DB2 Universal Database for OS/390 and z/OS

IBM

Version 7

Net Search Extender Administration and Programming

SC27-1171-01

DB2® Universal Database for OS/390 and z/OS

IBM

Version 7

Net Search Extender Administration and Programming

SC27-1171-01

Note!
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 75.

Second Edition (March 2004)

This edition applies to DB2 Net Search Extender, a priced feature of DB2 UDB Server for OS/390, V6 (5645-DB2) and DB2 UDB Server for OS/390 and z/OS, V7 (5675–DB2). This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions.

This and other books in the DB2 for OS/390 and z/OS library are periodically updated with technical changes. These updates are made available to licensees of the product on CD-ROM and on the Web (currently at `www.ibm.com/software/data/db2/os390/library.html`). Check these resources to ensure that you are using the most current information.

Contents

About this book

DB2 Net Search Extender contains a DB2® stored procedure that adds the power of fast full-text retrieval to Net.Data®, Java®, or DB2 CLI applications. It offers application programmers a variety of search functions, such as fuzzy search, stemming, Boolean operators, and section search.

Searching using Net Search Extender can be particularly advantageous in the Internet when performance is an important factor.

Who should use this book

This book is intended for:

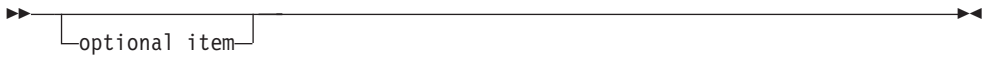
- Administrators who prepare databases for text search by creating Net Search Extender indexes, and who update and maintain these indexes.
- Database application programmers who implement text search for end users.

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.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.



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

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

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

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

- Keywords appear in uppercase; they must be spelled exactly as shown. Variables appear in lowercase (for example, `srcpath`). They represent user-supplied names or values in the syntax.

  In all administration commands, if the keyword DATABASE is used, infer your subsystem. If the database name is requested, use the location name of your subsystem.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products enable users to:

- Use assistive technologies such as screen reader and screen magnifier software
- Operate specific or equivalent features by using only a keyboard
- Customize display attributes such as color, contrast, and font size

Assistive technology products, such as screen readers, function with the z/OS user interfaces. Consult the documentation for the assistive technology products for specific information when you use assistive technology to access these interfaces.

Online documentation is available in the DB2 Information Center, which is an accessible format when used with assistive technologies such as screen reader or screen magnifier software. The DB2 Information Center for z/OS solutions is available at the following Web site:
`http://publib.boulder.ibm.com/infocenter/db2zhelp`.

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 order 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.

Part 1. Guide

Chapter 1. Introduction

DB2 Net Search Extender adds the power of a fast full-text retrieval engine to Net.Data, Java, or DB2 CLI applications integrated into DB2 Universal Database™.

Optimized for performance, the benefits of using Net Search Extender are:

- Fast indexing of very large data volumes
- Searching at high speed with a large number of concurrent users
- Storing presorted table columns in main memory at indexing time to avoid expensive database access

Features

The key features of Net Search Extender include:

- Indexing
  - A case-insensitive index type
  - Multiple different indexes on the same text column or view are possible
  - Indexing proceeds without locking data
  - Incremental updating of indexes to reflect changes in the database
  - Identification of numeric columns to restrict search
- Search
  - Provides a stored procedure on the server
  - Allows word, phrase, stemmed, or fuzzy search
  - Identifies and restricts searching to sections within documents that have been marked by special tags
  - Offers numeric search on ranges of values
  - Supports Boolean and wildcard operations
- Search results
  - Lets you specify how the search results are sorted at indexing time, or uses relevance rank values for sorting
  - Lets you specify search result subsets when large data volumes are searched and large result lists are expected
  - Lets you set a result limit on search terms with a high hit count
  - Allows positioning (cursor-setting) access on search results

These features enable Net Search Extender to provide solutions for the Internet, intranet, and other DB2–based applications.

Note

Net Search Extender for OS/390 and z/OS does not support the DB2 UDB for OS/390 and z/OS data sharing function in a Parallel Sysplex environment.

Chapter 2. Customization

This chapter describes how to customize DB2 Net Search Extender.

System requirements

Net Search Extender runs on one of the following operating systems:

- OS/390® V2.10 or higher, and z/OS® V1.0 or higher

If you want to use Net.Data to access the stored procedure of Net Search Extender, a minimum of Net.Data V6.1 needs to be installed.

If you want to use Java to access the stored procedure, a minimum of JDK 1.1.6 needs to be installed.

The Workload Manager must also be installed.

There is a minimum memory requirement of 256 MB RAM.

The overall memory requirement depends on the number of documents to be indexed, and on the performance optimization of your indexes. See point “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

OS/390 customization

For OS/390, DB2 Net Search Extender requires a Workload Manager (WLM) environment as described in the DB2 Program Directory. The DB2 Net Search Extender instance owner must have a super user ID, DB2 SYSADM authority and a home directory in the UNIX® system services on OS/390.

After you have completed the SMP/E installation, follows the steps described here to set up the Net Search Extender so that it can run with DB2.

Customizing

Net Search Extender is customized in MVS™ datasets and files within the Hierarchical File System (HFS). During SMP/E installation, you can specify target datasets and HFS directories. By default, the following HFS directories are used:

- /usr/lpp/db2nx
- /usr/lpp/db2nx/IBM
- /usr/lpp/db2nx/bin
- /usr/lpp/db2nx/icu/lib
- /usr/lpp/db2nx/icu/data
- /usr/lpp/db2nx/nls
- /usr/lpp/db2nx/samples
- /usr/lpp/db2nx/instance

If you want to specify a different set of directories during installation, you can prefix these directories with additional names, like /u/myname, however, you must not omit the default part of the directory names. The fully qualified name becomes, for example, /u/myname/usr/lpp/db2nx/IBM.

Customization

You must customize DB2 Net Search Extender in the UNIX Shell and HFS directories because it only provides its db2nx command-line interface via the UNIX shell.

Note that APF authorization is required for both the HFS libraries libdne-i18u.dll and libdne-ue.dll.

There are two ways to access the DB2 Net Search Extender load library hlq.SDNEMOD1 from the UNIX Shell of OS/390 environment:

- By adding an entry hlq.SDNEMOD1 to your Linklist, that is, the PROGxx or the LNKLSTxx of your PARMLIB
- By adding hlq.SDNEMOD1 to the STEPLIB environment variable

The Linklist approach is recommended for performance reasons, and to simplify the customization steps. If you cannot add hlq.SDNEMOD1 to the Linklist, then you must use STEPLIB. If you work with the STEPLIB environment variable ensure all users of DB2 Net Search Extender have at least one of the following:

- RACF® READ access for the OS/390 load library hlq.SDNEMOD1 which contains all load modules
- RACF EXEC access if there is a PROGRAM profile for the load library members

Creating an instance

To create a Net Search Extender instance, define and create a directory under the instance owner home directory where all instance files can be copied or linked to with a symbolic link, for example:

```
mkdir /u/instanceownerusername/db2nx
```

Copy or create symbolic links to the following installation directories in the instance owners home directory:

```
ln -s /usr/lpp/db2nx/bin /u/instanceownerusername/db2nx/bin

cp /usr/lpp/db2nx/samples/db2nxprofile /u/instanceownerusername/db2nx

ln -s /usr/lpp/db2nx/icu /u/instanceownerusername/db2nx/icu

ln -s /usr/lpp/db2nx/msg /u/instanceownerusername/db2nx/msg

cp -r /usr/lpp/db2nx/samples /u/instanceownerusername/db2nx/samples
```

You now need to customize the db2nxprofile which contains settings of important environment variables to define your Net Search Extender instance.

DB2NX_INSTOWNERHOMEDIR
: The directory in which the Net Search Extender instance has been created. This is usually the home directory of the instance owner, for example, `/u/instanceownerusername`

DB2NXINSTALLDIR
: The instance installation directory, for example, `/u/instanceownerusername/db2nx`

DB2NX_DB2CCSID
: Set to the CCSID of the DB2 subsystem. This CCSID is specified during the

DSNHDECP installation job (known as the DSNTIJUZ job). If the DB2 Performance Monitor is installed, check the CCSID of the subsystem by:

- Selecting your subsystem using Spufi
- Selecting the DB2 Performance Monitor
- Selecting ″View online activity″ option 4
- Selecting ″Display system parameter″ option 5
- Selecting ″Application programming defaults″

Note that the CCSID of the table space that contains the tables that you want to index also needs to be set to the same value.

ICU_DATA
The directory where code page conversion data is stored, for example, `/u/instanceownerusername/db2nx/icu/data`

DSNAOINI
The definition file for DB2 SQL CLI interface. In this file the DB2 subsystem to be used will be defined. Customize the contents of the DSNAOINI file according to your needs. Note that the parameter MULTICONTEXT=1 can cause administration commands to fail with error DES7606N. On the other hand, MULTICONTEXT=1 is required to optimize search performance in some multithreaded search scenarios. In this case, two separate DSNAOINI files for administration and search are recommended. See DB2 ODBC documentation for more details. Net Search Extender provides an example in hlq.SDNESAMP(DNEDSNAO).

PATH Update path to where the executable for the administration part is stored, for example, `${PATH}:/u/instanceownerusername/db2nx/bin`

LIBPATH
Update the library path to where the code page conversion functions are stored, for example, `${LIBPATH}:/u/instanceownerusername/db2nx/icu/lib`

STEPLIB
Update steplib to where the dataset containing Net Search Extender and DB2 ODBC load libraries is stored, for example, STEPLIB=$STEPLIB:hlq.SDNEMOD1:DSN.SDSNEXIT:DSN.SDSNLOAD. You do not need to do this if you are using the LNKLIST approach to access your datasets.

LANG Set lang to specify what language messages are to appear in. Net Search Extender searches $DB2NX_INSTOWNERHOMEDIR/msg/$LANG (for example, `/u/instanceownerusername/db2nx/msg/En_US`) for a message file. If it cannot find one there, it searches $DB2NXINSTALLDIR/msg/En_US (for example, `/u/instanceownerusername/db2nx/msg/En_US`) for the US English message file.

You will find it useful to code the following call from your .profile to the db2nxprofile so that you always have the correct runtime options set for DB2 Net Search Extender:

```
#Set DB2 Net Search Extender environment variables
if [-f ./db2nx/db2nxprofile ];
then ../db2nx/db2nxprofile
fi
```

Workload Manager related tasks

For the next Workload Manager (WLM) related tasks, you need a user ID with UPDATE access to the proclib dataset to edit the customization information of the started task of your WLM environment.

Configuring the WLM-started task

To configuring the WLM-started task, take the skeleton for the WLM-started task provided in hlq.SDNESAMP(DNEWLM) and copy this job to the proclib defined in your WLM environment. Ensure that you have at least WLM read access to the following load libraries:

- hlq.SDNEMOD1
- hlq.SDSNEXIT
- hlq.SDSNLOAD

Errors caused by insufficient access rights can only be found in the system log after starting the WLM. For WLM to find these libraries, they must be defined in one of the following:

- The Linklist
- The STEPLIB section of the WLM job

In the skeleton, you will find the ddname DNEENVAR. The dataset specified by this ddname contains the valid environment variables for executing the DB2 Net Search Extender stored procedure. The sample job hlq.SDNESAMP(DNECSP) which creates the stored procedure shows this ddname in the runtime options of the CREATE STORED PROCEDURE statement. A sample is provided in hlq.SDNEDATA(DNEENVAR). The environment variable definitions are the same as described above in “Creating an instance” on page 6. Review and change them according to your needs.

Activating the WLM application environment

To activate the load library hlq.SDNEMOD1, and to make your changes to the configuration of the started task effective, you must refresh the application environment in use for the specified DB2 subsystem.

1. Enter the following OS/390 system command:

   ```
   /v wlm,applenv=<WLM AppEnv Name>,refresh
   ```

   The WLM AppEnv Name is the name of the WLM environment that you use for running the DB2 Net Search Extender stored procedure. It must have been defined previously in the WLM environment and have been defined by your DB2 administrator for use with a specific subsystem.
2. Check the correct startup of this application environment in the system log.

DB2 related tasks

DB2 Net Search Extender exploits some of the functionality of DB2 using four stored procedures to provide search functionality for:

- Standard search (no ranking)
- Standard search with tracing (no ranking)
- Standard search with ranking
- Standard search with ranking and tracing

These stored procedures can then be used within Net.Data, Java , or DB2 CLI applications integrated into DB2.

The signatures for the stored procedures are shown below.

Creating the stored procedure

On OS/390, all the stored procedures mentioned above can be created by a sample job provided in hlq.SDNESAMP(DNECSP). Modify the job according to your needs and system settings. Note that the runtime options contain a DNEENVAR ddname for the ENVFILE settings.

It is important to adapt the contents of the dataset referenced by DNEENVAR according to your installation. See “Configuring the WLM-started task” on page 8 and “Creating an instance” on page 6 for more on setting the environment variables. To start the stored procedures, enter the following DB2 command in the relevant subsystem:

```
-sta proc(DB2NX.TEXTSEARCH)       - for a standard search (no ranking)
-sta proc(DB2NX.TEXTSEARCH_T)     - for a standard search with tracing (no ranking)
-sta proc(DB2NX.TEXTSEARCH_R)     - for a standard search with ranking
-sta proc(DB2NX.TEXTSEARCH_RT)    - for a standard search with ranking and tracing
```

Binding the Net Search Extender stored procedure package

On OS/390, DB2 Net Search Extender uses two methods to access DB2. The administration part uses the SQL CLI (DB2 ODBC), whilst the query part uses embedded SQL. For this reason, you need to use a package to bind Net Search Extender to the DB2 subsystem.

A package file hlq.SDNEDBRM(DNESRCH) and a sample job, hlq.SDNESAMP(DNEBIND) are provided for this purpose. Customize the job according to your system and install the package.

OMVS related tasks

For each text index a certain amount of shared memory and a certain number of shared memory semaphores are required. Depending on the number and the size of text indexes than you want to create, it may be necessary to increase the following OMVS parameters:

- MAXMMAPAREA

  For more detail, refer to page 56.
- MAXSHAREPAGES
- IPCSHMMPAGES
- IPCSHMNIDS
- IPCSHMNSEGS
- IPCSHMSPAGES
- MAXASSIZE
- TSO Region Size

Refer to “Calculating the amount of resources needed” on page 24 to estimate the resource requirements.

Chapter 3. Running the sample programs

A set of sample programs is provided with Net Search Extender. These programs are located in the db2nx/samples directory. The sample programs are a means by which you can verify your installation.

The sample programs can only be run after Net Search Extender has been installed and the relevant DB2 and Net Search Extender instances have been created .

The sample provides a database setup program and three search examples, two that run in Java and one that runs in Net.Data.

The sample job hlq.SDNESAMP(DNELSAMP), creates and populates the sample database and the nxsampleenable shellscript enables the data for search.

You can then use the Net.Data and Java samples to run searches against that data.

These are the sample programs:

hlq.SDNESAMP(DNELSAMP) and nxsampleenable
: The job hlq.SDNESAMP(DNELSAMP) creates and populates a sample database. The shell script nxsampleenable enables the data in three sample indexes on OS/390.

  You must run this program before running the Net.Data sample nxsample.d2w and the two Java samples NXSample.java and NXSampleRank.java.

nxsample.d2w
: A Net.Data macro that displays a Web page that allows searching on the indexes created by the sample database setup program nxsample.

NXSample.java
: A Java program that allows searching without ranking on the search results from indexes created by the sample database setup program nxsample.

NXSampleRank.java
: A Java program that allows searching with ranking on the search results from indexes created by the sample database setup program nxsample.

To prepare your own database and run searches against your own data, see Chapter 4, “Creating and maintaining Net Search Extender indexes,” on page 17.

Running the sample database setup program

To run the sample database setup program:

On OS/390, run the job from the TSO environment.

If you don’t want to use an existing database, create a new one.

A connection is always made to the subsystem specified in the dataset, or file referenced by the DSNAOINI environment variable.

The hlq.SDNESAMP(DNELSAMP)job and nxsampleenable shellscript perform the following tasks:

- Creates a table db2nx.sample in the specified database

- Puts sample data into that table
- Runs `db2nx enable database` to enable the database for use by Net Search Extender
- Creates three sample indexes to be searched by the Net.Data sample macro `nxsample.d2w` or the Java sample programs `NXSample.java` and `NXSampleRank.java`

Monitoring the Net Search Extender environment

After running `hlq.SDNESAMP(DNELSAMP)`, two commands can be used to monitor the Net Search Extender environment:

get status

Issuing the `db2nx get status` command provides the following information about the status of the database:

- Whether it is enabled for Net Search Extender
- The tables that have text columns enabled, and the text column and index names

```
Database is enabled for 'DB2 Net Search Extender'

Table DB2NX.SAMPLE is enabled
  IndexName   ColumnName
  ---------   ----------
  COMMENT     COMMENT
  TITLE       TITLE
  FPD         FORMAT_PUB_DATE

DES7800I  The command completed successfully.
```

For more detail on this command, refer to “GET STATUS” on page 60.

get index status

Issuing the `db2nx get index status` command provides all the status information relating to the index/setup parameters of one of the indexes that has been created.

```
Settings of index 'TITLE'

Table                = 'DB2NX.SAMPLE'
Column               = 'TITLE'
Index directory      = '/home/user1/db2nx/indices/'
Temporary directory  = '/home/user1/db2nx/indices/'
Key-column           = 'DOCID'
Order by             = 'DOCID ASC '
Optimize on:
1.  = TITLE
2.  = AUTHOR
3.  = COMMENT
4.  = FORMAT_PUB_DATE
5.  = PRICE
6.  = year ( last_updated )  as year_last_updated
Numeric:
1.  = PRICE
Tags                 = no tags defined

In-Memory Table 'TITLE'
ID    Address    Permissions    Creator  Group
IPC status from /dev/mem as of Wed Nov 29 15:35:05 EST 2000
m        29 0x4908580e --rw-r--r--    user1    group1
m        30 0x5308580e --rw-r--r--    user1    group1
m        33 0x5308580f --rw-r--r--    user1    group1
```

```
m        31 0x4408580e --rw-r--r--    user1    group1

DES7220I  Index is active.   DES7800I  The command completed successfully.
```

For more detail on this command, refer to “GET INDEX STATUS” on page 59.

Running the Net.Data sample macro

Pre-requisites

The Net.Data sample macro can only be run if you have Net.Data installed to run with a Web server on your machine.

To run the Net.Data sample macro `nxsample.d2w`:

1. Copy `db2nx/samples/nxsample.d2w` to your Net.Data macro directory.
2. Open the macro for editing.
3. Update the values of the following variables:

```
DATABASE =  "your_database"
LOGIN    =  "your_user_ID"    Note: Remove this line, if already
                                    configured in Live connection
PASSWORD =  "your_password"   Note: Remove this line, if already
                                    configured in Live connection
```

4. Run the macro `http://your_hostname/cgi-bin/db2www/nxsample.d2w/main` in your Web browser.

After each Net.Data search, the connection from the DB2 client to the server is closed; DB2 reopens the connection for the next search. To enhance performance, you can use Net.Data’s Live Connection to keep the connection open after each search.

See “Using your Net.Data search function” on page 28 for further information.

Running the Java sample programs

In the first example, the Java sample program `NXSample.java` is used. The second example then shows you how to run the Java sample program with ranking `NXSampleRank.java` on a client machine.

To run the Java sample program `NXSample.java` on the database server

1. Set up the environment to run DB2 Java programs according to the Building Java Applications and Applets section of the *DB2 Applications Development Guide.*
2. Copy `db2nx/samples/NXSample.java` to a work directory and ensure that the `CLASSPATH` environment variable points to that directory.
3. To compile the sample program, run:

```
javac NXSample.java
```

4. To start the sample program, run:

```
java NXSample yourDatabase "yourSearchTerm"
```

Sample search calls

```
java NXSample yourDatabase "\"kid\""
java NXSample yourDatabase "(\"bestseller\" | \"pulitzer\") & \"book\""
java NXSample yourDatabase "\"kid\" & not \"duck\""
```

The index name `comment` is used by default. You can change the index name and the other search parameters in the variable-declaration section of the source file.

To run the Java sample program `NXSampleRank.java` on a DB2 client

1. Set up the environment to run DB2 Java programs according to the Building Java Applications and Applets section of the *DB2 Applications Development Guide* for the appropriate platform.
2. Copy the sample program to a working directory of the client machine to which the DB2 instance owner has write access. Ensure that the `CLASSPATH` environment variable points to that directory.
3. To compile the sample program, run:

   ```
   javac NXSampleRank.java
   ```
4. To start the sample program, run:

   ```
   java NXSampleRank yourDatabase yourSearchTerm
   ```

Note

Search terms need to be included in a single set of quotation marks (″ ″). See the previous Java sample `NXSample.java` for details. For the syntax of search terms, see Chapter 6, “Search syntax,” on page 35.

Stored procedures for searching

Underlying the provided samples are calls to a DB2 stored procedure. The calls provide DB2 Net Search Extender with four different types of search functionality:

- A standard search (no ranking)
- A standard search with tracing (no ranking)
- A standard search with ranking
- A standard search with ranking and tracing

These stored procedures can then be used within Net.Data, Java, or DB2 CLI applications integrated into DB2.

The signatures for the stored procedures are shown below.

Standard search without ranking

The following stored procedures are used:

- `db2nx.textsearch` Standard search
- `db2nx.textsearch_t` Standard search with tracing

```
DB2NX.TEXTSEARCH (
   OUT TOTALDOCS         INTEGER      ,
   IN  SEARCHTERM        VARCHAR(32000) ,
   IN  MAXHITCOUNT       INTEGER      ,
   IN  MAXINTERMHITCOUNT INTEGER      ,
   IN  STARTROWNUM       INTEGER      ,
   IN  MAXROWSTORETURN   INTEGER      ,
   IN  INDEXNAME         CHAR(50)     ,
   IN  INDEX_DIRECTORY   CHAR(50)     ,
   IN  TMP_DIRECTORY     CHAR(50)     ,
   IN  SQLSTATEMENT      VARCHAR(500) ,
   IN  DATASOURCE        INTEGER      ,
   OUT WORDCOUNTS        CHAR(250)
)
```

Note that the signature for DB2NX.TEXTSEARCH_T is identical.

The parameters for the procedure are explained in conjunction with using the Net.Data sample macro and the sample Java program. See “Using your Net.Data search function” on page 28 for further information.

Standard search with ranking

The following stored procedures are used:

- db2nx.textsearch_r Standard search with ranking
- db2nx.textsearch_rt Standard search with ranking and tracing

```
DB2NX.TEXTSEARCH_R (
   OUT TOTALDOCS         INTEGER      ,
   IN  SEARCHTERM        VARCHAR(32000),
   IN  MAXHITCOUNT       INTEGER      ,
   IN  MAXINTERMHITCOUNT INTEGER      ,
   IN  STARTROWNUM       INTEGER      ,
   IN  MAXROWSTORETURN   INTEGER      ,
   IN  INDEXNAME         CHAR(50)     ,
   IN  INDEX_DIRECTORY   CHAR(50)     ,
   IN  TMP_DIRECTORY     CHAR(50)     ,
   IN  SQLSTATEMENT      VARCHAR(500) ,
   IN  RANKOPERAND       INTEGER      ,
   IN  DATASOURCE        INTEGER      ,
   OUT WORDCOUNTS        CHAR(250)
)
```

In this stored procedure, there is an additional parameter for ranking, RANKOPERAND. Note that the signature for DB2NX.TEXTSEARCH_RT is identical.

The parameters for the procedure are explained in conjunction with using the Net.Data sample macro and the sample Java program. See “Using your Net.Data search function” on page 28 for further information.

Chapter 4. Creating and maintaining Net Search Extender indexes

This chapter provides an overview of the tasks involved in using Net Search Extender. This includes the following:

- Preparing a subsystem for Net Search Extender search
- Creating a text index
- Maintaining the Net Search Extender environment
- Index maintenance

DB2 requires you to specify a codepage for the subsystem. Net Search Extender supports all codepages that DB2 supports on the respective platforms.

Enter the Net Search Extender commands at the UNIX system services prompt.

When a `db2nx` command is issued, a connection is always made to the subsystem (and implicitly to the location name) specified in the dataset, or file referenced to by using the DSNAOINI environment variable. The database parameter of Net Search Extender administration commands must match the location name information given through DSNAOINI file.

Creating a text index

Before preparing a subsystem, ensure that a Net Search Extender instance has been created.

The job `DNELSAMP` and shellscript `nxsampleenable` from Chapter 3, “Running the sample programs,” on page 11 are used as examples in this section.

1. **Create and load a table.**

   The following table is created by `DNELSAMP`:

```
db2 "create table db2nx.sample(\
         docid            INTEGER NOT NULL,\
         author           VARCHAR(50),\
         title            VARCHAR(50),\
         subject          VARCHAR(100),\
         comment          LONG VARCHAR,\
         format_pub_date  LONG VARCHAR,\
         price            DECIMAL(8,2),\
         last_updated     timestamp,\
         PRIMARY KEY (docid) )"
```

2. **Identify the text data and related columns.**

   Identify the following information:

   - The table name and text column name where specific text data resides. The column type must be CHAR, VARCHAR, LONG VARCHAR, or CLOB.
   - The primary key of the table, and an index name of your choice.

   Optional information includes:

   - Tagging. Tags are used to identify sections of your document text that you want to limit searches to. Refer to “ENABLE TEXT COLUMN” on page 53 for details. A maximum of 250 tags can be specified. In `nxsample`, three tags are defined, `zzformat, zzpub, and zzdate`.
   - Optimized columns whose data is to be loaded into memory.

- If you plan to index numeric data from your table, the data types must be INTEGER, DOUBLE, or DECIMAL. See “ENABLE TEXT COLUMN” on page 53 for further information.
- Order-by columns, if the search results are always to be sorted in a predefined order. If no order-by columns is specified, then order-by key column is the default.
- A location for the created index, if you do not want to store the index in the default directory. For different subsystems, use different directories. This avoids a potential naming problem if you want to use the same index name for indexes of different subsystems.

3. **Calculate the amount of resources needed.**

   To calculate the amount of resources needed when creating an index, refer to “Calculating the amount of resources needed” on page 24.

4. **Verify and set the environment variables.**

   A connection is always made to the subsystem specified in the dataset, or file referenced by the DSNAOINI environment variable. Refer to the DB2NX_DB2CCSID environment variable at “Creating an instance” on page 6. Make sure that you have set this variable.

5. **Enable the subsystem.**

   At the command prompt, enter: `db2nx enable database` to prepare the subsystem, which creates the Net Search Extender administration tables.

6. **Enable the text column - create a Net Search Extender index.**

   Create an index by using the ENABLE TEXT COLUMN command described in “ENABLE TEXT COLUMN” on page 53.

Tip

The full list of options for the `db2nx` command is displayed when you enter the command `db2nx help`.

The command returns:

```
db2nx db2nx-command

List of db2nx commands
HELP
ENABLE DATABASE  [database [USER user USING password]]
                 [TABLESPACE tablespace]
DISABLE DATABASE [database [USER user USING password]]
GET STATUS       [DATABASE database [USER user USING password]]
ENABLE TEXT
COLUMN           table text-column INDEX index USING key-column
                 [TAGS ( tag [{,tag} ...] ) ]
                 [OPTIMIZE ON ( opt-column [ {,opt-column} ...] ) ]
                 [NUMERIC ( num-column [ {,num-column} ...] ) ]
                 [ORDER BY column {ASC|DESC} [{,column {ASC|DESC}} ... ]]
                 [DIRECTORY directory [TEMP DIRECTORY temp-directory ]]
                 [FASTRECOVERY]
                 [DATABASE database [USER user USING password]]
                 [INCREMENTAL] [TABLESPACE tablespace]
DISABLE TEXT
COLUMN           INDEX index [DATABASE database [USER user USING
                 password]]
UPDATE INDEX     index [DATABASE database [USER user USING password]]
                 [INCREMENTAL [COMMITCOUNT commitcount]]|[REORG]|[RESET]
GET INDEX STATUS index [DATABASE database [USER user USING password]]
ACTIVATE INDEX   index [DATABASE database [USER user USING password]]
DEACTIVATE INDEX index [DATABASE database [USER user USING password]]
```

Examples:

```
db2nx enable text column DB2NX.SAMPLE comment index comment using DOCID
  optimize on (title,author,comment,format_pub_date,price,
  {year(last_updated) as year_last_updated})  numeric (price)
  order by DOCID ASC

db2nx enable text column DB2NX.SAMPLE title index title using DOCID
  optimize on (title,author,comment,format_pub_date,price,
 {year(last_updated) as year_last_updated}) numeric (price)
 order by DOCID ASC

db2nx enable text column DB2NX.SAMPLE format_pub_date index fpd using DOCID
  TAGS  (zzformat,zzpub,zzdate)
  optimize on (title,author,comment,format_pub_date,price,
  {year(last_updated) as year_last_updated})  numeric (price)
  order by DOCID ASC
```

where `comment`, `title` and `format_pub_date` are the columns containing the text data to be indexed for a later Net Search Extender search, and `docid` is the primary key of table `db2nx.sample`

If data is expected to change frequently, you may need to use the `INCREMENTAL` keyword; see “Incremental index update” on page 22 for further information.

For more details about the parameters, see the description of the “ENABLE TEXT COLUMN” on page 53.

Note

All keywords used on the command line, such as the location name of the subsystem, table name, column name, index name, tag name, directory, and so on, must be specified using SBCS (Single Byte Character Set) characters.

The indexes can now be searched.

Monitoring the Net Search Extender environment

Net Search Extender has two administration commands to provide you with status information:

- GET STATUS for information about the status of the database
- GET INDEX STATUS for information about the status of a Net Search Extender index

Displaying the status of the database

When you need information about the status of the database, enter:

```
db2nx GET STATUS
```

This displays a list of created indexes. It shows which index belongs to which table column.

```
Database is enabled for 'DB2 Net Search Extender'

Table DB2NX.SAMPLE is enabled
  IndexName   ColumnName
  ---------   ----------
  COMMENT     COMMENT
  TITLE       TITLE
  FPD         FORMAT_PUB_DATE

DES7800I  The command completed successfully.
```

Displaying the settings and the status of an index

When you need to determine which parameters you used during `enable text column` and the current status of an index, enter:

`db2nx GET INDEX STATUS` *`index-name`*

Here is an example of the information displayed as it appears on a OS/390 system:

```
Settings of index 'TITLE'

Table                = 'DB2NX.SAMPLE'
Column               = 'TITLE'
Index directory      = '/home/user1/db2nx/indices/'
Temporary directory  = '/home/user1/db2nx/indices/'
Key-column           = 'DOCID'
Order by             = 'DOCID ASC '
Optimize on:
1.  = TITLE
2.  = AUTHOR
3.  = COMMENT
4.  = FORMAT_PUB_DATE
5.  = PRICE
6.  = year ( last_updated )  as year_last_updated
Numeric:
```

```
1.  = PRICE
Tags                 = no tags defined

In-Memory Table 'TITLE'
ID    Address    Permissions    Creator  Group
IPC status from /dev/mem as of Wed Nov 29 15:35:05 NFT 2000
m        29 0x4908580e --rw-r--r--    user1    group1
m        30 0x5308580e --rw-r--r--    user1    group1
m        33 0x5308580f --rw-r--r--    user1    group1
m        31 0x4408580e --rw-r--r--    user1    group1

DES7220I  Index is active.   DES7800I  The command completed successfully.
```

Note

You will see a slightly different output for indexes that were created with the FASTRECOVERY or INCREMENTAL options.

Maintaining indexes

These are the administration tasks for maintaining existing indexes:

- **Activating or deactivating an index**

  An index must be activated before you can search on it. This is done automatically when you create an index using the ENABLE TEXT COLUMN command. However, in the following cases you must activate an index explicitly:

  – When the index has been deactivated explicitly by the DEACTIVATE INDEX command
  – When the system has been restarted

  To activate an index, run:

  ```
  db2nx ACTIVATE INDEX myIndex
  ```

  To deactivate an index, run:

  ```
  db2nx DEACTIVATE INDEX myIndex
  ```

  When an index is activated, considerable amounts of data may be loaded into memory. To prevent the system from running out of resources, you should explicitly deactivate indexes that are not being used. For example, there may be some applications that require two indexes, but never both indexes at the same time. In such cases, you should switch indexes by deactivating one index and activating the other.

  For large indexes, use the FASTRECOVERY option to minimize the time taken to activate an index.

  See point “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

- **Disabling a text column to delete an index that is no longer needed**

  To disable a text column to delete an index, run:

  ```
  db2nx disable text column index-name
  ```

  Note

  As you can have multiple indexes on a text column, the command uses *index-name* to determine, which one(s) are deleted.

- **Disabling a database when all the indexes are no longer needed**

To disable a database, run:

```
db2nx disable database database-name
```

- **Updating an index**

  Update an index whenever the content of its associated text column changes. Net Search Extender does **not** automatically update indexes.

  To update an index, run:

  ```
  db2nx UPDATE INDEX myIndex
  ```

  When you update an index, the in-memory table associated with that index is recreated. This means that you cannot search the index while it is being updated.

  However, if you want to frequently update an index containing large volumes of data, see the following section, “Incremental index update.”

Incremental index update

As Net Search Extender always indexes all the data in the text column, the update process is expensive in system resources for large volumes of data. If this data changes frequently, then updating the index can cause operational problems.

To overcome this constraint, use incremental index update to reduce the time required to update an index. Note that, as the index needs to be enabled for incremental update when it is first built, you cannot change an existing index into an incrementally updatable index; see “ENABLE TEXT COLUMN” on page 53.

Creating an index with incremental index update enabled introduces the following changes to the database:

- Triggers on the table containing the text column are introduced to keep track of the database changes.
- An additional table is added; This is used to log changes to the database.

As the updating of indexes is not automatic, use the UPDATE INDEX command with the `INCREMENTAL` keyword, see “UPDATE INDEX” on page 62.

This only processes data additions, changes and deletions from the selected columns in the index, which considerably reduces the time required to update the index and therefore, the time during which the index is unavailable for searching.

Working with incremental index update

Creating an incremental index on a table or view

To use the incremental index update feature, you need to rebuild existing indexes adding the INCREMENTAL option on the ENABLE TEXT COLUMN command.

Additional space is reserved in the in-memory table at index creation time to keep rows that are added or updated later. The maximum amount of memory to use for the complete in-memory table needs to be specified by the DB2NX_MAXSHAREDMEM environment variable before running the ENABLE TEXT COLUMN command. If DB2NX_MAXSHAREDMEM is not explicitly set, the default amount allocated is only 1 MB.

For example, `export DB2NX_MAXSHAREDMEM=104857600` This will reserve an in-memory table of 104857600 bytes (100 MB). If the number of rows that are copied to the in-memory table, or the amount of memory needed by the optimized

columns exceeds this maximum, a memory allocation error message appears. Refer to “Calculating the amount of resources needed” on page 24 to check how much memory you need for your index.

After the index has been created, the GET INDEX STATUS command can be used to get details on the amount of memory that is available in the in-memory table for new and updated rows.

Note that this applies to both, ordinary indexes and those that are created using the FASTRECOVERY option.

Performing changes to the index subsystem table

Use DB2 commands to make changes to the table. For example:

- **Index was created on a table**

  If the index was created on a table, DB2 triggers and an index specific log table DB2NX.INX<indexname>_UPDATES are implicitly created. The triggers cause appropriate entries to be added to the log table whenever a row of the indexed table is deleted, updated, or added. The changes are applied to the Net Search Extender index using the UPDATE INDEX command.
- **Index was created on a view**

  If the index has been created on a view, only the log table, and not DB2 triggers are created, since DB2 does not allow creating triggers on a view. Therefore, entries have to be explicitly added to the log table whenever a row is deleted, updated, or added to the view. For security reasons, the entries should not be added directly to the log table but to an updatable view that is created on the log table at indexing time.

  The name of the updatable view is DB2NX.log<indexname> and consists of the following rows:

  ```
  OPERATION     CHAR        NOT NULL DEFAULT 'I'
  TS            TIMESTAMP   NOT NULL DEFAULT CURRENT TIMESTAMP
  uniquekey     uniquekey   NOT NULL
  ```

  The following values have to be added:

  – `OPERATION`: the change operation on the source table. Possible values are: I (Insert), U (Update), D (Delete)

  – `TS`: the timestamp of the changeoperation

  – `uniquekey`: value of the unique key column of the new, updated, or deleted column. Note that `uniquekey` is the same unique key used when creating the index.

  If the value of `uniquekey` has to be updated in the indexed table, two events have to be added to the log table view. The first one is a delete operation using the old value of the key column, the second one an insert operation using the new value of the key column.

> **Note**
> You can use an incremental index created on a table and look into the log table for a detailed view of which entries were added. In this case, the log table entries are added automatically.

Check that all changes to the indexed table are correctly reflected in the log table. Missing events or incorrectly entered events will cause incomplete or incorrect search results.

When you make changes to the indexed table, the index is not synchronous with the table until you have run the UPDATE INDEX command with the INCREMENTAL option.

Updating an index

Use the UPDATE INDEX command with the INCREMENTAL option to update the index. This computes all the events in the log table and performs the appropriate changes to the index. After the command has completed, there are no rows left in the log table. The index is synchronous with the indexed table again. Note that if there is not enough space left in the in-memory table to keep the new or updated rows, the command fails. In this case, you need to reorganize the index.

Reorganizing an index

Because the additional space that is reserved in the in-memory table for new or modified rows is limited, you must issue the UPDATE INDEX command with the REORG option from time to time. You must check the output of the GET INDEX STATUS command regularly to see how much reserved space is left in the in-memory table.

If you delete rows from your base table or view, and are running incremental update, these rows still require space in the in-memory table. Reorganizing does not free this space. In order to free all memory, you must recreate the index from time to time.

Resetting an index

The UPDATE INDEX command with the RESET option can be used whenever an index becomes unavailable because of an unexpected error during incremental update or index reorganization. It recovers the index and thereafter, you can issue the command that failed again.

Recreating an index from scratch

Use the UPDATE INDEX command without any option to recreate the complete index from scratch.

Calculating the amount of resources needed

Memory requirements

When an index is created, considerable amounts of data may be loaded into memory (actually, shared memory) of the machine on which the subsystem is running.

The amount of shared memory required by an index depends on the following parameters:

- The number of rows in the subsystem table in which the indexed documents are stored
- The width of the key column specified by the USING parameter of the ENABLE TEXT COLUMN command when the index was created
- The width of the columns specified by the OPTIMIZE ON parameter of the ENABLE COLUMN command

In addition to the amount of data put into shared memory, DB2 Net Search Extender allocates memory during the indexing process when retrieving data from DB2.

Note

All operating systems have an upper limit for the amount of addressable memory allocated per process which is also strongly dependent on the applications running in the process. The message stating that there is not enough memory available for the index update process to continue, unfortuantely appears quite a while after the process has began, as it takes some time to calculate the memory required in advance.

To achieve higher performance, data is retrieved from DB2 in blocks of 100 rows at a time.

To calculate the upper limit of the amount of shared memory required by an index, use the following formula (all values are in bytes):

```
(sum of width of all optimized columns + width of key column + 44) *
number of rows + 14.000
```

Note that in the case of numeric columns, the values are handled as strings, for example, the integer value 123000 has a width of 6 bytes.

For incremental index update, the number of rows to calculate is greater as you need to consider the rows that you plan to add in future as well.

The formula calculates the upper limit, and does not consider that Net Search Extender optimizes the memory requirements by not saving trailing spaces. The effective amount of required memory can be much less than the amount calculated by the formula above.

You can limit the maximum amount of shared memory used for one index by setting the DB2NX_MAXSHAREDMEM environment variable. If the number of rows copied to the in-memory table or the amount of memory needed by the optimized columns exceeds this maximum, a merrory allocation error message appears. Note that the DB2NX_MAXSHAREDMEM variable has a special meaning if you are creating an index for incremental index update. Refer to “Incremental index update” on page 22 for additional information.

You may need to adjust the UNIX System Services parameters related to shared memory segments, such as MAXASSIZE and IPCSHMMPAGES in BPXPRMxx, and memory mapped files, such as MAXMMAPAREA, according to the calculated amount of shared memory size required.

It is important to retain the shared memory of only those indexes that are currently needed for search. You can free the shared memory of an index that is currently not needed by issuing the DEACTIVATE INDEX command. Note that no search is possible on an index while it is deactivated. To get the index data back into shared memory and make it searchable again, run the ACTIVATE INDEX command.

If your operating system has been restarted, all indexes are deactivated.

Also note that other applications may be using system memory.

DB2 requirements

Net Search Extender creates a global temporary table in a temporary database containing all the rows of the result set. Whenever an in-memory search is performed, this temporary table is populated. Make sure the temporary table space and temporary database are large enough to store all the temporary result set tables, and that the maximum size for a temporary table is set to an appropriate value.

For each index, some auxiliary tables are created in the tablespace specified when creating the text index. For more information, refer to the TABLESPACE option at “ENABLE TEXT COLUMN” on page 53.

Disk space requirements

The minimum requirement is 1.3 times the data stored in the text column in addition to the numeric columns you want to index.

With FASTRECOVERY, add the in-memory table requirements obtained when applying the formula for calculating the upper limit (see Memory requirements above) to your minimum disk space requirement.

With REORG, you need to double the amount required for the minimum disk space and the in-memory table.

Chapter 5. Searching Net Search Extender indexes

This chapter provides an overview of search related tasks. These include:

- Creating the Search function
- Running the Search function
- Optimizing search performance

Searching an index

You search a Net Search Extender index by calling one of the stored procedures `textSearch`, `textsearch_t`, `textsearch_r`, or `textsearch_rt` in one of the following ways:

- Use the DB2 CALL statement as a static SQL statement, or run it from a CLI program using functions like SQLPrepare() or SQLExecute().

However, more comfortable ways of interfacing to the stored procedure are provided by the following methods:

- Using a Net.Data macro
- Using a Java program

This section describes how to make a search using the last two methods to call a stored procedure. For each method, two examples are used, one with ranking and one without ranking.

Creating a Net.Data search function with ranking

To make use of the stored procedure within a Net.Data macro, define a Net.Data function called `search` with the following parameter layout:

```
%FUNCTION(DTW_SQL) search (OUT INTEGER totalDocs,
                           IN VARCHAR(500) searchTerm,
                           IN INTEGER maxHitCount,
                           IN INTEGER maxIntermediateHitCount,
                           IN INTEGER startRow,
                           IN INTEGER maxRowsToReturn,
                           IN CHAR(50) indexname,
                           IN CHAR(100) indexDirectory,
                           IN CHAR(100) tmpDirectory,
                           IN VARCHAR(500) sqlStatement,
                           IN INTEGER rankOperand
                           IN INTEGER dataSource,
                           OUT CHAR(250) wordCounts,
                           OUT outTable)

                    returns (RESULT) {
  CALL DB2NX.TEXTSEARCH_R(outTable)


  %REPORT(outTable){
  <center> The first column is $(V1). The second column is $(V2). </center>
  %}
%}
```

The line `CALL DB2NX.TEXTSEARCH_R()` runs the stored procedure with ranking. To run the stored procedure without ranking use `CALL DB2NX.TEXTSEARCH()`.

Note

In the parameter layout displayed above, an additional line has been added for ranking:

```
IN INTEGER RANKOPERAND
```

If ranking is not required, this line **must not** be present, as the sequence and number of parameters is important for DB2 to be able to correctly handle the stored procedure.

The results are returned to the `%REPORT` section where you can format the display of the output or use Net.Data's default report feature.

Using your Net.Data search function

To use the search function that you have just created, place the Net.Data function call where you would like to issue the search in your Net.Data macro.

```
@search(totalDocs, searchTerm, maxHitCount, maxIntermediateHitCount,
        startRow, maxRowsToReturn, indexname, indexDirectory,
        tmpDirectory, sqlStatement, rankOper, dataSource, wordCounts, outTable)
```

For a functional description of the parameters used, refer to "Parameters of the search stored procedure" on page 35.

Example of a Net.Data search function call without ranking

The Net.Data ranking search function call returns rows 0 to 50 of search results from search term `stemmed form of "test"` and returns the columns ProductId, Title, Author, Format, and Year.

The results come from DB2 (data source ″1″).

```
@search(outTotalDocs, "stemmed form of \"test\"", "32000", "32000", "0", "50",
            "title",   "/home/myuserid/indexes", "/home/myuserid/tmp",
            "select ProductId, Title, Author, Format,  year(PublicationDate)
            as Year from db2nx.sample
            where productid in (%s)
            order by Title, Author for read only",
            "1", outWordCounts, outTable)
```

Note

The search with ranking works in a similar way to the above example.

Creating a Java search function to search a subsystem

Calling the stored procedure `textSearch` from a Java program is very similar to the way you call the stored procedure in Net.Data. Refer to the sample Java source code provided in the `samples` directory. If you look at the source code in the sample file `NXSample.java`, three sections of code relevant to the definition and calling of the stored precedure can be identified:

- A section on default parameters; modify the parameters that are passed to the stored procedure here.

- A section on the call statement for the stored procedure. The parameters set in the previous section are passed on here.
- A later section that defines the signature of the stored procedure.

The parameter descriptions used in the Net.Data example also apply to the Java example. For further information, see “Using your Net.Data search function” on page 28.

The sample source file `NXSampleRank.java` has a similar structure, but uses the stored procedure for search with ranking instead.

For an example of a Java call, see “Running the Java sample programs” on page 13.

Solving textSearch problems

To solve textSearch problems:

- Check the error log.
- For more information, do a trace.
- Check the error messages.
- Check ownership and permissions.
- Check shared memory.

Checking the error log

If an error occurs, information is written to a log file in the directory specified by the `tmpDirectory` parameter of the stored procedure. The name of the log file is the same as the index name and has the extension `.LOG`

For example, if the temporary directory is `/tmp` and the index name is `COMMENT`, the log file is `/tmp/COMMENT.LOG`

If no `tempDirectory` parameter is specified, the error log is written to `/tmp` directory.

Tracing

To get more detailed information during search, use the tracing variants of the search functions `TEXTSEARCH_T` (without ranking) and `TEXTSEARCH_RT` (with ranking).

The search trace function adds additional information to the log file. Note that the search function with tracing runs slower than without.

Error messages

If you get one of the following message during search:

- `SQL0104N An unexpected token "VALUES) as " was found ...`, check that the startRow is not equal to the result size. The startRow default value is 0.
- `DES7302N index does not exist`, check whether you issued a numeric search, but did not enable a numeric index.

Ownership and permissions

One of the more common problems during search is the incorrect ownership and permission of the index files. This is the index files directory structure (relative to the specified index directory):

```
./<INDXNAME>.shm
./<INDXNAME>/Fullmain/gtr.KEY/
                     /gtr.POS/
                     /gtr.LEY/
                     /gtr.QOS
                     /ItemMain/gtr.MEY/
                     /gtr.ROS
                     /gtr.NEY/
                     /gtr.SOS
```

For FASTRECOVERY indexes there are also the following files:

```
<INDXNAME>.SHM73
<INDXNAME>.SHM83
<INDXNAME>.SHM97
<INDXNAME>.SHM98 (if INCREMENTAL UPDATE enabled)
...
```

For an example, see `$DB2NX_INSTOWNERHOMEDIR/db2nx/indices` after running `nxsample`.

All files in this directory structure must be readable by the DB2 instance owner. If a different user ID is used for running the stored procedure, these files must also be readable by the owner of `$DB2NX_INSTOWNERHOMEDIR/sqllib/adm/.fenced`

Shared memory monitoring

As Net Search Extender makes extensive use of shared memory, this resource needs to be closely monitored to avoid paging and negative effects on other applications:

- Use `vmstat`.

Identify shared memory segments belonging to indexes that you want to keep. Delete those that you do not want to keep by deactivating the corresponding indexes, or if the indexes are no longer required, by disabling the corresponding text columns.

Query problems

To avoid query problems:

- Ensure that the index is activated.
- Ensure that the query statement is not too big (the limit of returned data for a query is 32700 bytes, the DB2 limit on the size of the VALUES statement). If it is, reduce the number of returned rows using `maxhitcount`, and reduce the number of `optimized for` columns (this requires re-enabling of the subsystem).
- Ensure that the directory name specified with the directory parameter during enable text column is the same as the `indexDirectory` input parameter of the textSearch procedure call.

Optimizing search performance

To optimize the search performance on a Net Search Extender enabled database, consider the following aspects of index creation and of the search call:

- **Optimize performance on certain table columns**

The stored procedure `TEXTSEARCH` returns a result table which contains information about the documents found by a search. The columns of that result table are either retrieved from the DB2 table or from an in-memory table. This is triggered by the `dataSource` parameter value.

Retrieving the result table from memory can be much faster than retrieving it from the original table. “ENABLE TEXT COLUMN” on page 53 provides an OPTIMIZE ON parameter that lets you specify a list of columns to be stored in memory, but you should use this feature carefully to avoid running out of system resources.

- **Avoid memory paging**

  The most significant advantage of Net Search Extender is that you can search without accessing the hard disk of the database server. This is possible by holding the required data in memory. However, if more memory is needed than is provided by the hardware, the operating system starts paging out parts of memory to disk. This negates the advantage in performance.

  Keep in mind that other applications may be running on your server that also use considerable amounts of memory. See item “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

- **Limit the search result**

  Limit the search result by setting the parameters `maxHitCount` and `maxIntermediateHitCount` of the stored procedure `textSearch`. This is useful when one of the search terms has a large number of matches. See “Using your Net.Data search function” on page 28 for more information.

- **Use Live Connection**

  If you are using Net.Data, use the Live Connection feature to prevent a new connection to the DB2 server from being re-established for every search call.

- **Use the search trace function only when necessary**

  If you do not need trace information, do not use the search trace function textSearch_t; instead, use the stored procedure textSearch. This function can be very useful, but it degrades search performance. It is described in “Solving textSearch problems” on page 29.

- **Disk layout**

  For better performance during indexing and searching, it is advisable to distribute the DB2 tables, the DB2 indexes and the Net Search Extender indexes across separate physical disks.

- **Monitoring the performance of WLM managed DB2 stored procedures**

  For more information, refer to *DB2 for OS/390 and z/OS Administration Guide V7, Chapter 36 Monitoring and tuning stored procedures and user-defined functions*.

- **Increase search performance in multithreaded search applications**

  If your applications issues multiple concurrent searches from multiple threads, you should set the DSNAOINI parameter MULTICONTEXT=1 to allow for multiple queries to be performed in parallel within multiple WLM address spaces. Note that MULTICONTEXT must be set to 1. Refer to page 7 for more information.

Optimizing index performance

To optimize index performance on a Net Search Extender enabled database, consider the following aspects of index creation:

- **Incremental index update**

If you have enabled your Net Search Extender index for incremental index update and there are many rows in the log table (more than 1000), you can increase the performance considerably by creating the index on the unique key column of the log table DB2NX.INX<index name>_UPDATES. Refer to “ENABLE TEXT COLUMN” on page 53 for more detail.

Part 2. Reference

Chapter 6. Search syntax

This section deals with:

- Parameters of the search stored procedure
- Description of the search syntax
- Using masking characters, capitals and blanks in a query

Parameters of the search stored procedure

Below is a functional description of the search stored procedure parameters. There are two types of search, with and without ranking. The only parameters that are different for each type of search are:

`sqlStatement`
`RankOper`

totalDocs The total number of documents found by the full-text query.

The totalDocs variable does not always reflect the total number of documents returned by the query. If the search term parameter is combined with an SQL statement, the SQL statement often decreases the result list and this change is not reflected in the totalDocs value.

searchTerm The search argument.

maxIntermediateHitCount
The maximum possible result size of a ranked result list. This parameter only has an effect when using the rank Stored Procedures textSearch_r or textSearch_rt. The maximum value is 64000 and a value of 0 means no restriction.

High values for maxHitCount can lead to an out-of-memory error during searches. To avoid this problem, reduce the maxHitCount value.

maxHitCount The value of the maxIntermediateHitCount search parameter can be positive or negative. A positive value specifies the maximum count of term occurrences. A negative value specifies the maximum count of documents. For example, if you are searching for the term `lion` and maxIntermediateHitCount is set to 30, and `lion` appears 30 times in the first 10 documents, these 10 documents are returned. If maxIntermediateHitCount is set to –30, only 30 documents with the term `lion` are returned. If maxIntermediateHitCount is set to 0, there are no restrictions.

startRow The number of the row at which the result buffer is to start storing results. For example, if 1000 rows contain hits for a given search and you want results beginning at row 50, set startRow to 50. The startRow parameter assumes that the first row begins at 0. Hence, if you want the result set contained in the first row, set startRow to 0.

maxRowsToReturn
The maximum number of rows to return beginning at startRow.

indexname The index to be used. This is the name of the index you specified in the `enable text column` command.

indexDirectory
The directory where the indexes are stored. If you do not specify this parameter, the default directory is used: `$DB2NX_INSTOWNERHOMEDIR/db2nx/indices`.

tmpDirectory The temporary directory to be used by the stored procedure. The subsystem instance must have write access to this directory. On UNIX systems, the default value is `/tmp`. On Windows NT®, the default value is the directory specified in the `TEMP` environment variable.

sqlStatement An SQL statement that specifies the columns and rows of the result table returned by the stored procedure. The statement for a search **without ranking** has the structure:

```
select ... from ... where ... <key-column> in (%s) ...
```

For example:

```
select DOCID,TITLE,AUTHOR from DB2NX.SAMPLE where DOCID in (%s)
```

A statement for a search **with ranking** has the structure:

```
select RSCORE, ... from ... %s <key-column>
       where <key-column> in (%s) [order by RSCORE desc]
```

For example:

```
select RSCORE,DOCID,TITLE from DB2NX.SAMPLE %s DOCID \
       where DOCID IN (%s) AND PRICE > 19 ORDER BY RSCORE DESC
```

RSCORE is the calculated rank score of the documents and `key-column` is the unique key column specified in the USING clause of the ENABLE TEXT COLUMN command. The %s variable is expanded to a list of all key-column values relating to the documents found by the search.

Columns in the select statement are returned in the `%RESULTS` section of the Net.Data macro and accessed by referencing $(Vn), where n is the column number.

If `dataSource` is 0, this parameter is ignored.

Note

You can use any valid SQL statement in the search query. The statement is passed as it is to DB2, only `%s` is replaced by a list of the key column values of the qualified rows separated by commas. You can check your SQL statement by removing the `<key-column> in (%s)` condition.

You cannot combine attribute searches on the same numeric values in the same search argument.

rankOper A flag that specifies how to get the result set of documents scored. This can be seen in an example, based on the search term ″Albert″ & ″Bernard″ and the following document table:

Document Number	Document Content
1	Albert

2	Bernard
3	Albert and Bernard

0 no_ranking:

desfpssp!textSearch_r works in the same way as desfpssp!textSearch and only document 3, Albert and Bernard is returned.

1 fuzzy_ranking:

All the documents in the set that contain one or more of the search items are scored using the fuzzy operation. In this case, the logical operators are interpreted as fuzzy operators. This search returns all three documents, Albert, Bernard, and, Albert and Bernard, with the document 3 ranked highest.

Use this parameter value with care, especially when searching terms contain NOT clauses. The document containing the excluded term will still be in the result set, but with a low rank value.

2 strict_ranking:

The documents in the set obtained by the traditional strict logical operation are scored. This search returns only document 3, Albert and Bernard.

Note that ranking the result size is very time consuming.

dataSource A flag that determines if row values come from DB2 or from memory.

0 Results come from the in-memory table only.

All columns specified in the OPTIMIZED ON section are returned together with the unique key.

1 Results are retrieved from DB2 according to the parameter `sqlStatement`.

wordCounts An output variable that returns the number of hits for each search term. These are listed in the order in which the search terms were entered. For example, if ″history″ and ″Japanese″ are the search term, and wordCounts contains ″1000 210″ after the search, this means ″history″ was found in 1000 documents, and ″Japanese″ in 210 documents. Note that wordCount values returned from sectioned indexes are not section specific.

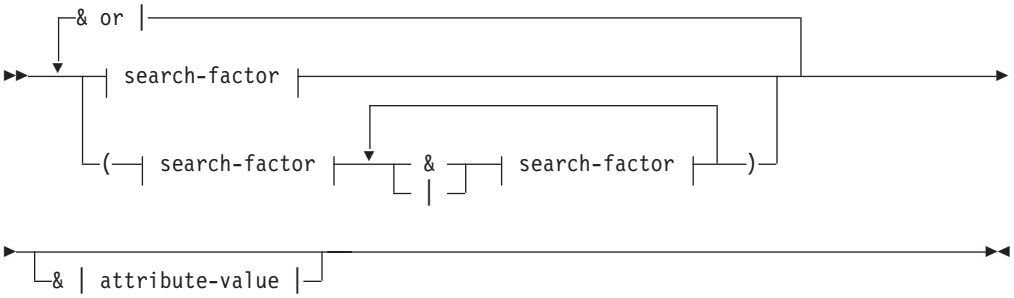
outTable An output variable that returns the results table.

This parameter is specific to Net.Data. If you call the stored procedure in a C program or in a Java program, do not use this parameter. See “Running the Java sample programs” on page 13.

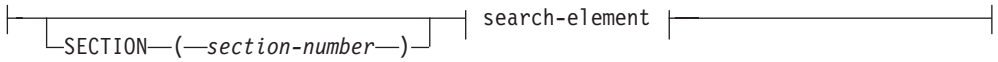
The search syntax described here is for the searchTerm parameter of the Net Search Extender stored-procedure.

Search argument

Command syntax

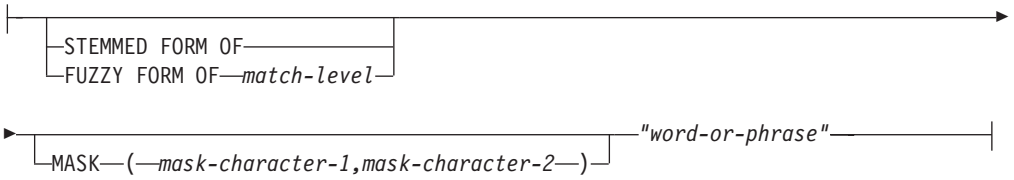


search-factor:

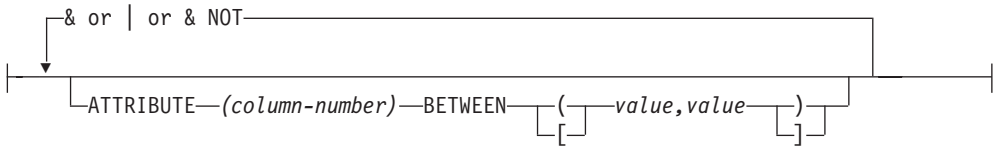


search-element:

search-primary:



attribute-value:



Command parameters

SECTION *(section-number)*

A keyword used to specify a section that the search is to be restricted to. To specify a section, use one of the section numbers displayed when running GET INDEX STATUS. The section IDs are determined by the the tag order during ENABLE TEXT COLUMN.

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 An operator that lets you exclude from your search text documents that contain a particular term.

IN SAME SENTENCE AS
A keyword that lets you search for a combination of terms occurring in the same sentence.

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 option supports English inflections of words only. You may receive results for other languages, but the stemming rules of the search engine apply to English only.

FUZZY FORM OF *match-level*
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`.

The *match-level* is a value from 1 to 100, where 100 means least fuzzy (exact) and 1 means most fuzzy. Only character strings whose matching levels are equal to, or higher than the specified limit are returned. The matching level is higher if more consecutive matching characters are contained in the string, and lower if fewer matching characters are found. For example, search for ″communicate″ and find ″communicating″ and ″communication″

When using the FUZZY FORM OF option, the first three characters of the search word must match thefound words. Performing a fuzzy search on ″wheather″ finds ″whether″, but not ″wether″ or ″weather″.

Fuzzy and wildcard searching cannot be combined, in other words, you cannot select the fuzzy search option using masking characters.

"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.

Masking characters (also known as ″wildcard″ characters)

A word can contain masking characters. You can use the MASK parameter to specify which characters are to be the masking characters. The default masking characters are:

% (percent)
Represents any number of arbitrary characters. If a word consists of a single %, then it represents an optional word of any length.

_ (underscore)
Represents any single arbitrary character.

A word cannot be composed exclusively of masking characters, except when a single % is used to represent an optional word.

When searching for terms containing wildcard characters, such as %e%, the expansion of the wildcard expression into candidate words for matches is limited to a maximum of 1000. This avoids long queries and storage-allocation problems.

MASK *(mask-character-1,mask-character-2)*
Masking characters that can be used in the search primary ″word or phrase″. For example: MASK (*,?)

mask-character-1 specifies the character that is to mask any number of arbitrary characters.

mask-character-2 specifies the character that is to mask any single arbitrary character.

Note that the MASK option has to be repeated for every word of the search term.

For more information on working with wildcard characters, see “Using Masking characters” on page 43

ATTRIBUTE *(column-number)* **BETWEEN** *(value,value)*

You can add numeric search terms that limit the results of the query. The attribute value is related to the column name order during ENABLE TEXT COLUMN. For example, use ’attribute(1)’ for searches against the first numeric column, attribute(2) for the second column, and so on. Numeric search terms can only be used in the following ways:

1. The numeric search terms must follow the last text search term in the searchTerm parameter.
2. The relational operators & (AND), | (OR), & NOT (AND NOT) can be used between numeric search terms. They must be separated from the numeric search terms by at least one blank space.

Note

The value of *column-number* corresponds to the position of the column name in the NUMERIC option during ENABLE TEXT COLUMN. If you used the following command for indexing, this can be seen below:

```
ENABLE TEXT COLUMN...NUMERIC (PRICE, WEIGHT, NUMBER_IN_STOCK)
```

you would search for items costing between $5 and $9, yet weighing less than 7kgs using

```
"... &ATTRIBUTE(1) BETWEEN [5.00,9.00] &ATTRIBUTE(2) BETWEEN [0,7]"
```

You cannot combine attribute searches on the same numeric values in the same search argument.

The brackets define the lower and upper limits of the interval. The left value defines the lower range value, the right the higher range value.

The brackets also define how the values inside are used. When searching for ranges of numbers, you can specify that the search **includes** the minimum or maximum values by using square brackets. To indicate exclusive matching of the minimum or maximum values (these values are **not included**), use parentheses.

- (means >x
- [means >=x
-) means <x
-] means <=x

The brackets can be paired in any combination ([with], [with), and (with], for example). Note that ranking is only applied to the results of the text portion of a given query.

'MIN' and 'MAX' can also be used as minimum or maximum values of a range to bound a search by the smallest or largest values.

When searching for exact matches against a numeric column, specify the desired number as both the minimum and maximum value, bounded by square brackets.

Below are some sample queries with further explanations of the numeric search function.

″book″ & ATTRIBUTE (1) BETWEEN [1234,1234]

This search returns all the rows where the text column contains ″book″ and where the first numerically indexed column contains the value 1234 in that same row.

″book″ & ATTRIBUTE (2) BETWEEN [MIN,20.00)

This search returns all the rows where the text column contains ″book″ and where the second numerically indexed column contains values within the range MIN and 20.00, but **not equal to** 20.00.

Examples of query strings:

Search for documents containing `emergency` in section 5, and containing `security department` with 60% or more matching.

```
section (5) "emergency" & fuzzy form of 60 "security department"
```

Searching for terms surrounded by quotation marks requires additional quotation marks in the search term. For example, if you want to find documents containing the term ″Hitchhikers Guide″ including the quotation marks, you have to enter the search string ″\″\″\″Hitchhikers Guide\″\″\″″. The outer quotation marks are removed by the command shell of the operating system. The search term is ″″″Hitchhikers Guide″″″ whereby the outer quotation marks are interpreted as the start and end identifiers of the search term. The remaining term, ″″Hitchhikers Guide″″ is sent to the internal search engine which transforms it to ″Hitchhikers Guide″. Note that if queries are not issued on the command line, the outer quotation marks and the backslashes can be omitted.

Search for either `birds` or `nature`, and also `government`.

```
( "birds"  |  "nature" ) &  "government"
```

Search for inflectional endings of the word `shock`, such as `shocked`, `shocking`, and so on:

```
stemmed form of "shock"
```

Find two words in the same sentence (this assumes that the sentences are separated by periods):

```
"computer" in same sentence as "book"
```

Use masking characters to find documents that contain words that begin with `night` and also contain words like `dreams`, `dreamy`, and so on:

```
"night%" & "dream_"
```

Perform the same search, but change the masking character:

```
mask (*,?) "night*" & mask (*,?) "dream?"
```

For details on how blanks and capitals are handled, refer to “Using capitals and blanks” on page 43

Note

Rows with NULL values in numeric columns cannot be searched for using the attribute value parameters.

Using Masking characters

Which documents are found

A series of letters, for example, a-z and A-Z, are treated as one word, but series of numbers or other characters are treated as separate words, even if they are side by side with no blank space between. For example, searching for ″2″ or ″&″ on a table which appears in an indexed document as ″abcd1234%&$?″ will be found, but searching for ″ab″ will not, because the search engine treats ″2″ and ″&″ (as well as ″1″,″3″, ″$″ etc.) as single words, but not ″ab″. Searching for the complete word, ″abcd″, or using a wildcard (″ab%″), will also retrieve the document. Correspondingly, ″abcd1″ will be found, but not ″abcd1%″. The reason is that wildcards are designed to replace one or more characters of a word, but not the complete word. If you are searching for just the ″%″ wildcard, it will not be treated as a wildcard, but as the ″percent″ character.

Selecting wildcard characters

The default wildcard characters used by Net Search Extender are '%' for strings of arbitrary length and '_' for single characters. These are also the default wildcard characters used in SQL. You can replace these characters with different characters using the MASK specifier. For example, the search term ' MASK (*,?) ″han?so*″ ' will find the word ″handsome″ because ″*″ and ″?″ replace the default characters ″%″ and ″_″. Also, the first parameter in the MASK statement ″*″ is the wildcard character for matching strings of arbitrary (including zero) length, whereas the second parameter ″?″ is the wildcard character for matching a single (non-zero) character.

Not all characters can be used as wildcard characters. Good choices are ″%″ and ″_″, or ″*″ and ″?″, but depending on the operating system, these characters may also have special meanings if issued on the command line. For example, the command line interpreter on Windows uses '%' as an indicator for variable names, which leads queries submitted from the command line to be misinterpreted, and resulting in an token error in Net Search Extender. In this case, use ″*″ and ″?″ when expanding your query using the MASK option

Result limit

When the number of differently found words reaches 1000 (the default value), the wildcard search stops and the result is returned. This is done to avoid wasting resources when the wildcard searches is too general, such as ″A*″.

Using capitals and blanks

Handling capital characters

The text indexes used by Net Search Extender are not case sensitive. Uppercase and lowercase characters are regarded as being the same.

Handling blank characters

Multiple blank characters between alphabetical or alphanumerical characters are regarded as one blank character. For example, ″communication manager″ is regarded to be the same as ″communication manager″. Blank characters between an alphabetical or alphanumerical character and another character are ignored. This is because a series of letters, for example, a-z and A-Z are treated as one word.

However, a series of numbers or other special characters are treated as different words, even if they are located side by side without a blank space, for example, ″Berlin2000″. Every number or other sign is also treated as a separate word. For example, ″&§$ IBM″ and ″& § $ IBM″ are regarded as being the same as ″&§$IBM″. The two groups ″&§$″ and ″IBM″ are still treated as separate words, since the first part consists of special characters whilst the second part consists of common characters.

Chapter 7. Administration commands

Command	Purpose	Page
ACTIVATE INDEX	Makes search possible after the index has been deactivated, or after the system has been rebooted	48
DEACTIVATE INDEX	Frees memory occupied by data associated with the index	49
DISABLE DATABASE	Deletes all indexes associated with a subsystem; the subsystem is no longer available for use by Net Search Extender	50
DISABLE TEXT COLUMN	Deactivates and deletes an index	51
ENABLE DATABASE	Prepares a database for use by Net Search Extender	52
ENABLE TEXT COLUMN	Prepares a text column for use by Net Search Extender and creates an individual text index for the column	53
GET INDEX STATUS	Displays the index settings and the status of an index	59
GET STATUS	Displays the status of a database	60
HELP	Displays help information	61
UPDATE INDEX	Updates and activates an index	62

This chapter is a reference for the Net Search Extender administration commands. While Chapter 5, “Searching Net Search Extender indexes,” on page 27 provides an overview of Net Search Extender administration, this chapter includes information about the syntax of each command and describes the related parameters.

Administration commands

Enter the Net Search Extender commands at the UNIX system service prompt. You prefix each command with `db2nx`, for example, `db2nx ? ENABLE TEXT COLUMN`.

Except for the `HELP` command, each administration command always establishes a subsystem connection to the subsystem specified in the dataset or file referenced to via the `DSNAOINI` environment variable.

Net Search Extender environment variables

The environment variables that can be set include:

DSNAOINI
On OS/390, a connection is always made to the subsystem (and implicitly to the location name) specified in the dataset or file referenced to by the DSNAOINI environment variable. The subsystem parameter of the Net Search Extender administration commands must match the location name information in the DSNAOINI file.

ICU_DATA
Path where the DATA-File containing the conversion-tables is located.

DB2NX_INSTOWNERHOMEDIR
The directory in which the Net Search Extender instance was created. This is usually the home directory of the instance owner, for example, `/u/instanceownerusername`.

DB2NXPATH
Installation path, used to locate bind and message directory.

Obtional environment variables include:

DB2NX_MAXSHAREDMEM
Refer to “Calculating the amount of resources needed” on page 24 for details on how to set this value.

DB2NXINSTALLDIR
The instance installation directory, for example, `/u/instanceownerusername/db2nx`

DB2NX_DB2CCSID
Set to the CCSID of the DB2 subsystem. This CCSID is specified during the DSNHDECP installation job (known as the DSNTIJUZ job). If the DB2 Performance Monitor is installed, check the CCSID of the subsystem by:

- Selecting your subsystem using Spufi
- Selecting the DB2 Performance Monitor
- Selecting ″View online activity″ option 4
- Selecting ″Display system parameter″ option 5
- Selecting ″Application programming defaults″

Note that the CCSID of the table space that contains the tables that you want to index also needs to be set to the same value.

For a list of environment variables on OS/390, see “Creating an instance” on page 6.

Tracing

All `DB2NX` commands can be used in trace mode if the command is followed by an extra `trace` parameter. This parameter triggers the output of additional status and progress information. Note that using `trace` with the `DB2NX ENABLE TEXT COLUMN` command can produce potentially huge amounts of trace output, depending on the amount of data to be indexed.

ACTIVATE INDEX

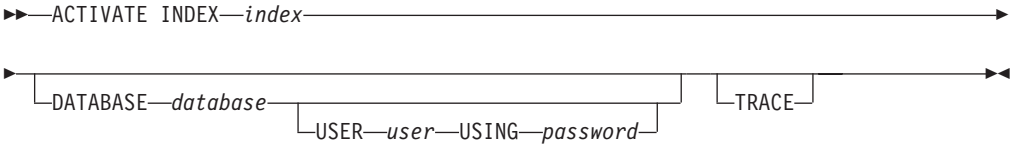
This command makes search on an index possible after the index has been explicitly deactivated by the DEACTIVATE INDEX command, or after the system has been restarted.

The command reads the key column and all the OPTIMIZE ON columns from DB2 into the shared memory.

Note

If the index was created with the FASTRECOVERY option, the ACTIVATE INDEX call is not needed.

Command syntax



Command parameters

index The index name specified when the index was created using the enable text column command.

DATABASE database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you want to work with.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

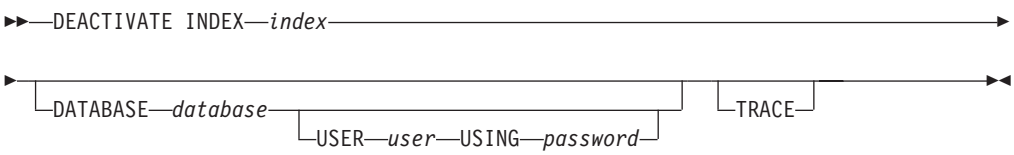
DEACTIVATE INDEX

This command frees the shared memory occupied by an index. After running this command, you cannot search on the specified index until you run the ACTIVATE INDEX command. This command does not affect the index itself, but only the contents of the in-memory table.

Note

If the index was created with the FASTRECOVERY option, the DEACTIVATE INDEX call is not needed.

Command syntax



Command parameters

index The index name specified when the index was created using the enable text column command.

DATABASE database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for thesubsystem you want to work with.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

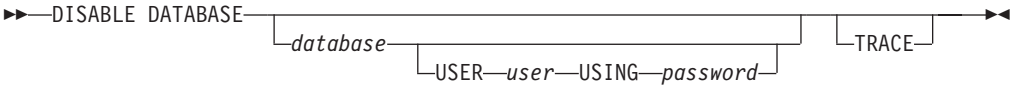
Usage notes

To prevent the system from running out of resources, you should explicitly deactivate indexes that are not currently being used, especially indexes with the `optimize on` parameter specified when the index was created. If the index was created with the FASTRECOVERY option however, this is not necessary. See point “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

DISABLE DATABASE

This command resets any preparation work done by Net Search Extender for a subsystem and disables all its text columns for use by Net Search Extender. All index files for this subsystem are removed from disk.

Command syntax



Command parameters

database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you are disabling.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

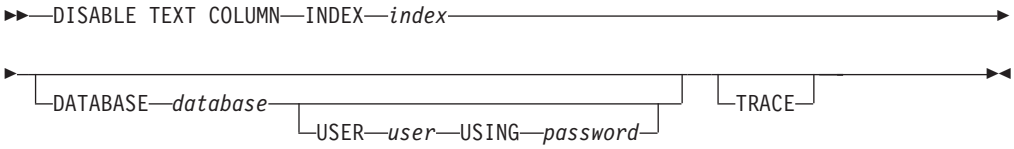
Usage notes

This command resets the connected subsystem so that it can no longer be searched by Net Search Extender. All Net Search Extender index files are deleted. To re-enable the subsystem for Net Search Extender searching, use the ENABLE DATABASE command.

DISABLE TEXT COLUMN

This command deletes a Net Search Extender index and frees memory occupied by the index.

Command syntax



Command parameters

INDEX index
The unique index name specified when the index was created using the enable text column command.

DATABASE database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you want to work with.

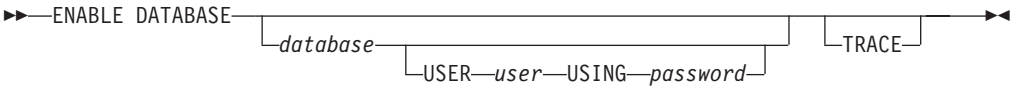
USING password
The password for the DB2 instance.

TRACE
Activates tracing.

ENABLE DATABASE

This command prepares a subsystem to be used for text search by Net Search Extender.

Command syntax



Command parameters

database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you are enabling.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

Usage notes

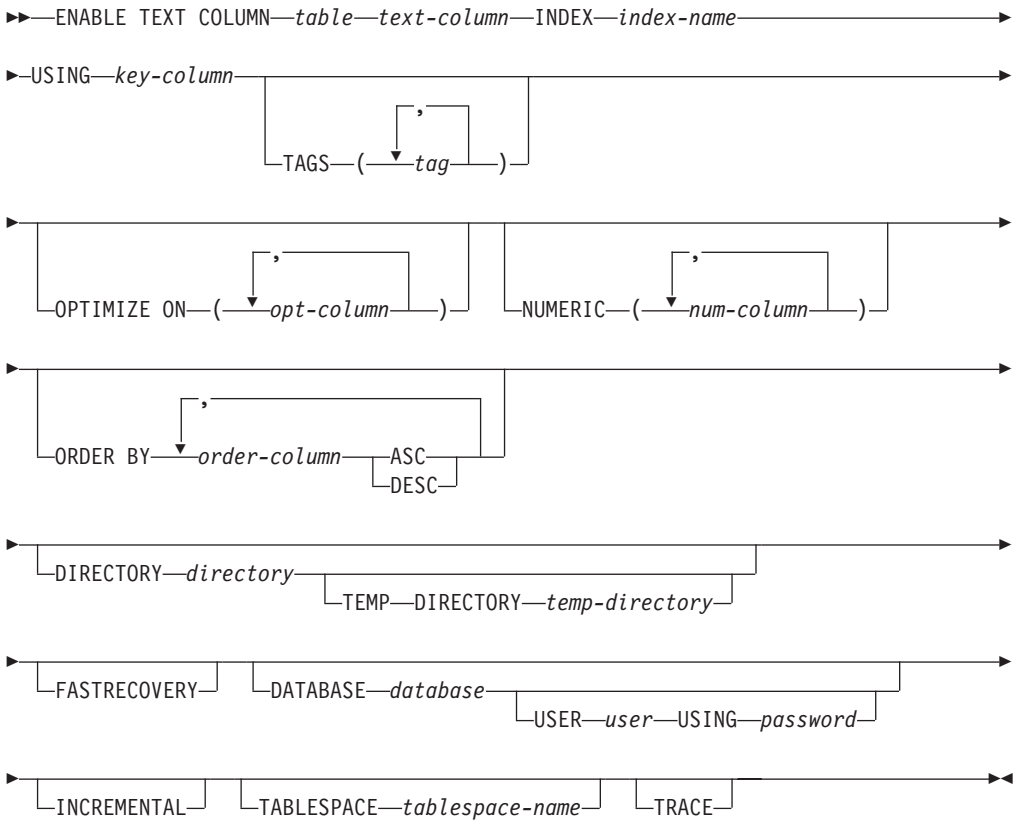
The `enable database` command creates the following tables:

- db2nx.fpcolumn
- db2nx.fptags
- db2nx.fpmemory
- db2nx.fpnumber

ENABLE TEXT COLUMN

This command creates an index for the specified text column.

Command syntax



Command parameters

table The name of the text table in the connected subsystem 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
The name of the column to be enabled for use by Net Search Extender. This column must be of the type CHAR, VARCHAR, LONG VARCHAR, or CLOB.

INDEX index-name
A name to be given to the index. The name must be unique in the specified directory and not longer than 8 characters. The length is restricted to 7 characters if the INCREMENTAL option is used. When choosing names for indexes, subsystems, and columns make sure that they are not db2nx command keywords, such as index, numeric or optimize.

USING key-column
The name of the key column used to establish a relation between the indexed documents and the subsystem rows. In combination with the keyword INCREMENTAL, the key-column must be unique as it must be possible to join a record in shared memory with a record in the source table. If possible, use the primary key of the table.

The key column must be of type INT, SMALLINT, BIGINT, CHAR, VARCHAR, TIMESTAMP, DATE, or TIME.

Note that the contents of the *key-column* are held in an in-memory table while the index is activated. If the width of your key column is very large, you may run into system limitations. See point “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

TAGS tag

The names of up to five tags specified in the documents to support sections. If the format looks like:

```
xtitlex Document Title...

xbodyx Main text of document...

xfooterx Some footer information...
```

where there is a space after the tags `xtitlex`, `xbodyx`, and `xfooterx`, and there are two blank lines separating each section, then this is what the `TAGS` portion of the command would look like:

```
... TAGS (xtitlex, xbodyx, xfooterx)
```

Note

Tags that would appear as regular words in the documents should be avoided.

A TAG value must be specified using only SBCS characters.

The section number specified as the searchTerm parameter for the Net Search Extender stored procedure call corresponds to the order of the tags. In this example, `xtitlex` becomes 1, `xbodyx` becomes 2, and `xfooterx` becomes 3. See Chapter 6, “Search syntax,” on page 35 for the use of the section number.

Use the `get index status command` to get a list of the tags you specified during the creation of an existing index.

OPTIMIZE ON opt-column

Up to 15 columns to be held in an in-memory table of the subsystem server. This enables the stored procedure to retrieve them in a result table without accessing the original DB2 table. This feature is an important contributor to the high search performance of Net Search Extender.

To take advantage of this feature, set the parameter `dataSource` of the `textSearch` stored procedure to 0 when issuing the search. If you don’t do this, the results are retrieved from DB2.

Note that the memory resources of your subsystem server are limited, and that other applications may be running on your server that are using parts of memory. So it is important to estimate the amount of memory required for your indexes; see point “Calculating the amount of resources needed” on page 24, ″Calculate the amount of resources needed″ for further information.

This option supports a wide range of datatypes. However, LOBs other than CLOB are not supported.

Instead of specifying an existing table column, you can instead specify an SQL expression. You can specify any expression allowed in the SELECT clause of an SQL query on the table containing the documents to be indexed.

Example: The following command creates an additional index for the sample table `db2nx.sample`, which is created by the sample program `nxsample`:

```
db2nx "ENABLE TEXT COLUMN db2nx.sample comment
                INDEX sample
                USING docid
                OPTIMIZE ON (title,{SUBSTR(comment,1,30)
                            as commentheader})"
```

Note that an SQL expression must be surrounded by braces { }.

NUMERIC num-column

A keyword to limit text search results to a range of values, or equal to a value. For example, search on the word `dog` but only return results between dates 19981201 and 20001201. Alternatively, search on `history` but only return books with price less than $50.00.

The following data types are supported:

- INTEGER
- DECIMAL
- FLOAT
- NUMERIC
- SMALLINT
- REAL
- DOUBLE
- CHAR
- VARCHAR

Note that with CHAR and VARCHAR data types, the contents of the column must be valid numeric strings.

For example, a subsystem column (data type INTEGER), called **date** would be entered as:

```
...NUMERIC (date)
```

Note

Single or multiple numeric columns can be used in the command. However, ensure that their order corresponds to the first, second, and so on numeric fields of the search term.

The maximum number of numeric columns is 20.

Also note that the JULIAN_DAY scalar function can be applied to column type DATE or TIMESTAMP to convert the contents to INTEGER. See the *SQL Reference* documentation for further information.

ORDER BY order-column ASC or DESC

The columns used to specify a sequence during indexing, so that

documents can be retrieved from the index in ascending or descending sorted order. When retrieving data from DB2 rather than from the in-memory table, this order must be used during a search to return the results in the requested order.

> **Note**
> The sort order cannot be preserved after an incremental update.
>
> Also note that if order mask columns have NULL values, then according to SQL standards these always rank highest in any given sort order.

DIRECTORY directory
The directory where the index is to be stored. If specified, the directory must exist. If you don't specify a directory, the default directory `db2nx/indices` is used.

TEMP DIRECTORY temp-directory
The directory where temporary index files are to be stored. If specified, the directory must exist. If you do not specify a directory, the default directory `/tmp` is used.

FASTRECOVERY
A keyword to enable faster response times for the `DB2NX ACTIVATE INDEX` command after a shut down or server crash. When this keyword is set, Net Search Extender stores data in memory-mapped files and not directly in shared memory. Note that the same size calculations apply for additional disk space as when calculating the amount of shared memory required; See "Calculating the amount of resources needed" on page 24 for further information. Also note that indexing and search performance is slower when this keyword is set.

If you specify the FASTRECOVERY option when you create your index, memory mapped files as opposed to ordinary shared memory is used to keep the in-memory table. In this case, both the ACTIVATE INDEX and DEACTIVATE INDEX commands are not supported as the operating system automatically optimizes which in-memory tables to keep in memory. If you want to create large in-memory tables, set the MAXMMAPAREA parameter in the operating system to a minimum of 16777216.

DATABASE database
The location name of the subsystem you want to work with. The `DSNAOINI` environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem that contains the table.

USING password
The password for the DB2 instance.

INCREMENTAL
A keyword to enable the Net Search Extender index for incremental update. Triggers and a log table are created to identify and protocol changes within the subsystem for update, delete and change requests. The in-memory tables are organized differently for incremental update. When specifing this option, a subsequent UPDATE INDEX command only selects added, deleted or changed rows in the subsystem for the indexing process. If

INCREMENTAL is not specified when enabling a text column, UPDATE INDEX always indexes the entire text column.

Note

Impact during Administration: Due to additional processing in logging added, deleted, or changed rows in the subsystem, all subsequent subsystem inserts, deletes, or updates are expected to be slower. See also “Incremental index update” on page 22.

Impact during query: After an update the query performance is lower for all subsequent queries expecting the correct sort order. See also “UPDATE INDEX” on page 62 for information on how to minimize this problem.

TABLESPACE tablespace-name

A keyword to define the DB2 tablespace where the log table and any additional internal tables are created. If not defined, the log table will be created with DB2 defaults, which is the first tablespace created by the user. The tablespace must be created before the command is executed. Note that if you do not specify a tablespace, the DB2 default tablespace is used which can become too small.

Two tables are created in the tablespace for each Net Search Extender index:

- DB2NX.INX<index name>_REACT is needed to keep the in-memory table consistent with the base table. It consists of two columns, one integer and one of the same type as the key column of the index. The number of rows is same as for the base table.
- DB2NX.INX<index name>_UPDATES is only created if you specify the INCREMENTAL option. It keeps track of the insert, update, and delete operations on the base table. Refer to “Working with incremental index update” on page 22 for details.

TRACE

Activates tracing.

Note

The `ENABLE TEXT COLUMN` command writes progress information to `stdout` indicating the amount of data indexed with time stamps to track performance information.

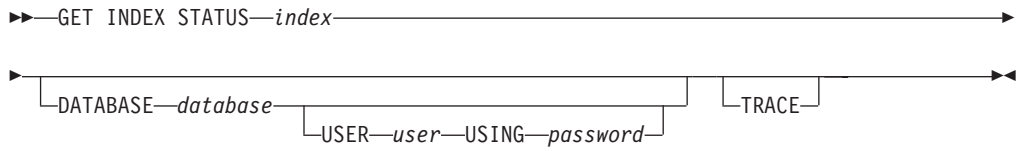
Example:

```
Fri Dec  8 11:56:51 2000 Full Start...
Fri Dec  8 11:56:51 2000 Item Start...
Calculating In-Memory Table size...
0

Reading data from DB2NX.SAMPLE and creating index...
0
Completed reading database
Fri Dec  8 11:56:52 2000 Full Flush <DOC :   6>< 1M>- Sort .
                                                    - Writing -  .
Fri Dec  8 11:56:52 2000 Full X_start...
Fri Dec  8 11:56:52 2000 Full Flush <PART :101>< 1M>- Sort .
                                                    - Writing -  .
Fri Dec  8 11:56:52 2000 Full End
-----------------------------------------------------------------------
Sort1                             0 (sec)
Sort2                             0 (sec)
Write                             0 (sec)
Other(including caller's time)    1 (sec)
Total                             1 (sec)
```

GET INDEX STATUS

This command returns information about the index, including its current status.

Command syntax



Command parameters

index The index name specified when the index was created using the enable text column command.

DATABASE database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you want to work with.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

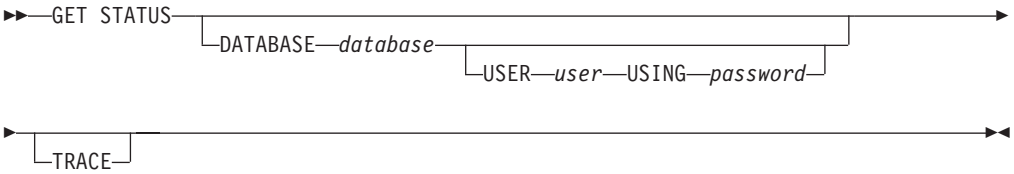
Examples

```
Settings of index 'TITLE'

Table                = 'DB2NX.SAMPLE'
Column               = 'TITLE'
Index directory      = '/home/user1/db2nx/indices/'
Temporary directory  = '/home/user1/db2nx/indices/'
Key-column           = 'DOCID'
Order by             = 'DOCID ASC '
Optimize on:
1.  = TITLE
2.  = AUTHOR
3.  = COMMENT
4.  = FORMAT_PUB_DATE
5.  = PRICE
6.  = year ( last_updated )  as year_last_updated
Numeric:
1.  = PRICE
Tags                 = no tags defined

In-Memory Table 'TITLE'
ID    Address    Permissions    Creator  Group
IPC status from /dev/mem as of Wed Nov 29 15:35:05 NFT 2000
m        29 0x4908580e --rw-r--r--    user1    group1
m        30 0x5308580e --rw-r--r--    user1    group1
m        33 0x5308580f --rw-r--r--    user1    group1
m        31 0x4408580e --rw-r--r--    user1    group1

DES7220I  Index is active.   DES7800I  The command completed successfully.
```

GET STATUS

This command displays information about the enabled status of a subsystem, and shows a list of its indexes.

Command syntax



Command parameters

database
Thelocation name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem for which you are getting the status.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

Examples

```
Database is enabled for 'DB2 Net Search Extender'

Table DB2NX.SAMPLE is enabled
  IndexName   ColumnName
  ---------   ----------
  COMMENT     COMMENT
  TITLE       TITLE
  FPD         FORMAT_PUB_DATE

DES7800I  The command completed successfully.
```

HELP

This command displays a list of the administration commands provided by Net Search Extender.

Command syntax

Command parameters

None.

Examples

The output of the `db2nx help` command looks like this:

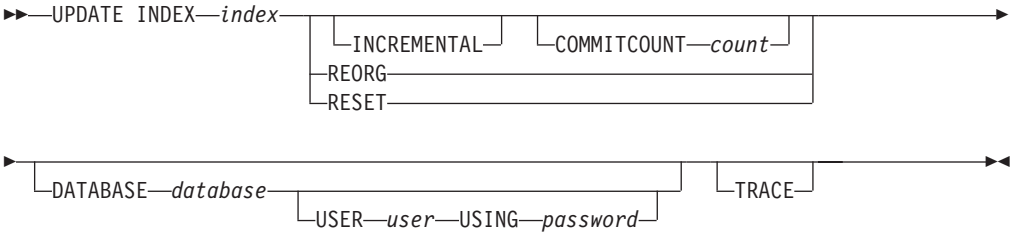
```
List of db2nx commands
HELP
ENABLE DATABASE      [database [USER1 user1 USING password]]
                     [TABLESPACE tablespace]
DISABLE DATABASE     [database [USER1 user1 USING password]]
GET STATUS           [DATABASE database [USER1 user1 USING password]]
ENABLE TEXT COLUMN   table text-column INDEX index USING key-column
                     [TAGS ( tag [{,tag} ...] ) ]
                     [OPTIMIZE ON ( opt-column [ {,opt-column} ...] ) ]
                     [NUMERIC ( num-column [ {,num-column} ...] ) ]
                     [ORDER BY column {ASC|DESC} [{,column {ASC|DESC}} ... ]]
                     [DIRECTORY directory [TEMP DIRECTORY temp-directory ]]
                     [FASTRECOVERY]
                     [DATABASE database [USER1 user1 USING password]]
                     [INCREMENTAL] [TABLESPACE tablespace]
DISABLE TEXT COLUMN  INDEX index [DATABASE database [USER user USING password]]
UPDATE INDEX         index [DATABASE database [USER1 user1 USING password]]
                     [INCREMENTAL [COMMITCOUNT commitcount]]|[REORG]|[RESET]
GET INDEX STATUS     index [DATABASE database [USER1 user1 USING password]]
ACTIVATE INDEX       index [DATABASE database [USER1 user1 USING password]]
DEACTIVATE INDEX     index [DATABASE database [USER1 user1 USING password]]
```

UPDATE INDEX

This command starts indexing using the same settings defined during ENABLE TEXT COLUMN. Using this command without any specific options creates the index files from scratch.

Performance consideration for incremental index update: use this command for a complete reorganization of the in-memory and index-data for keeping high query performance whenever the ORDER BY option is used during ENABLE TEXT COLUMN.

Command syntax



Command parameters

index The index name specified when the index was created using the enable text column command.

INCREMENTAL

A keyword to specify that only added, deleted or changed rows in the subsystem are used for the in-memory and index update. Additional shared memory areas are used for in-memory usage and additional index files are used for the text-search engine. Depending on the amount of data and number of incremental index update commands, the internal system must be reorganized to reduce memory usage and to maintain high indexing speed and query performance.

A REORG is undertaken whenever internal thresholds are exceeded. See also the REORG keyword.

The physical sort order specified by the ORDER BY option of the ENABLE TEXT COLUMN is not retained by the INCREMENTAL index update. As the correct sort order is calculated during search, perform a complete index update to avoid any performance degradation.

REORG

A keyword to reorganize the in-memory data and index-data. Shared memory is optimized and the additional index files are merged with the main index files. To be effective, the INCREMENTAL keyword must have been specified during ENABLE TEXT COLUMN, otherwise this keyword is ignored.

The correct sort order must still be calculated during search. A REORG is also triggered by an INCREMENTAL index update whenever internal thresholds are exceeded. See also INCREMENTAL index update.

RESET
A keyword to use whenever the index is unavailable because an unexpected error occurred during update, or while the index was being reorganized.

COMMITCOUNT count
A keyword to define the number of log records to be deleted from the log table after command completion and before a COMMIT command is performed. If no commitcount is set, all the records are deleted. If there are too many records in the log table, this may cause a LOG-FILE-FULL condition. If this occurs, specify a lower count value and execute the command again.

DATABASE database
The location name of the subsystem you want to work with. The DSNAOINI environment variable is always used.

USER user
The user ID of the DB2 instance with DBADM authority for the subsystem you want to work with.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

As with the `ENABLE TEXT COLUMN` command, using `TRACE` can produce huge amounts of trace output depending on the amount of changed data to be indexed.

Usage notes

Because updating an index makes it unavailable for search, your application is responsible for not allowing searches during updates. For example, if your application provides the results of text queries over the Web, then the search pages of the Web site should not be available during the update period.

If permanent availability of the search function is an issue, create two indexes and switch between them during UPDATE INDEX.

UPDATE INDEX

Chapter 8. Messages

Each message has a message identifier that consists of a prefix (DES), the message number, and a suffix letter.

I Information message

N Error (or “negative”) message

Note that the message portion of the sqlca structure, which is used to pass the error message back to the client, holds a maximum of 70 characters, so some messages might be truncated. However, the full error message gets logged to the log file.

DES0249N A DB2 Net Search Extender internal error has occurred. Line number: '*nnn*'.

Explanation: An unexpected internal error occurred in the Net Search Extender demon.

What to do:

1. Check your system resources and retry the last command.
2. Try using `nxstop` to stop Net Search Extender, then reactivate all indexes again.

DES0250N A DB2 Net Search Extender internal error has occurred. Reason code: '*rc* '.

Explanation: An unexpected internal error occurred.

What to do:

1. Check your system resources and retry the last command.
2. Use the operating system reason code to solve the problem.

DES0251N The syntax of the command is: DESFPDEM {START | STOP} DESFPDEM {START | STOP | POST}

Explanation: The command syntax is incorrect.

What to do: Change the command syntax and try again.

DES0255N DB2 Net Search Extender has already been started.

Explanation: The Net Search Extender is already running.

What to do: No action required

DES0256N No system resources. Reason code: '*rc*'.

Explanation: No more system resources are available.

What to do: Use the operating system return code to solve the problem.

DES0257N Memory allocation failed. Line number: '*nnn* '.

Explanation: A system call to get memory failed.

What to do: Check the system resources and retry the last command.

DES0259N DB2 Net Search Extender has not been started.

Explanation: To work with Net Search Extender you first have to start it.

What to do: Call `nxstart` to start Net Search Extender.

DES0261N The environment variable '*variablename*' is not set. See DB2 Net Search Extender documentation for more information.

Explanation: The specified environment variable has not been set.

What to do: Check the system settings. You may need to restart the operating system.

DES0263N DB2 Net Search Extender demon has abnormally terminated.

Explanation: Caused by an unexpected error.

What to do: Check the system settings. You may need to restart the operating system.

DES0264N Function *functionname* ended with return code *rc*.

Explanation: A function ended with the specified return code.

What to do: Use the return code of the function to solve the problem.

DES7000N Failed on database open.

Explanation: An attempt to connect to a database failed.

What to do: Change the specified database name or check the environment DB2DBDFT variable.

DES7001N Failed on disable database.

Explanation: A problem occurred with the administration tables while disabling a database.

What to do: Check that all the administration tables are available and that the entries can be read by this user ID.

DES7101N Memory could not be allocated.

Explanation: Large amounts of memory have had to be allocated during indexing, in particular if you are using a CLOB datatype. No more storage could be reserved for the application.

What to do: Use the environment variable DB2NX_ROWS_TO_FETCH which allows you to select the number of rows to be scroll fetched. The number must be between 1 and 100; if the environment variable is not defined, a default of 100 is used.

DES7103N In-memory table *table* could not be allocated.

Explanation: No shared memory could be reserved for the application.

What to do: Decrease the number of `optimize on` columns or call DEACTIVATE INDEX for indexes that are no longer required.

DES7104N In-Memory Table init error.

Explanation: The In-Memory Table could not be initialized.

What to do: Check if the system limits for shared resources are reached and, if necessary, free additional resources.

DES7105I No data found in table.

Explanation: No data was found in the specified table. No table rows have been indexed.

What to do: Check the table to ensure that it contains data.

DES7106N The column *columnname* is optimized more than once. Remove this column from the optimized section.

Explanation: Column *columnname* occurs more than once in the optimized section.

What to do: Remove one of the occurrences of the column from the optimized section.

DES7108N No column was found to join with the index.

Explanation: The key column following the USING tag is invalid or could not be used.

What to do: Check the key column following the USING tag, and, if necessary, change it.

DES7109N Invalid row count *rowcount* returned for DB2 table *schema.tablename*.

Explanation: The row count is invalid. It is either zero or greater than 539 251 480.

What to do: Use a different table or change the number of rows to be in the above range.

DES7110N Data Cache Overflow.

Explanation: DB2 Net Search Extender has not allocated enough memory for the user data.

What to do: Try to reduce the number of columns specified in the 'optimize on' brackets.

DES7111N Inode block Overflow.

Explanation: DB2 Net Search Extender has not allocated enough memory for the user data.

What to do: Try to reduce the number of columns specified in the 'optimize on' brackets.

DES7201N *token* is unexpected. Check the command syntax.

Explanation: An unexpected token was found.

What to do: Check the command syntax. Check if the search term has been specified in the correct codepage. Check the setting for the DB2NX_DB2CCSID environment variable in db2nxprofile described in "Creating an instance" on page 6.

DES7202N Parameter *parameter* is too long.

Explanation: The specified parameter is out of range.

What to do: Specify the parameter using a valid length.

DES7203N You reached the maximum number of OPTIMIZE ON columns.

Explanation: You specified more columns than allowed by the OPTIMIZE ON parameter.

What to do: Decrease the number of OPTIMIZE ON columns.

DES7204N You reached the maximum number of tags.

Explanation: You specified more tags than allowed by the TAG parameter.

What to do: Decrease the number of tags.

DES7205N 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.

DES7206N Please specify schema owner for table: OWNER.TABLENAME.

Explanation: There is an error with the table name.

What to do: Check the name of the table and try to specify the owner.

DES7207N Index *indexname* does not exist.

Explanation: You are trying to specify an index that does not exist.

What to do: Change the index name, then try again.

DES7208N Indexname *indexname* already exists.

Explanation: You are trying to specify an index name that already exists.

What to do: Change the index name, then try again.

DES7209N Database is not enabled.

Explanation: You are trying to use a database that was not enabled.

What to do: Enable the database and try again.

DES7210N Database is already enabled.

Explanation: You are trying to enable a database which was already enabled

What to do: Check that the database name is correct.

DES7211N You reached the maximum number of NUMERIC columns.

Explanation: The maximum number of NUMERIC columns has been reached.

What to do: Reduce the number of NUMERIC columns.

DES7212N Installation problem! Check *environmentvariable*

Explanation: You are trying to use Net Search Extender using an incorrect environment variable.

What to do: Check the specified environment variable.

DES7213N No database specified.

Explanation: There is no database information specified.

What to do: Either set the `DB2DBDFT` variable or specify the database in the command.

DES7215N Load of administration tables failed.

Explanation: There seems to be an inconsistency in the administration tables.

What to do: Check if all administration tables are available. See the table layout in “ENABLE DATABASE” on page 52.

DES7216N Directory *directory* does not exist.

Explanation: There was a directory specified which does not exist.

What to do: Check the directory name.

DES7218N File could not be created.

Explanation: A file could not be created on the file system.

What to do: Check that there is enough disk space available.

DES7219I Index *index* is already active.

Explanation: You are trying to activate an index that is already active.

DES7220I Index is active.

Explanation: You are trying to activate an index that is already active.

DES7221N Index is not active. No search is available.

Explanation: You are trying to use an index that is not active.

DES7222N CCSID *ccsid* not supported.

Explanation: CCSID of the database is not supported.

What to do: Create a database with database CCSID 819, 850 or 1252.

DES7230W DB2 Net Search Extender will expire in *nn* days.

Explanation: You are using a Try & Buy version of Net Search Extender.

What to do: You can continue to use this version until the expiry date.

DES7300N Index has not been activated.

Explanation: The stored procedure could not access the in-memory table associated with the index name passed to the stored procedure.

What to do: Initialize the in-memory table for the index by issuing the ACTIVATE INDEX command. If the command still fails, UPDATE the index or create it again (DISABLE and ENABLE the column). Verify that the directory you have specified contains the index files.

DES7301N SQL statement too long; please reduce the result set size.

Explanation: You have just requested a stored procedure to return rows. The stored procedure builds an SQL statement to either pull the rows from DB2 or to return the appropriate values from the in-memory table. However, you have requested so many rows that the SQL statement being built exceeds the maximum length of 32 KB. This limit is reached sooner for the in-memory table than for DB2.

What to do: Reduce the number of requested rows by using the maxRowsToReturn parameter for the stored procedure. You can step through the result set by calling the stored procedure multiple times with increasing value of the startRow parameter.

If the data source is the in-memory table associated with the index, you can also recreate the index with fewer (or smaller) optimization columns.

DES7302N retcode = *code*; *message*

Explanation: The search failed with return code *code* and message *message*.

What to do: Here are some of the possible causes:

- Improper search term syntax
- Illegal characters in the search string
- Null string passed as search term
- Bad index name
- Numeric index not enabled when a numeric search is issued

Check the search term syntax, index name, and so on.

DES7303N Bad value for parameter *maxRowsToReturn*

Explanation: The value of the parameter passed to the stored procedure was invalid.

What to do: Check the value of the parameter and make sure it is valid. For example, maxRowsToReturn must be a positive integer.

DES7304N Key column must be a unique key.

Explanation: The column specified in the ENABLE TEXT COLUMN command as the key column must be a unique key if the stored procedure is going to be called with dataSource=1, that is, if DB2 is the data source.

What to do: Use a key column that is a unique key if you want to use dataSource=1.

DES7305N Error parsing numeric search specification

Explanation: Syntax of the input string is invalid.

What to do: Check the command syntax.

DES7310N An unexpected token was found at position *position* rc *rc*.

Explanation: The syntax of the input string is invalid.

What to do: Correct the input string and try again.

DES7311N Invalid masking character usage.

Explanation: The characters you have used to specify masking are not valid.

What to do: Change the characters used for the mask token.

DES7400N Internal error occurred in *filename* at line *linenumber*.

Explanation: An internal function has produced an internal program error.

What to do: Make a note of the file name and line number, then contact your IBM representive.

DES7401N Environment variable *variablename* not found.

Explanation: The variable *variablename* was not set in the current environment.

What to do: Check that you are using the correct user ID, and that the user environment has been changed.

DES7402N Environment variable *variablename* value length *length* is invalid.

Explanation: The environment variable value exceeds the maximum size of an internal buffer.

What to do: Change the value of the variable to a smaller value.

DES7403N Message file path length *directory and filename* is invalid.

Explanation: The path length exceeds the maximum size of an internal buffer.

What to do: Change the value of the path to a smaller value.

DES7404N Message file *messagefile* not found.

Explanation: The file *messagefile* could not be found.

What to do: Check that the installation is correct and ensure that no files have been removed.

DES7405N Data type of numeric column *col-name* not supported

Explanation: The data type of the numeric column is not supported.

What to do: Use one of the supported data types.

DES7450N Unable to convert data of codepage *codepage*. Reason: *rc*

Explanation: There is a conversion problem for the data of the specified codepage.

What to do: Take action on the reason code displayed with the message.

DES7451N Unable to convert data to UTF-8. Reason: *rc*

Explanation: There is a problem to convert data to UTF-8.

What to do: Take action on the reason code displayed with the message.

DES7452N Data conversion to unicode failed. Reason: *rc*

Explanation: There is a problem to convert data to unicode.

What to do: Take action on the reason code displayed with the message.

DES7453N Unable to open converter *convertername*. Reason: *rc*

Explanation: There is a problem to open the specified converter.

What to do: Take action on the reason code displayed with the message.

DES7454N Unable to open converter for CCSID *ccsid*. Reason: *rc*

Explanation: There is a problem to open a converter related to the specified CCSID.

What to do: Take action on the reason code displayed with the message.

DES7455N Unable to convert data from unicode to *format*. Reason: *rc*

Explanation: There is a problem to convert data from unicode to the specified data format.

What to do: Take action on the reason code displayed with the message.

DES7456N Unable to convert *format* query string to UTF-8. Reason: *rc*

Explanation: Unable to convert the query string data of the specified format to UTF-8 format.

What to do: Take action on the reason code displayed with the message.

DES7520N In-Memory Table status not available.

Explanation: Could not read information from the shared memory.

What to do: Deactivate the index and activate it again.

DES7521N Request failed to delete In-Memory Table *table*

Explanation: Shared memory could not be deleted.

What to do: Check the current status using `ipcs -s` and do a cleanup manually.

DES7523N The requested In-Memory Table would require *nnn* **bytes which exceeds system limits.**

Explanation: The shared memory could not be allocated.

What to do: Check the system resources and limits.

DES7525N In-Memory Table *table* **could not be attached.**

Explanation: The shared memory could not be attached.

What to do: For UNIX systems, check the SharedMemory ID using `ipcs -s` and try to clean up.

DES7526N MapViewOfFile failed with return code *rc***.**

Explanation: The shared memory could not be accessed.

What to do: Check the return code to isolate the problem.

DES7527N In-Memory Table SHM file could not be accessed.

Explanation: The SHM file of the index could not be accessed.

What to do: Check if the SHM file in the index directory exists and could be read.

DES7600I Index *index* **is not enabled for Incremental Update.**

Explanation: The index was not enabled using the keyword INCREMENTAL. No update is possible.

What to do: If the index should be updated, it must be recreated using the INCREMENTAL keyword.

DES7601I No Update needed.

Explanation: There have been changes since the last index update, but they do not need to be added to the index. For example, a record was added, but has subsequently been deleted since the last update run.

DES7602I No Update record found in Log-Table.

Explanation: There have been no changes since the last index update.

DES7605N No Unique Key for Source Table given.

Explanation: The column used as key column for the index is not unique. The index cannot be enabled for incremental update.

What to do: Use a unique column as the key column.

DES7606N Name of Shared Memory Directory not found.

Explanation: The name of the shared memory directory could not be found in the database. There must be a problem with the Net Search Extender administration tables.

What to do: Try to disable the index and then run the enable index again. If this error message appears with SQLCODE = -981, REASON 00C12216, remove the parameter MULTICONTEXT=1 from your DSNAOINI file.

DES7607N Name of Temp-Shared Memory Directory not found.

Explanation: The name of the temp-shared memory directory could not be found in the database. There must be a problem with the Net Search Extender administration tables.

What to do: Try to disable the index and then enable the index again.

DES7608N No Space left in Shared Memory.

Explanation: The shared memory resources have been exhausted. There is no space left.

What to do: Try a REORG. Alternatively, delete some data.

DES7610N DATA-Shared Memory Segment not found.

Explanation: A shared memory segment is missing, the index has been corrupted.

What to do: The index must be recreated.

DES7613N Can't allocate more Memory.

Explanation: The program requires more RAM than is currently available.

What to do: Free some memory by terminating unnecessary processes. Add swop space.

DES7614N Error when accessing TextSearch-Engine.

Explanation: There is a problem accessing the underlying search engine.

What to do: Check the installation directory against the installation file inventory.

DES7615N Error when accessing Database. \nSQLCODE: *sqlcode***nMessage:** *message***nSQLSTATE:***sqlstate*

Explanation: The database returned the described SQLCODE and message.

What to do: Refer to the *DB2 User Guide* to solve the problem.

DES7616N Can't open Shared Memory segment.

Explanation: The shared memory cannot be opened with write access.

What to do: Check the rights of the current user. Retry the operation with a user who has write access to the shared memory.

DES7800I The command completed successfully.

Explanation: The specified command completed successfully.

DES7801N The command failed.

Explanation: The command was not executed sucessfully.

What to do: Check the error message to get more information about the problem.

DES7802N The DB2 Net Search Extender license is invalid or has been expired.

ES7803N A DB2 Net Search Extender Try & Buy license found.

Explanation: You are currently working with a Try & Buy license.

What to do: If you intend to use Net Search Extender after the expiry date, upgrade to the normal version.

DES7804I Starting reorganization of index *index*

Explanation: The reorganization of the index has started.

DES7805I Reorganization of index *index* **has been ended successfully.**

Explanation: The index has been reorganized.

DES7806I Reorganization of index *index* **has ended with errors**

Explanation: The index has not been reorganized because an error has occurred.

What to do: Check the output for the errors. Fix the error and retry.

DES7807I Starting Incremental Update of Index *index*

Explanation: Starting incremental update for index *index*.

DES7808I Incremental Update of Index *index* **raised SHM-full-condition - starting implicit reorg.**

Explanation: The shared memory ran out of space while updates where being applied. The shared memory will be reorganized and the update will continue after the reorganization has completed.

DES7809I Incremental Update of Index *index* **has ended successfully**

Explanation: The index has been updated.

DES7810I Incremental Update of Index *index* **has ended with errors.**

Explanation: An error occured while the index was being updated.

What to do: Check the output for the error. Fix the error and retry.

DES7999N Internal error: Invalid functionid.

Explanation: The command parser found a function that is not valid.

What to do: Check the command syntax and try again.

DES9997N An SQL error occurred. SqlState: *state*
SQL Error code: *code*
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.

Explanation: An internal processing error occurred.

What to do: Collect the available information and report it to your IBM service representative.

Part 3. Appendixes

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other 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.

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 Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains 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 contains 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:

(c) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both:

DB2	DB2 Universal Database	Net.Data
IBM	OS/390	z/OS
RACF	MVS	

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

J

M

N

O

P

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

DB2 Universal Database for OS/390 and z/OS
Net Search Extender
Administration and Programming
Version 7

Publication No. SC27-1171-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC27-1171-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SC27-1171-01

Cut or Fold
Along Line

Program Number: 5675-DB2

SC27-1171-01