2TX_INSTOWNERHOMEDIR}/db2tx/dicts`. For OS/2, check that the required dictionary is in the path specified in the text configuration settings. If necessary, install the required dictionary.

01H15   **A linguistic search term specification does not match the index type.**

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF should not be used with a linguistic index. The default linguistic specification is used as shown in Table 8 on page 194.

What to do: Adapt your application to prevent the

SQL states returned by DB2 Text Extender functions

specification of query options that conflict with the index type.

38700   **The Text Extender library is not current.**

Explanation: An attempt was made to use a handle that can be interpreted only by a later version of DB2 Text Extender.

What to do: Ensure that the path to the current library version is set correctly, and that you have the necessary permissions to access it.

Look in the DB2 catalog view SYSCAT.FUNCTIONS, in the IMPLEMENTATION column, for the function that caused the problem.

38701   *tracefile* **Cannot open this trace file.**

Explanation: An attempt was made to use a trace function that writes to the file DB2TX_TRACEFILE in the directory DB2TX_TRACEDIR. Either the file does not exist, cannot be found, or the necessary permissions for the file are not available.

38702   **Cannot open message file** *message-file***.**

Explanation: A situation occurred that caused DB2 Text Extender to attempt to return a message. The file containing the messages either does not exist or cannot be found, or the necessary permissions for the file are not available.

What to do: Ensure that the file exists, that the path is set correctly, and that you have the necessary permissions to open the file.

38704   **The format of the text handle is incorrect.**

Explanation: A handle having an incorrect format was used as an argument for a DB2 Text Extender function.

What to do: Ensure that the handle was not produced by INIT_TEXT_HANDLE.

38705   *udfname* **Incorrect UDF declaration.**

Explanation: The specific name of a DB2 Text Extender function has been changed in the script where the functions are declared. DB2 Text Extender function names can be changed, but not their specific names.

What to do: Check the script DESCVDF.DDL that contains the DB2 Text Extender function declarations, to ensure that the correct names are still being used. Check the names against those in the original distribution media.

38706   *attribute* **Cannot recognize this attribute value.**

Explanation: An attempt was made to set a CCSID, format, or language to an unknown value.

What to do: Refer to "Chapter 4. Planning for your search needs" on page 23 for the correct values.

38707   **The requested function is not yet supported.**

Explanation: The specified function is not yet supported.

What to do: Check the specified function.

38708   *return code*

Explanation: An error occurred while processing the search request.

What to do: Refer to the description of the return code in "Chapter 17. Return codes" on page 251.

38709   **Not enough memory available.**

Explanation: Not enough memory is available to run the DB2 Text Extender function.

What to do: Close any unnecessary applications to free memory, then try again.

38710   *errornumber* **Cannot access the search results.**

Explanation: An error occurred while attempting to read the list of found documents (result list) returned by the search service.

What to do: Try repeating the search. If this is not successful, restart the search service. If the problem persists, report it to your local IBM representative, stating the error number.

38711   **Severe internal error.**

Explanation: A severe error occurred.

What to do: Report the error to your local IBM representative, stating the circumstances under which it occurred.

38712   *indexname* **Incorrect handle in this text index.**

Explanation: A handle has been damaged.

What to do: Use UPDATE INDEX to rebuild the index.

38714   **Shorten DB2TX_INSTOWNERHOMEDIR environment variable.**

Explanation: The name of the home directory of the instance owner must be no longer than 256 characters.

What to do: Use links to reduce the length of the directory name.

38717   **The specified thesaurus could be found.**

Explanation: The specified thesaurus cannot be found.

What to do: Check the specified thesaurus name.

38718   **The specified relation name could not be found in the thesaurus.**

Explanation: The specified relation does not exist in the specified thesaurus.

What to do: Ensure that the specified relation exists.

38719   **A search processing error occurred. Reason code:** *rc***.**

Explanation: The search could not be made due to the specified reason.

What to do: Try to solve the problem reported by the reason code. If the specified reason is not helpful and no further information is found in the `desdiag.log` file, create a trace and report the information to your local IBM representative.

38720   **A shared memory attach error occurred.**

Explanation: The system is unable to get access to shared memory.

What to do: Check your system configuration and increase shared resources, or check the current shared resource usage (ipcs) and clean up resources that are no longer needed.

38721   **A semapore creation/access error occurred.**

Explanation: The system is unable to create or get access to a semaphore.

What to do: Check your system configuration and increase shared resources, or check the current shared resource usage (ipcs) and clean up resources that are no longer needed.

38722   **A search process didn't return.**

Explanation: An error occurred while processing the search request.

What to do: Verify your system configuration `descfgcl` and check if all nodes are up and running.

38723   **The index CCSID and query CCSID do not match.**

Explanation: The database CCSID used for the query string is not the same as the CCSID of the text index.

What to do: Disable the text index and recreate it using the CCSID of the database.

38724   **The section or model name is incorrect.**

Explanation: The specified section or model name in the query is incorrect.

What to do: Check the section or model name.

38726   **A model-file read error occurred.**

Explanation: The model-definition file was not found or cannot be opened.

What to do: Check that the model-definition file exists in the index directory.

Messages from DB2 Text Extender

Each message has a message identifier that consists of a prefix (DES), the message number, and a suffix letter. The suffix letter indicates how serious the occurrence is that produced the message:

I   Information message

W   Warning message

N   Error (or "negative") message

C   Critical error message.

DES0001N   **Incorrect number of arguments for the db2txinstance command.**

Explanation: The `db2txinstance` command needs two arguments.

What to do: Enter the command again with these arguments:

`db2txinstance instanceName db2InstanceName`

Messages from DB2 Text Extender

where *instanceName* is the login name of an existing UNIX user that is being assigned as the owner of this instance, and *db2InstanceName* is the login name of the owner of the corresponding DB2 instance.

DES0002N Invalid instanceName.

Explanation: The specified instance name must be the login name of an existing UNIX user.

What to do: Correct the instance name, or select an existing UNIX user, or create a UNIX user to be the instance owner.

Enter the `db2txinstance` command again as follows:

`db2txinstance instanceName`

where *instanceName* is the login name of the selected UNIX user.

DES0004N The specified instance already exists. The command cannot be processed.

Explanation: The *instanceName* specifies the login name of a UNIX user that is the owner of the instance. This instance owner already has a `db2tx` directory in the home directory.

What to do: To create the instance, remove the existing instance and then try the command again.

DES0005N The installation message catalog cannot be found.

Explanation: The message catalog required by the installation scripts is missing from the system; it may have been deleted or the database products may have been loaded incorrectly.

What to do: Verify that the db2tx_01_01_0000.client product option is installed correctly. If there are verification errors, reinstall the option.

DES0015W A linguistic search term specification does not match the index type.

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF should not be used with a linguistic index. The default linguistic specification is used as shown in Table 8 on page 194.

What to do: Adapt your application to prevent the specification of query options that conflict with the index type.

DES0016W A language specification is not supported for the current index type.

Explanation: The language you have specified is not supported for the specified index type.

What to do: See the documentation for a list of supported languages for the index type.

DES0017W Feature extraction has not been enabled.

Explanation: You used a feature search argument in your query but the index was not build with index option FEATURE_EXTRACTION.

What to do: Change the index option to FEATURE_EXTRACTION.

DES0018W *option* is not supported for the current index type.

Explanation: You requested a search option that is not supported for the current index type and index option.

What to do: Check which index type or index option supports the requested search option. See Table 8 on page 194.

DES0121N Memory could not be allocated (malloc failed).

Explanation: No storage could be reserved for the application.

What to do: Increase the paging space.

DES0333N The client cannot establish a session with the requested server.

Explanation: The DB2 Text Extender client cannot establish a session with the requested server.

What to do: Check that the DB2 Text Extender server has been started. If not, run TXSTART.

DES0377N A text index file I/O problem occurred.

Explanation: DB2 Text Extender cannot access the text index. This can happen if the DIRECTORY setting in the text configuration points to an invalid directory.

What to do: Check the text configuration settings.

DES0500N IBM Text Search Engine (5722DE1, option 3) is not properly installed.

Explanation: IBM Text Search Engine is a prerequisite license program option to DB2 Text Extender. It is not installed on the system.

What to do: Install option 3 of license program 5722DE1, then try again.

DES0700N The node number value *'node'* is not contained in the node group definition.

Explanation: The specified node number is invalid.

What to do: Check the DB2 node number.

DES0701N **The node number value** *'node'* **is out of range.**

Explanation: The specified node number is invalid.

What to do: Check the DB2 node number.

DES0704N **Format** *'format'* **requires the specification of an index property.**

Explanation: The document format is not compatible with the index type information.

What to do: Specify an index property that is compatible with the document format.

DES0705N **The specified document model name** *'model'* **was not found in the model definition file.**

Explanation: The document model name was not found in the model definition file. Not that the model name is case-sensitive.

What to do: Use a model name that is specified in the model definition file.

DES0706N **The model definition file cannot be accessed on the DB2 Text Extender server.**

Explanation: The model definition file was not found or cannot be opened.

What to do: Check that the model definition file exists.

DES0707N **Format** *'format'* **does not support the specified index property.**

Explanation: The document format does not support the index property.

What to do: Specify an index property that is compatible with the document format.

DES0709W **The dictionary for the specified language is not installed.**

Explanation: DB2 Text Extender cannot find the dictionary files.

What to do: Install, or reinstall the dictionary for the specified language.

DES0710N **A null pointer is not allowed for parameter** *'parameter'*.

Explanation: No value is specified for *parameter*.

What to do: Specify a value for the parameter.

DES0711N **An internal Text Extender error occurred. Diagnosis information:** *message*.

Explanation: An internal processing error occurred.

What to do: Check the diagnostic message to solve the problem. If the internal error was not caused by an installation problem, additional information may be in the desdiag.log file or in a created trace file. If this does not help, collect the available information and call your IBM service representative.

DES0712N **Parameter** *'parameter'* **is too long.**

Explanation: The specified parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0713N **The length of parameter** *'parameter'* **is incorrect: %d1.**

Explanation: The specified parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0714N **Specify parameter** *parameter* **either directly or in the configuration table.**

Explanation: CCSID, format, or language was not specified, and there is no text configuration setting for this value.

What to do: Either specify the missing parameter directly in the ENABLE TEXT COLUMN command, or set a value in the text configuration settings.

DES0715N **Data type** *schema.type* **is not supported for text data.**

Explanation: *schema.type* is the schema name and type name of the text column or the result of an access function. The data type for a text column is not supported by DB2 Text Extender. It must be CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB. If this is not the case, you must provide an access function whose input is the data type of the text column and whose output is VARCHAR, LONG VARCHAR, or LOB.

What to do: If *schema.type* is a text column type, you must register an access function with a result of type VARCHAR, LONG VARCHAR, or LOB. If *schema.type* is the result of an access function, it cannot be used. Provide an access function with a result of the required type.

Messages from DB2 Text Extender

DES0716N Format *format* **is not supported.**

Explanation: *format* is a format that is not supported by Text Extender.

What to do: Check the list of supported formats in "Which document formats are supported" on page 24.

DES0717N Language *language* **is not supported.**

Explanation: *language* is a language that is either not supported by DB2 Text Extender, or the selected language is not supported by the specified index type.

What to do: Check the list of supported languages in Table 5 on page 43.

DES0718N CCSID *ccsid_value* **is not supported.**

Explanation: You specified an invalid CCSID value.

What to do: See the documentation for a list of supported CCSIDs.

DES0719N **A call to the DB2 Text Extender program** *program* **failed with return code** *rc*.

Explanation: An error may have occurred during installation. The return codes are listed in file DES_EXT.H.

What to do: Check if the installation was successful. Check that the environment variables such as DB2TX_INSTOWNER and DB2TX_INSTOWNERHOMEDIR are set correctly.

DES0720N **The access function** *schema.function* **is not registered in the database.**

Explanation: The name of the function is either incorrect or has not been registered with the database.

What to do: Check the name of the access function. If it is correct, check that the function is known to the database system. Use the CREATE FUNCTION to register the access function with the database.

DES0721N **The database is inconsistent; a DB2 Text Extender catalog view is missing.**

Explanation: One of the DB2 Text Extender catalog views is not in the database.

What to do: Use the DISABLE DATABASE command to remove the remaining catalog views, then enter ENABLE DATABASE again. The index data is lost; reindex the text documents.

DES0722N Table *schema.table* **is not a base table in the database.**

Explanation: Either the table does not exist in the database or it is a result table or a view. A text column must be in a base table before it can be enabled for DB2 Text Extender.

What to do: Ensure that the table name is correct, and that it is a base table.

DES0723N **The creation of an index for the handle column** *handlecolumnname* **in table** *schema.table* **failed.**

Explanation: A text index could not be created for the handle column.

What to do: Use `txstatus` to check the status of the server. If the services on the server are running correctly, use DISABLE TEXT COLUMN or DISABLE TEXT TABLE to get a consistent state again. Then enable the text column again using ENABLE TEXT COLUMN or ENABLE TEXT TABLE.

DES0724N **An entry in the TextIndices catalog view for the handle column** *handlecolumn* **in table** *schema.table* **is missing.**

Explanation: The TextIndices catalog view is damaged.

What to do: Use DISABLE TEXT COLUMN or DISABLE TEXT TABLE to get a consistent state again. Then enable the text column using ENABLE TEXT COLUMN or ENABLE TEXT TABLE.

DES0725N **Request rejected by server. Licence check failed with message ID '%s1'.**

Explanation: No valid license key was found for DB2 Text Extender.

What to do: Request a licence key, install and then retry.

DES0727N Column *column* in table *schema.table* **is already enabled.**

Explanation: This message can occur if the table has been dropped and then recreated using the same text column, without first disabling the column.

What to do: Disable the column, then try again.

DES0728N Column *column* **does not exist in table** *schema.table*.

Explanation: You are trying to enable a text column that does not exist.

What to do: Change the table name or the column name, then try again.

DES0729N   **Handle column** *handlecolumn* **does not exist in table** *schema.table***.**

Explanation: You are trying to use a handle column that does not exist.

What to do: Use the GET STATUS command to check if the handle column exists, and that its name has been specified correctly.

DES0730N   **Table** *schema.table* **is already enabled as a common-index table.**

Explanation: You are trying to enable a table that has already been enabled as a common-index table.

What to do: Either continue without enabling the table, or run the DISABLE TEXT TABLE command to disable the table before enabling it again.

DES0731N   **Table** *schema.table* **is not enabled for Text Extender; it cannot be disabled.**

Explanation: You are trying to disable a table that has not been enabled.

What to do: Check the table name.

DES0732N   **The update frequency is incorrect near location** *location***; expected was** *parameter***.**

Explanation: The *parameter* specification for the Update Frequency was not correct.

What to do: Check the update frequency parameter and reenter the command.

DES0733N   **Table** *schema.table* **contains an enabled column; it cannot be enabled as a common-index table.**

Explanation: This table contains a text column that already has its own index. You cannot create a common index for all the text columns while this individual index exists.

What to do: Use DISABLE TEXT COLUMN to disable the enabled columns, then enter the ENABLE TEXT TABLE command again.

DES0734N   **Handle column** *handlecolumn* **belongs to the partial-text table** *schema.table***; it cannot be disabled separately.**

Explanation: You cannot disable a single text column in a table that was enabled as a partial-text table.

What to do: Disable the complete partial-text table.

DES0736N   *handlecolumn* **is already a handle column in table** *schema.table***.**

Explanation: You are trying to use an existing handle column name.

What to do: Reenter the command, using a different name for the handle column.

DES0737N   **Table** *schema.tablename* **is enabled as a common-index table with STORAGE option** *storage_option***.**

Explanation: It is not possible to enable a common index table for external files.

What to do: If you want to enable a table for external files, use a multi-index table.

DES0738N   **Access function** *schema.function* **has incorrect parameters.**

Explanation: The input or output parameters of *schema.function* are incorrect.

- There can be only one input parameter, and it must be of the data type of the text column to be enabled.
- The output parameter must be of type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB.

DES0739W   **The index update program for table** *schema.table***, handle column name** *handlecolumn***, could not be started.**

Explanation: The program that updates indexes could not be started. An error may have occurred during installation.

What to do: Check if the installation was successful. Check that the environment variables such as DB2TX_INSTOWNER and DB2TX_INSTOWNERHOMEDIR are set correctly.

DES0740N   **Handle column** *'column'* **can be reused only for a text column of type** *columntype***.**

Explanation: The handle column has already been used for another handle column type.

What to do: Specify a new handle column name.

DES0741N   **The program or file** *parameter* **was not found or could not be started.**

Explanation: The ENABLE DATABASE or DISABLE DATABASE command could not open the file *parameter*. An error may have occurred during installation.

What to do: Check if the installation was successful.

Messages from DB2 Text Extender

DES0742N **This table uses column indexes. Specify a handle column in the command.**

Explanation: The specified table is enabled as multi index table. To work with a specific column specify the related handle column.

What to do: Specify a handle column.

DES0745N **The DB2TX instance owner** *instance-owner* **is not a valid user ID.**

Explanation: The environment variable DB2TX_INSTOWNER does not contain a valid user ID.

What to do: Correct the environment variable.

DES0747N **The current CCSID is not supported for index type** *index_type***.**

Explanation: You specified a CCSID that is not supported for the requested index type.

What to do: See the documentation for a list of supported CCSIDs.

DES0748N **The commit count value** *'commitcount'* **is not supported by DB2 Text Extender.**

Explanation: The specified commit count value is not supported.

What to do: Specify a valid commit count value.

DES0749N **The update index value** *'indexvalue'* **is not known by DB2 Text Extender.**

Explanation: The specified 'update index' value is invalid.

What to do: Specify a valid update index value.

DES0750N **The index type value** *'indextype'* **is not known by DB2 Text Extender.**

Explanation: The specified index type value is invalid.

What to do: Specify a valid index type value

DES0751N **You do not have the authorization to perform the specified operation.**

Explanation: You do not have the required database administrator authorization to do this operation.

What to do: Have this operation done by a database administrator.

DES0756N **The database is not enabled for Text Extender.**

Explanation: The database must be enabled before this command can be run.

What to do: Run ENABLE DATABASE, then resubmit the command.

DES0763N **The update frequency is incorrect.**

Explanation: The specified 'update frequency' value is invalid.

What to do: Specify a valid 'update frequency'.

DES0765N **The database is already enabled for Text Extender.**

Explanation: You are trying to enable a database that is already enabled.

What to do: Either continue without enabling the database, or use DISABLE DATABASE to disable the database before enabling it again.

DES0766N **An action has caused the maximum row size of the table or a temporary table to be exceeded.**

Explanation: The ENABLE TEXT COLUMN command adds a handle column to the table. If the table is already large, this can cause the row size of the table to exceed the maximum value of 4005.

The ENABLE TEXT COLUMN command also creates a temporary table whose size is proportional to the number of text columns that are already enabled. If many text columns are already enabled, the size of the temporary table may exceed the maximum value.

What to do: Use the ENABLE TEXT COLUMN only on tables that do not cause this limit to be exceeded.

DES0769W **Warning: Index characteristic have been specified but will be ignored. Table** *'tablename'* **is a common-index table.**

Explanation: The specified table is a common index table therefore no index characteristics can be specified.

What to do: No action required.

DES0770N **The environment variable** *env-variable* **is not defined.**

Explanation: A parameter for a command was not specified and the system tried to read the default value from the environment variable *env-variable*, but this environment variable is not defined.

What to do: Define the required environment variable.

DES0774N **The value length** *length* **for variable** *'variable'* **is out of range.**

Explanation: The value length of the parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0775N **The index directory value** *'directory'* **is incorrect.**

Explanation: The index directory value is incorrect, may be the directory length is incorrect.

What to do: Specify a valid index directory value.

DES0776N **The table space name** *'tablespace'* **is incorrect.**

Explanation: The specified table space name is incorrect, may be the table space value length is incorrect.

What to do: Specify a valid table space value.

DES0777N **The table space** *tablespace* **is not known by the database management system.**

Explanation: The specified table space is not known to the database system.

What to do: Check that the specified table space exists in the database.

DES0778N *'tablespace'* **is not a REGULAR table space. It was created with keyword** *'keyword'*.

Explanation: The data type for the specified table space is not supported.

What to do: Specify a regular table space.

DES0779I **Indexing has been started successfully. To check indexing status use 'GET INDEX STATUS'.**

Explanation: The indexing program has started. You can use the 'GET INDEX STATUS' command to check the status of the indexing process.

What to do: Check output of GET INDEX STATUS command.

DES0780I **Index reorganization has been started successfully. To check the index status use 'GET INDEX STATUS'.**

Explanation: The reorganization program has started. You can use the 'GET INDEX STATUS' command to check the status of the reorganization process.

What to do: Check output of GET INDEX STATUS command.

DES0789W **Warning: The partitioning map of the current nodegroup has been updated, please call the TXNCHECK utility.**

Explanation: The partitioning map of the current nodegroup has been updated.

What to do: Use the TXNCHECK command.

DES0800I **The** *command* **command completed successfully.**

Explanation: The specified command completed successfully.

What to do: No action required.

DES0810N **Closing quotation mark is missing.**

Explanation: A quotation mark has been found, but the second quotation mark is missing.

What to do: Check the syntax of the command and try again.

DES0811N *"token"* **is unexpected. Check the index characteristics or the text information.**

Explanation: The index characteristics or the text information is incorrect.

What to do: Check the syntax and try again.

DES0812N **Table** *schema.table* **does not exist or is not enabled for DB2 Text Extender.**

Explanation: While running the GET command, either the name of a database table is incorrect, or the table does not exist, or it has not yet been enabled.

What to do: If the table name is correct, use GET STATUS to check that it has been enabled. Enable the table and try again.

DES0813N **Table** *schema.table* **does not exist or is not enabled for DB2 Text Extender or does not contain a handle column** *column*.

Explanation: While running the GET command, no entries for the handle column are found in the table. If the table exists, it is not enabled or it does not contain a handle column.

What to do: If the table name is correct, use GET STATUS to check that it has been enabled. Enable the table and try again.

DES0814N **Table** *tablename* **does not exist or is not enabled for DB2 Text Extender or no text column is enabled within this table.**

Explanation: The specified table is not enabled for DB2 Text Extender.

Messages from DB2 Text Extender

What to do: Enable the table for DB2 Text Extender.

DES0815N **Empty quotes "" found. A name is expected inside the quotes.**

Explanation: Two consecutive quotation marks were found with no text between them.

What to do: Check the syntax and try again.

DES0816N **The word** *"token"* **is unexpected. Use one of the keywords** *keyword* **or** *keyword*.

Explanation: An unexpected token was found.

What to do: In the command, use one of the keywords given in the message.

DES0817N *"token"* **is unexpected. Use the keyword** *keyword*.

Explanation: An unexpected token was found.

What to do: In the command, use the keyword given in the message.

DES0818N **Unexpected end of command. The keyword** *keyword* **is expected.**

Explanation: A keyword is missing.

What to do: In the command, use the keyword given in the message.

DES0819N **Unexpected end of command. One of the following keywords is expected:** *keyword* **or** *keyword*.

Explanation: A keyword is missing.

What to do: In the command, use one of the keywords given in the message.

DES0820N **Index option** *index_option* **is not supported for index type** *index_type*.

Explanation: You specified an index option that is not supported for the given index type.

What to do: See the documentation for supported index options for the given index type.

DES0821N **The name** *"token"* **is too long. Only** *nn* **characters are allowed for** *variable* **names.**

Explanation: A name is too long.

What to do: Specify a name having an acceptable length.

DES0822N **The command contains an unrecognized token** *"token"*. **End of command is expected.**

Explanation: End of command found, but a keyword is expected.

What to do: Check the syntax of the command and try again.

DES0823N **A table name is expected following** *"schema."*.

Explanation: A table name or a function name is missing after the ".".

What to do: Check the syntax of the command and try again.

DES0824N **Unexpected end of command; a** *keyword* **is required.**

Explanation: The keyword in the message is missing from the syntax.

What to do: Check the syntax of the command and try again.

DES0825N *'alias'* **is not a known database alias name. Only** *nn* **characters are allowed.**

Explanation: The specified database alias name is unknown to the database system.

What to do: Check that the specified alias name is valid.

DES0826N **Database alias** *alias* **must not be in quotation marks.**

Explanation: The name in the message has been interpreted as a database alias. It must not be in quotation marks.

What to do: Check the syntax of the command and try again.

DES0827N **The CCSID** *"ccsid"* **is not supported.**

Explanation: The CCSID is not one of those supported by DB2 Text Extender.

What to do: Refer to the documentation for a list of the supported CCSIDs.

DES0829N **The user name** *userid* **must not be in quotation marks.**

Explanation: You entered a user name in quotation marks.

What to do: Remove the quotation marks.

DES0830N   **Parameter** *"parameter"* **in the** *enable/disable*
DATABASE command not recognized.
End of command expected.

Explanation: The commands ENABLE DATABASE
and DISABLE DATABASE do not take parameters.

What to do: Enter the command again without
parameters.

DES0831N   **Unexpected end of command. The table**
name is missing.

Explanation: The command requires a table name.

What to do: Enter the appropriate table name.

DES0832N   **Unexpected end of command. The**
database name is missing.

Explanation: The command requires a database name.

What to do: Enter the appropriate database name.

DES0833N   **Unexpected end of command. The**
column name is missing.

Explanation: The command requires a column name.

What to do: Enter the appropriate column name.

DES0899N   **Unknown DB2TX command:** *command*.

Explanation: The specified command is not supported
by DB2 Text Extender

What to do: Type **db2tx ?** to get a list of the
commands.

DES0971N   **Index directory can be specified once**
without node specification or multiple
times with node specification.

Explanation: The specification of the index
directory/ies is not correct.

What to do: Check the specification of the index
directory. You can specify one index directory without a
node specification or multiple index directories with
node specification.

DES0972N   **The node specification is not correct. An**
unsigned digit value is expected.

Explanation: A non digit value is specified for the
node number.

What to do: Specify an unsigned digit value for the
node number.

DES0973N   **The node specification is not correct.**
One or more unexpected characters
found before right parenthesis.

Explanation: The node specification is syntactical
incorrect.

What to do: Check the syntax of the node
specification and try again.

DES0974N   **The sequence of the node numbers in a**
TO clause is not correct (second node
smaller than first node).

Explanation: The node specification is syntactical
incorrect.

What to do: Check the syntax of the node
specification and try again.

DES0975N   **The syntax for the specification of the**
node information is not correct.

Explanation: The node specification is syntactical
incorrect.

What to do: Check the syntax of the node
specification and try again.

DES0976N   **The node information is incomplete.**

Explanation: The node specification is incomplete -
some information is missing.

What to do: Check the syntax of the node
specification and try again.

DES0977N   **The node information is incomplete.**
The left parenthesis is missing.

Explanation: The node specification is incomplete -
left parenthesis is missing.

What to do: Correct the node specification and try
again.

DES0998N   **The document model name length is**
incorrect.

Explanation: The length of the 'document model
name' value is incorrect.

What to do: Check the model name value and try
again.

DES0999N   **The syntax for the specification of the**
document model(s) is not correct.

Explanation: The specification of the model name(s) is
syntactical incorrect.

What to do: Check the syntax of the model(s)
specification and try again.

Messages from DB2 Text Extender

DES5250E Unable to create the Text Extender instance directory.

Explanation: The DB2 Text Extender directory /QIBM/userData/DB2Extenders/Text/instance cannot be created.

What to do: Check that you have the required authority to create the directory, and then try again.

DES5251I The Text extender instance already exists.

Explanation: The instance cannot be created as it already exists.

What to do: Use the existing instance, or drop and recreate the instance.

DES5252E Unable to drop the Text Extender instance.

Explanation: The instance directory /QIBM/UserData/DB2Extenders/Text/instance does not exist. The instance cannot be dropped.

What to do: Check if the instance has already been dropped by running TXSTATUS. If it has been dropped, you may wish to recreate the instance.

DES5253I The Text extender instance does not exist.

Explanation: You tried to drop an instance that does not exist.

DES9994N A Text Search Engine error occurred. Reason code: reason-code

Explanation: The text search engine used by DB2 Text Extender raised an error.

What to do: Check the search engine reason code in "Chapter 19. Search engine reason codes" on page 269. If the reported reason does not help to solve the problem additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

DES9995N A Text Extender error occurred. Message text: message-text

Explanation: A DB2 Text Extender error occurred.

What to do: Use the message provided by DB2 Text Extender to solve the problem. If the reported message does not help to solve the problem additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

DES9996N An internal DB2 Text Extender error occurred. Reason code: *reason_code*

Explanation: An internal processing error occurred.

What to do: Check that the DB2 Text Extender installation has been completed successfully. If yes, note the reason code and call your IBM service representative.

DES9997N An SQL error occurred. SqlState: *state* **QL Error code:** *rc*; **SqlErrorMessage:** *message*

Explanation: An SQL error occurred.

What to do: Take action on the SQL error message that is displayed with the message.

DES9998N An SQL error occurred. No further information is available.

DES9999N No corresponding error message.

Explanation: An internal processing error occurred.

What to do: Check the diagnostic message to solve the problem. If no installation problem causes the internal error additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

Chapter 19. Search engine reason codes

This chapter lists the reason codes that are returned by DB2 Text Extender's search engine.

Table 21. Search engine reason codes

Reason codes	Values
RC_DONE	0
RC_CONTINUATION_MODE_ENTERED	1
RC_END_OF_INFORMATION	2
RC_EMPTY_LIST	3
RC_MORE_INFORMATION	4
RC_INDEX_GROUP_SEARCH_ERROR	7
RC_INDEX_SPECIFIC_ERROR	8
RC_DICTIONARY_NOT_FOUND	9
RC_PROCESSING_LIMIT_EXCEEDED	12
RC_UNKNOWN_SERVER_NAME	16
RC_INCORRECT_AUTHENTICATION	17
RC_DATASTREAM_SYNTAX_ERROR	18
RC_QUERY_SCOPE_TOO_COMPLEX	20
RC_QUERY_TOO_COMPLEX	22
RC_MEMBER_OF_INDEX_GROUP	23
RC_UNKNOWN_INDEX_NAME	24
RC_INCORRECT_HANDLE	25
RC_INDEX_NOT_MEMBER_OF_GROUP	26
RC_UNKNOWN_SESSION_POINTER	27
RC_UNKNOWN_COMMUNICATION_TYPE	29
RC_UNKNOWN_SERVER_INFORMATION	30
RC_INVALID_MASKING_SYMBOL	31
RC_UNEXPECTED_ERROR	32
RC_SERVER_NOT_AVAILABLE	33
RC_INDEX_ALREADY_OPENED	35
RC_MAX_NUMBER_OF_OPEN_INDEXES	36
RC_MAX_NUMBER_OF_RESULTS	37
RC_CCS_NOT_SUPPORTED	41
RC_LANGUAGE_NOT_SUPPORTED	42
RC_CONFLICT_WITH_INDEX_TYPE	43
RC_MAX_INPUT_SIZE_EXCEEDED	46
RC_SERVER_BUSY	47
RC_SERVER_CONNECTION_LOST	48
RC_SERVER_IN_ERROR	49
RC_REQUEST_IN_PROGRESS	50
RC_UNKNOWN_INDEX_TYPE	51
RC_INCORRECT_INDEX_NAME	52
RC_INCORRECT_LS_EXECUTABLES	53
RC_INCORRECT_LIBRARY_ID	54
RC_INDEX_ALREADY_EXISTS	55
RC_MAX_NUMBER_OF_INDEXES	56
RC_INCORRECT_LOCATION	57
RC_LOCATION_IN_USE	58
RC_UNKNOWN_CONDITION	59

Search engine reason codes

Table 21. Search engine reason codes (continued)

Reason codes	Values
RC_INDEX_DELETED	60
RC_INDEX_SUSPENDED	61
RC_INDEX_NOT_ACCESSIBLE	62
RC_MAX_NUMBER_OF_BUSY_INDEXES	63
RC_CONFLICTING_TASK_RUNNING	64
RC_NOT_ENOUGH_MEMORY	65
RC_MAX_OUTPUT_SIZE_EXCEEDED	68
RC_COMMUNICATION_PROBLEM	70
RC_NO_ACTION_TAKEN	71
RC_EMPTY_INDEX	72
RC_EMPTY_QUERY	73
RC_INSTALLATION_PROBLEM	74
RC_FUNCTION_DISABLED	75
RC_FUNCTION_IN_ERROR	76
RC_IO_PROBLEM	77
RC_WRITE_TO_DISK_ERROR	78
RC_SERVER_VERSION_NOT_CURRENT	79
RC_FUNCTION_NOT_SUPPORTED	80
RC_RESULT_ALREADY_RANKED	81
RC_RESULT_VIEW_EXISTS	82
RC_INDEX_NOT_OPEN	83
RC_NO_RANKING_DATA_AVAILABLE	84
RC_LINGUISTIC_SERVICE_FAILED	85
RC_THESAURUS_PROBLEM	86
RC_INVALID_IDENTIFIER	88
RC_DOCUMENT_MODEL_ALREADY_EXISTS	89
RC_UNKNOWN_DOCUMENT_SECTION_NAME	90
RC_DOCMOD_READ_PROBLEM	91
RC_UNKNOWN_DOCUMENT_MODEL_NAME	92
RC_SECTION_NAME_ALREADY_EXISTS	94
RC_SECTION_TAG_ALREADY_EXISTS	95
RC_MAX_NUMBER_OF_TASKS	96
RC_LS_NOT_EXECUTABLE	97
RC_LS_FUNCTION_FAILED	98
RC_CAPACITY_LIMIT_EXCEEDED	99
RC_DOCUMENT_NOT_ACCESSIBLE	100
RC_DOCUMENT_CURR_NOT_ACCESSIBLE	101
RC_DOCUMENT_NOT_TO_INDEX	102
RC_DOCUMENT_NOT_FOUND	103
RC_DOCUMENT_IN_ERROR	104
RC_DOCUMENT_NOT_SUPPORTED	105
RC_CROSSIDX_SEARCH_NOT_ALLOWED	110
RC_DOCUMENT_GROUP_NOT_FOUND	111
RC_INVALID_ATTRIBUTE_VALUE	112
RC_INVALID_SECTION_TYPE	113
RC_INCORRECT_RELEVANCE_VALUE	120
RC_NO_RAT_EXPANSION	130
RC_DOCUMENT_NOT_IN_VIEW	131

Chapter 20. Error event reason codes

This chapter lists the error events that can occur when DB2 Text Extender indexes documents. This can occur, for example, when:

- Documents cannot be indexed
- Documents are indexed, but a problem occurs
- A language dictionary cannot be found.

Tip

If a reason code is not documented:

1. Check that there is enough disk space.
2. Collect all the error information that is available:
   - desdiag.log file
   - Event message
3. Call your IBM service representative.

---

1 Out of storage. The server ran out of memory. Reduce the workload.

64 The index process is still running, or the index is still reorganizing. This reason code is for information only.

116

Datastream syntax error

280

The document has not been indexed. One of the index files could not be opened.

281

The document has not been indexed. One of the index files could not be read.

441

The document has not been indexed. This message occurs for Ngram indexes only. The document's code page is different from the one the index was created with. This may happen for HTML and XML documents if the index was not created in UTF8.

500

The document has not been indexed. The Library Services could not be loaded. Check that the DLL is available and that the resource path is valid.

501

The document has not been indexed. Lib_Init in Library Services failed On Flat File systems: DIT file not found or not on a valid directory, or DIT contents not correct.

502

The document has not been indexed. An error has occurred while reading the document content in library service LIB_read_doc_content.

503

The document has not been indexed. An error occurred in library service LIB_access_doc.

Error event reason codes

504

The document has not been indexed. The library service LIB_doc_index_status returned an error.

505

Close document failed. The library service LIB_close_doc returned an error.

506

End Library Services failed The library service LIB_end returned an error.

507

The library service call LIB_read_doc_content failed with an unexpected return code.

508

The library service call LIB_close_doc returned a RC_TERMINATION error.

545

The document has not been indexed. One of the temporary index files could not be opened.

546

The document has not been indexed. One of the temporary index files could not be closed.

548

Internal error. Send the information in the diagnosis log to your IBM representative.

549, 550

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

551-564

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

565

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

566-587

The document has not been indexed. One of the index files could not be opened, read, written to or closed.

588-590

Internal error. Send the information in the diagnosis log to your IBM representative.

591-604

The document has not been indexed. One of the index files could not be opened, read, written to or closed.

605

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

606-623

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

624

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

625-631

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

632

One of the temporary files created during indexing could not be opened with write access. Check the access rights.

633

One of the temporary files created during indexing could not be closed.

634

One of the temporary files created during indexing could not be written. Check that the index working directory has enough disk space.

635

One of the temporary files created during indexing could not be read.

636

One of the temporary files created during indexing could not be opened with read access. Check the access rights.

659

One of the temporary files created during indexing could not be opened.

660

One of the temporary files created during indexing could not be written.

661

One of the temporary files created during indexing could not be closed.

662

One of the temporary files created during indexing could not be opened.

663

One of the temporary files created during indexing could not be written.

664

One of the temporary files created during indexing could not be closed.

665

One of the temporary files created during indexing could not be opened.

667

One of the temporary files created during indexing could not be written.

668-669

The document was not indexed. There was a matching problem with section tags encountered in the document versus those defined in document models file.

670-672

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

673

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

Error event reason codes

674

Internal error, send the information in the diagnosis log to your IBM representative.

675-687

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

688, 690

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in the configuration file.

689

Internal error. Send the information in the diagnosis log to your IBM representative.

691-695

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

696-707

Internal error, send the information in the diagnosis log to your IBM representative.

708

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

709-718

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

719-721

Internal error, send the information in the diagnosis log to your IBM representative.

722, 729

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

730, 732, 733, 735-738

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk for the index and that access rights are correct.

731, 739-742, 744-746, 749, 755-758, 760-761, 767

Internal error, send the information in the diagnosis log to your IBM representative.

743, 748, 750-754, 759, 765-766, 768-770

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

747, 763, 764

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

815

There are two possible causes:

- One of the resource files needed to support the language used for the document causing the failure was not found
- The language requested by that document is not supported by DB2 Text Extender

831

The document has not been indexed. No text has been found. The document length is 0 bytes.

860

File open error. Either some dictionaries or the thesaurus files could not be found. Check your resource path for the dictionary files. If you have specified path information for your thesaurus files during search, check the location and file name.

861

The document has not been indexed. Tokenization of the text found no valid tokens. Check the documentÆs content for validity, with respect to supported languages and contained words. This error can be caused by a document containing only stopwords.

954-956

Internal error. Send the information in the diagnosis log to your IBM representative.

957-967

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

1000

An error occurred during file open. Check access rights.

1001

An error occurred during file append. Check access rights.

1002

An error occurred during file read. Maybe the file is corrupted.

1003

An error occurred during file write. Check disk space and access rights.

1005

An error occurred during file read (positioning within file). Maybe the file is corrupted.

1006

An error occurred during rename of temporary file. Check the access rights.

1007

An error occurred during file create. Check access rights.

1008

An error occurred during file compression. Check the access rights.

1009

An error occurred during file close. Maybe the file is corrupted.

1010

The specified index name is already in use. Use another index name.

1011

The specified path is already in use. Use another location.

Error event reason codes

1012

The same path is used for data and working directory. Use another location.

1013

The specified index name is invalid. Index names must be uppercase or digits and not longer than 8 characters.

1014

An error occurred during file copy. Check access rights and disk space.

1017

The index name is unknown. Check correct spelling.

1019

An error occurred during file deletion. Check access rights. This error message can occur as a "secondary error" - look up the diagnosis file to see if a preceding error entry gives more information.

1020

General file error. Check access rights.

1070-1074

The document has not been indexed. The code page specified is either invalid generally, or is invalid for the index being accessed.

1085

The document has not been indexed. An error occurred when reading the index queue.

1086

The document has not been indexed. The index queue is empty.

1116-1117

The document has not been indexed. Information from the server instance initialization file could not be processed. Make sure entries in the initialization file are valid, and that the file is accessible to the application.

1129

No document has been indexed. Starting the background processing failed.

1158

An error occurred while renaming a file. Check access rights and disk space.

1162

The index files of an Ngram index may be corrupted.

1163, 1164

The document has not been indexed due to an unexpected error.

1165

The document has not been indexed due to an unexpected end-of-file condition.

1176

No more documents can be indexed for Ngram index. There is an overflow condition for document numbers (overflow of long). If there were many deletions or repeated updates of the same document, try a call to EhwReorg to solve the problem. If not, consider using a second index for new documents.

1177

The document has not been indexed. It was considered too big by the Ngram indexer.

1189

The document has not been indexed. There was a problem with boundary sequence (Korean-language specific).

1198-1200

The document has not been indexed. There was a problem with index access. The index may be corrupted.

1201

The document has not been indexed. The document code page could not be converted to the index-specific code page. This error is for Ngram indexes in UTF8 code page only.

1202

The document has not been indexed. The document code page could not be converted to the index-specific code page due to invalid data in the document. This error is for Ngram indexes in UTF8 code page only.

1500-1505

The document-analysis component has problems. It could either not be initialized (check LIBPATH and content of configuration file) or failed due to internal problems. See the diagnosis file for more information.

1904

The document has not been indexed. There is a problem with accessing the document model for a section-enabled index. Check access rights and for the existence of the file.

2000

The document has not been indexed. The document type is not supported. Library service Lib_access_doc returned an invalid document type.

2001

The document has not been indexed. An incorrect sequence of fields has been detected in the document's data stream.

2002

The document has not been indexed. An incorrectly structured field has been detected in the document's data stream.

2003

The document has not been indexed. Only one text section is allowed for a document in DB2 Text Extender text format.

2005

The document has not been indexed. A language specified in the document's data stream is not supported.

2006

The document has not been indexed. A CCSID specified in the document's data stream is not supported.

2007

The expected document format given by the library or by the default rule is not correct. The document header is incorrect for the format. Check if the default rule is a document with a special document header, and change if the rule is not correct.

2008

The document was not indexed because it could not be accessed.

Error event reason codes

2009

The document was not indexed because it was in use and could not be accessed.

2010

The document has not been indexed. The specified CCSID is not correct.

2011

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. End-of-page must be the last control in the body text of the document.

2012

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. A structured field contains an incorrect length specification.

2013

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An incorrect control has been detected in the document.

2014

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An incorrect multi-byte control or structured field has been detected in the document.

2015

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. Duplicate document parameters have been found.

2016

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An empty text unit has been found.

2018

Either the document is in a format that is not supported, or there is an "exclude" entry in the DIT for the document's extension. Check that the document has a extension that allows it to be indexed.

2020

The document has not been indexed. It is neither a WordPerfect document nor a WordPerfect file.

2021

The document has not been indexed. It is a WordPerfect file but not a WordPerfect document.

2022

The document has not been indexed. This version of WordPerfect is not supported.

2023

The document has not been indexed. It is an encrypted WordPerfect file. Store the document without encryption.

2026

An END_TXT occurred in a footnote or an endnote. Check the WordPerfect file, it may be damaged.

2028

The parser returned non-document text. Check the file content, especially with respect to format-specific words. Check whether the document format is supported. If automatic format recognition fails, ensure that the correct parser is called.

2030

The document has not been indexed. Either it is not a Microsoft Word file or it is a version of Word that is not supported.

2031

The document has not been indexed. Unexpected end-of-file has been detected in a Microsoft Word document.

2032

The document has not been indexed. An incorrect control has been detected in a Microsoft Word document.

2033

The document has not been indexed. It was saved in *complex* format with the *fastsave* option. Save it with the *fastsave* option off.

2034

The document has not been indexed. A required field-end mark is missing in a Microsoft Word document.

2035

The document is encrypted. Store the document in Microsoft Word without encryption.

2036

This is a Word for Macintosh document; it cannot be processed. Store the document in Word for Windows format.

2037

This Word document contains embedded OLE objects.

2040

The document has not been indexed because it is not a valid ECTF file.

2041

The document has not been indexed. It contains an .SO LEN control that is not followed by a number.

2042

The document has not been indexed. It contains an .SO LEN control that is followed by an incorrect number. The number must be between 1 and 79.

2043

The document has not been indexed. Only one .SO DOC control is allowed. Save each ECTF document in a separate file.

2044

The document has not been indexed. An .SO HDE control must be followed by begin and end tags.

2046

The document has not been indexed. The document contains text before the .SO DOC control.

2047

The document has not been indexed. The document contains text before an .SO PID control.

2048

The document has not been indexed. An end tag is missing after a begin tag.

2050

The document has not been indexed. Incorrect tags have been detected following an .SO HDE control.

Error event reason codes

2051

The document has not been indexed. End-of-line has been detected after an .SO control.

2052

The document has not been indexed. Unexpected end-of-text has been detected.

2060

The document has not been indexed. Either it is not an AmiPro document or it is a version of AmiPro that is not supported.

2061

The document has not been indexed. The length of a control in an AmiPro document is too long.

2062

The document has not been indexed. This version of AmiPro is not supported. Only AmiPro Architecture Version 4 is supported.

2063

AmiPro Style Sheets have not been indexed.

2064

The document has not been indexed. An incorrect character set has been detected. Only Lotus Character Set 82 (Windows ANSI) is supported.

2065

The document has not been indexed. Unexpected end-of-file has been detected in an AmiPro document.

2072

The document cannot be scanned because it is encrypted.

2073

The document format is inconsistent.

2074

The document has the "bad file" flag bit set.

2080

The document has not been indexed. Either it is not an RTF document or it is a version of RTF that is not supported.

2081

The document has not been indexed. An RTF control word has been detected that is too long.

2083

The document has not been indexed. Macintosh code page is not supported.

2084

The document has not been indexed. It is an RTF document, but this RTF version is not supported. Only RTF Version 1 is supported.

2090

The document has not been indexed. It is an HTML document, which contains a tag considered too long by the parser.

2093

The document has not been indexed. It is an XML document, which was rejected by the XML parser.

2100

The document is damaged or unreadable for some other reason. A new common parser could correct the problem.

2101

The document cannot be indexed because it is empty or it contains no text. Check whether the document contains only graphics.

2102

The document cannot be indexed because it is either password-protected or encrypted.

2105

The document type is known, but the filter is not available.

2106

The document cannot be indexed because it is empty.

2107

The document cannot be indexed because it cannot be opened. Check document access.

2112

The document cannot be indexed because it is an executable file.

2113

The document cannot be indexed because it is compressed.

2114

The document cannot be indexed because it is a graphic. If the graphic document format returns an acceptable piece of text, then request to include this document format in the indexing process.

2120

The output file of the user exit does not exist or is not accessible. A new common parser version could correct the problem.

2121

The output file cannot be opened for read or it is empty. A new common parser version could correct the problem.

2122

Attempting to use a user-exit output file, but no file name has been given or set in the object.

2130

The user exit program could not be run. Check if the executable can be found in the path set by the PATH environment variable. Create a trace and dump to get additional information about the environment (errno) return codes.

2131

The user exit program failed with a bad return code. Create a trace and dump to get additional information about the environment (errno) return codes.

Part 3. Appendixes

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 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:

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM	IBM
AISPO	IMS/ESA
AIX	LAN Distance
AIXwindows	MVS
AnyNet	MVS/ESA
APPN	MVS/XA
AS/400	Net.Data
BookManager	OS/2
CICS	OS/390
C Set++	OS/400
C/370	PowerPC
DATABASE 2	QBIC
DataHub	QMF
DataJoiner	RACF
DataPropagator	RISC System/6000
DataRefresher	RS/6000
DB2	S/370
DB2 Connect	SP
DB2 Extenders	SQL/400
DB2 OLAP Server	System/370
DB2 Universal Database	System/390
Distributed Relational	SystemView
Database Architecture	VisualAge
DRDA	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
	WIN-OS/2

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.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Glossary

This glossary defines many of the 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 Text Extender.

API. Application programming interface.

application programming interface (API). A general-purpose interface between application programs and the DB2 Text Extender information retrieval services.

B

Boolean search. A search in which one or more search terms are combined using Boolean operators.

bound search. A search in Korean documents that respects word boundaries.

browse. To view text displayed on a computer monitor.

browser. A DB2 Text Extender function that enables you to display text on a computer monitor.

C

catalog view. A view of a system table created by DB2 Text Extender for administration purposes. A catalog view contains information about the tables and columns that have been enabled for use by DB2 Text Extender.

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 DB2TX that:

> Allows you to enter DB2 Text Extender commands
> Processes the commands
> Displays the result.

common-index table. A DB2 table whose text columns share a common text index. See also *multi-index table*.

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

data stream. Information returned by an API function, comprising text (at least one paragraph) containing the term searched for, and information for highlighting the found term in that text.

DB2 Extender. One of a group of programs that let you store and retrieve data types beyond the traditional numeric and character data, such as image, audio, and video data, and complex documents.

DBCS. Double-byte character support.

dictionary. A collection of language-related linguistic information that DB2 Text Extender uses during text analysis, indexing, retrieval, and highlighting of documents in a particular language.

disable. To restore a database, a text table, or a text column, to its condition before it was enabled for DB2 Text Extender by removing the items created during the enabling process.

distinct type. See *user-defined distinct type*.

document. See *text document*.

document handle. See *handle*.

document model. The definition of the structure of a document in terms of the sections that it contains. A document model makes DB2 Text 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, a text table, or a text column, for use by DB2 Text Extender.

environment variable. A variable used to provide defaults for values for the DB2 Text Extender environment.

environment profile. A script provided with DB2 Text Extender containing settings for *environment variables*.

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.

extended matching. A process involving the use of a *dictionary* to highlight terms that are not obvious matches of the search term.

extender. See *DB2 Extender*.

external file. A text document in the form of a file stored in the operating system's file system, rather than in the form of a cell in a table under the control of DB2.

F

feature search. A search for terms such as names of people, places, or organizations, made in a linguistic index created using the FEATURE_EXTRACTION indexing option.

file handle. See *handle*.

format. The type of a document, such as ASCII, or WordPerfect.

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

handle. A binary value that identifies a text document. It includes:

   A document ID

   The name and location of the associated index

   The document's *text information*

   If the document is located in an external file not under the control of DB2, the path and name of the file.

A handle is created for each text document in a text column when that column is *enabled* for use by DB2 Text Extender.

highlighting information. See *data stream*.

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 directory where the index is stored

   The index type

   The frequency with which the index is updated

   When the first index update is to occur.

index type. A characteristic of a *text index* determining whether it contains exact or linguistic forms of document terms. See *precise index, linguistic index,* and *Ngram index*.

initialized handle. A *handle*, prepared in advance, containing only the text format, or the text language, or both.

instance. A logical DB2 Text Extender environment. You can have several instances of DB2 Text Extender on the same workstation, but only one instance for each DB2 instance. You can use these instances to:

   Separate the development environment from the production environment

   Restrict sensitive information to a particular group of people.

instance variable. A variable used to provide a default value for the name of the *instance* owner, or the name of the instance owner's home directory.

L

language. The name of a *dictionary* to be used when *indexing*, searching and *browsing*.

linguistic index. A *text index* containing terms that have been reduced to their base form by linguistic processing. "Mice", for example, would be indexed as "mouse". See also *precise index* and *Ngram index.*

logical node. A *node* assigned with other nodes to the same physical machine. See also *physical node*.

log table. A table created by DB2 Text 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.

multi-index table. A DB2 table whose text columns have individual *text indexes*. See also *common-index table*.

N

Ngram index. A *text index* that supports DBCS documents and fuzzy search of SBCS documents. See also *linguistic index* and *precise index.*

node. A server in a *partitioned database* environment. See also *logical node*, *physical node*, and *nodegroup*.

nodegroup. A named subset of one or more database partition servers. *node* assigned to a physically separate machine. See also *logical node*.

O

occurrence. Synonym for *match*.

P

partitioned database. A database consisting of several parts, each of which is maintained by a separate database partition server.

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.

physical node. A *node* assigned to a physically separate machine. See also *logical node*.

precise index. A *text index* containing terms exactly as they occur in the text document from which they were extracted. See also *linguistic index* and *Ngram index.*

profile. See *environment profile*.

R

rank. 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.

refine. To add the search criteria from a previous search to other search criteria to reduce the number of *matches*.

retrieve. To find a text document using a search argument in one of DB2 Text Extender's search functions.

S

SBCS. Single-byte character support.

search argument. The conditions specified when making a search, consisting of one or several search terms, and search parameters.

shell profile. See *environment profile*.

stop word. A common word, such as "before", in a *text document* that is to be excluded from the *text index*, and ignored if included in a *search argument*.

T

text column. A column containing *text documents*.

text configuration. Default settings for index, text, and processing values.

text document. Text of type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB, 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. See also *precise index* and *linguistic index.*

text information. Properties of a *text document* describing:
> The *CCSID*
> The *format*
> The *language*.

text table. A DB2 table containing *text columns*.

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 distinct 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 distinct 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. DB2 Text Extender provides administration and search functions, such as CONTAINS, in the form of UDFs.

W

wildcard character. See *masking character*.

Index

Special Characters

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 U.S.A., 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 U.S.A., then you can call one of the following numbers:
* 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
* 1-800-879-2755 to order publications.

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

http://www.ibm.com/software/data/db2/library/
The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

Note: This information may be in English only.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
The International Publications ordering Web site provides information on how to order books.

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

ftp.software.ibm.com
Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools relating to DB2 and many other products.

comp.databases.ibm-db2, bit.listserv.db2-l
These Internet newsgroups are available for users to discuss their experiences with DB2 products.

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://www.ibm.com/support/, and then select the IBM Software Support Handbook link near the bottom of the page.

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

IBM ®

Printed in U.S.A.

Spine information:

IBM® DB2® Universal Database   DB2 Text Extender Administration and Programming   Version 7.1.0.1

IBM