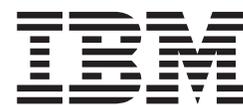


DB2 Universal Database for OS/390 and z/OS



Net Search Extender Administration and Programming Guide

Version 7

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 61.

First Edition, March 2001

This edition applies to DB2 Net Search Extender Version 7, program number 5675-DB2. This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2001. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	v
Who should use this book.	v
How to read the syntax diagrams	v
How to send your comments	vi

Part 1. Guide 1

Chapter 1. Introduction	3
Features	3

Chapter 2. Customization	5
System requirements.	5
OS/390 customization	5
Customizing	5
Creating an instance.	6
Workload Manager related tasks.	7
DB2 related tasks.	8

Chapter 3. Running the sample programs.	11
Running the sample database setup program	11
Monitoring the Net Search Extender environment	12
Running the Net.Data sample macro	13
Running the Java sample programs	13
Stored procedures for searching	14
Standard search (without ranking).	14
Standard search with ranking	15

Chapter 4. Using Net Search Extender	17
Preparing a database for Net Search Extender search	17
Searching the index.	19
Creating a Net.Data search function with ranking to search a database	20
Using your Net.Data search function	20
Example of a Net.Data search function call (without ranking)	23
Creating a Java search function to search a database	23
Solving textSearch problems.	24

Chapter 5. Administration overview	25
Monitoring the Net Search Extender environment	25
Displaying the status of the database.	25
Displaying the settings and the status of an index	26
Maintaining indexes	26
Backing up and restoring databases	27
Optimizing search performance.	27

Part 2. Reference 29

Chapter 6. Search syntax for the searchTerm parameter (SQLMM version)	31
Search argument.	32

Chapter 7. Administration commands	37
Environment variables.	37
Tracing	38
ACTIVATE INDEX	39
DEACTIVATE INDEX	40
DISABLE DATABASE	41
DISABLE TEXT COLUMN	42
ENABLE DATABASE	43
ENABLE TEXT COLUMN	44
GET INDEX STATUS	49
GET STATUS.	50
HELP	51
UPDATE INDEX	52

Chapter 8. Messages.	53
---------------------------------------	-----------

Part 3. Appendixes. 59

Notices	61
Trademarks	62

Index	65
------------------------	-----------

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.

About this book



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

How to send your comments

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

- Send your comments from the Web. Visit the Web site at:
<http://www.ibm.com/software/data/db2/extenders>

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

- Send your comments by e-mail to swsdid@de.ibm.com. Be sure to include the name of the book, the part number of the book, the version of the product, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
- Fill out a Readers' Comments form at the back of this book and return it by mail, by fax, or by giving it to an IBM representative. The mailing address is on the back of the form. The fax number is +49-(0)7031-16-4892.

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

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.

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 table columns in main memory at indexing time to avoid expensive paging at search time

Features

Net Search Extender provides complementary functionality to that provided by DB2 Text Extender, the other full-text retrieval search engine integrated into DB2. The key features of Net Search Extender are:

- Indexing
 - One very fast index type (Ngram)
 - Triggers are not used to dynamically update an index
 - Allows multiple indexes on the same text column
 - Indexing proceeds without locking data
- Search
 - Stored procedure on the server instead of UDFs
 - Allows word, phrase, stemmed, or fuzzy search
 - Identifies and restricts searching to sections within documents that have been marked by special tags
 - Supports Boolean or wildcard operations
- Search results
 - Lets you specify how the search results are sorted at indexing time
 - Lets you specify search result subsets when large data volumes are searched and large result lists are expected
 - Lets you set a 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.

However, if your solution requires any of the following features:

- Integrating SQL functionality into your text search
- Extensive linguistic search capabilities (tokenization, sentence recognition, term normalization, base form term reduction, stop-word filtering, and decomposition)
- Additional index types (linguistic and precise)
- Dynamic updating of indexes

you should consider using DB2 Text Extender, as they are not available in the Net Search Extender solution.

Chapter 2. Customization

This chapter describes how to customize DB2 Net Search Extender on OS/390®.

System requirements

Net Search Extender runs on one of the following versions of operating system:

- OS/390 V2.7 or higher

Net Search Extender requires that DB2 EE V7.1 is installed.

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.

The memory requirement depends on the number of documents to be indexed, and on the performance optimization of your indexes. See point 3 on page 18, "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. Some of these steps are carried out by a system programmer and some by a database administrator.

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/db2/db2nx
- /usr/lpp/db2/db2nx/IBM
- /usr/lpp/db2/db2nx/bin
- /usr/lpp/db2/db2nx/icu/lib
- /usr/lpp/db2/db2nx/icu/data
- /usr/lpp/db2/db2nx/nls
- /usr/lpp/db2/db2nx/samples
- /usr/lpp/db2/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

Customization

must not omit the default part of the directory names. The fully qualified name becomes, for example, /u/myname/usr/lpp/db2/db2nx/IBM.

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.

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/db2/db2nx/bin /u/instanceownerusername/db2nx/bin
```

```
cp /usr/lpp/db2/db2nx/bin/db2nxprofile /u/instanceownerusername/db2nx
```

```
ln -s /usr/lpp/db2/db2nx/icu /u/instanceownerusername/db2nx/icu
```

```
ln -s /usr/lpp/db2/db2nx/nls /u/instanceownerusername/db2nx/msg
```

```
cp -r /usr/lpp/db2/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_INSTOWNER

The userid of the instance owner

DB2NX_INSTOWNERHOMEDIR

The instance owners home directory, for example,
/u/instanceownerusername

DB2NXINSTALLDIR

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

DB2NX_DB2CCSID

The CCSID of text documents stored in DB2 which need to be indexed. Default is 500.

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. 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:DNE.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/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

Customization

- 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 DNEENVA. 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(DNEENVA). 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:

- 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.

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 DNEENVA ddname for the ENVFILE settings.

It is important to adapt the contents of the dataset referenced by DNEENVA according to your installation. See “Configuring the WLM-started task” on page 7 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.

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 can only be run after Net Search Extender has been installed and the relevant DB2 and Net Search Extender instances created. The sample programs are a means by which you can verify your installation.

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

The sample job `h1q.SDNESAMP(DNELSAMP)`, creates and populates the sample database and the `nxsampleenable` shellscript enables the data for search.

The Net.Data and Java samples can then be used to run searches against that data.

These are the sample programs:

h1q.SDNESAMP(DNELSAMP) and nxsampleenable

The job `h1q.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. Using Net Search Extender" 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.

Running the sample programs

The `h1q.SDNESAMP(DNELSAMP)job` and `nxsamp1eenable` shellsript perform the following tasks:

- Creates a table `db2nx.sample` in the specified database.
- Puts sample data into that table
- On Windows NT, starts the background daemon
- 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 `h1q.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 one of the indexes that have been created:

```
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.
```

Get Index Status

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

```
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.
```

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.

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 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

Running the sample programs

1. Set up the environment to run DB2 Java programs according to the Building Java Applications and Applets section of the *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 NXSamp1eRank.java
```
4. To start the sample program, run:

```
java NXSamp1eRank yourDatabase yourSearchTerm
```

Note

Search terms need to be included in a single set of " ". See the previous Java sample NXSamp1e.java for details. For syntax of the valid search terms, see "Chapter 6. Search syntax for the searchTerm parameter (SQLMM version)" on page 31

Stored procedures for searching

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

- 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 on the platform:

- db2nx.textsearch standard search
- db2nx.textsearch_t standard search with enhanced tracing

On OS/390

```
DB2NX.TEXTSEARCH (
  OUT TOTALDOCS          SMALLINT      ,
  IN  SEARCHTERM         CHAR(250)     ,
  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.

Running the sample programs

The parameters for the platform are explained in conjunction with using the Net.Data sample macro and the sample Java program, see “Chapter 4. Using Net Search Extender” on page 17 for further information.

Standard search with ranking

The following stored procedures are used on the platform:

- db2nx.textsearch_r standard search with ranking
- db2nx.textsearch_rt standard search with ranking and enhanced tracing

On OS/390

```
DB2NX.TEXTSEARCH_R (  
  OUT TOTALDOCS      SMALLINT      ,  
  IN  SEARCHTERM     CHAR(250)     ,  
  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 platform are explained in conjunction with using the Net.Data sample macro and the sample Java program, see “Chapter 4. Using Net Search Extender” on page 17 for further information.

Chapter 4. Using Net Search Extender

This chapter provides an overview of the tasks involved in using Net Search Extender on databases that you have created, or are about to create. This includes the following:

- Preparing the database for Net Search Extender search
- Enabling the database for Net Search Extender search (Creating an Index)
- Creating the Search function
- Running the Search function

DB2 requires you to specify a codepage for the database. Net Search Extender supports all codepages that DB2 supports on the respective platforms. See the *DB2 documentation* for a complete list of supported CCSIDs.

When a `db2nx` command is issued, a connection is always made to the subsystem specified in the dataset, or file referenced by the `DSNAOINI` environment variable.

For information on monitoring and maintaining indexes you have created, administration-related tasks and database backup and retrieval, see “Chapter 5. Administration overview” on page 25.

Preparing a database for Net Search Extender search

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

The job `DNELSAMP` and shellsript `nxsamp1enable` from “Chapter 3. Running the sample programs” on page 11 are used as examples in this section.

1. Design, create (tablespace and tables) and load a database with text data.

The following table is created in `DNELSAMP`:

```
db2 "create table $TABLE(
      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 44 for details. A maximum of five tags can be specified. In `nxsample`, three tags are defined, `zzformat`, `zzpub`, and `zzdate`.

Using Net Search Extender

- Optimized columns whose data is to be loaded into memory
- Order-by columns, if the search results are always to be sorted in a predefined order
- A location for the created index, if you do not want to store the index in the default directory. For different databases, use a different directory. This will avoid a potential naming problem if you want to use the same index name for indexes of different databases.

3. Calculate the amount of resources needed.

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

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

- The number of rows in the database 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

To calculate the maximum amount of shared memory required by an index, multiply the sum of the widths of the key column and the columns specified by the OPTIMIZE ON parameter, by the number of rows in the database table:

sum of the column widths * number of rows

Because 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.

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.

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.

Also note that other applications may be using system memory.

To achieve higher performance, data is retrieved from DB2 in blocks of 100 rows at a time. Note that if these rows contain large CLOBs, the memory required is 100 times the size of each row. For example, rows containing CLOBs of 4 MB, will have a memory requirement — just for the retrieval — of at least 400 MB.

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.

5. Enable the database.

At the command prompt, enter: `db2nx enable database` to prepare the database, which makes the stored procedure known to the database.

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

Create an index by entering the command:

```
db2nx enable text column table text-column index index-name using key-column
```

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)\ 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 DATABASE ASC
```

where `comment` is the column containing the text data to be indexed for later Net Search Extender search, `docid` is the primary key of table `db2nx.sample`

For more details about the parameters, see the “ENABLE TEXT COLUMN” on page 44 for further information.

Note

All keywords used on the command line, such as database name, 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 on.

Searching the 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 latter 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 search a database

Note

In this section we assume that the user is familiar with Net.Data.

In order to make use of the stored procedure within a Net.Data macro, you need to define a Net.Data function called search with the following parameter layout:

```
%FUNCTION(DTW_SQL) search (OUT INTEGER totalDocs,  
                           IN VARCHAR(250) 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>  
%}  
%}
```

On OS/390, 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:

On OS/390: 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)
```

Below is a functional description of the parameters used in more detail.

Note

Both types of search, with ranking and without are covered in this section. However, the only parameters that are different for each type of search are:

sqlStatement
RankOper

- totalDocs** An output variable that returns the total number of *hits* returned. *Hits* is the number of matches in the documents. You can have multiple matches per document or row.
- searchTerm** The word or words to be searched for. See “Chapter 6. Search syntax for the searchTerm parameter (SQLMM version)” on page 31.
- maxHitCount** The maximum number of hits to process. This is useful when one of the search terms has a high hit count and you want to improve performance by limiting the search result. Set to “0” or “” if you do not want the number of hits to be limited. If maxHitCount is less than the number of hits found, some of the search results will not be shown in your report. Note that the number of rows to be returned is set by maxRowsToReturn.
- maxIntermediateHitCount**
For searches with multiple search terms, the maximum hit count to be reached before retrieving hits for the next search term. This is useful when one of the search terms has a high hit count and you want to improve performance by limiting the result before it is joined with the result of the next search term. Set to “0” or “” if you do not want the number of intermediate hits to be limited. Use this parameter with caution: an intermediate result limited in this way can change the final search result report because the full intermediate queries may not be completed. This parameter is provided to help you improve performance, especially if the data is not well distributed, and if your system has limited memory resources for use during searching.
- 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.
- maxRowsToReturn**
The maximum number of rows to return beginning at startRow.
- indexname** The index to be used. This is the id-name 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 database instance must have write access to this directory. In Unix systems, the default value is /tmp. In 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

Using Net Search Extender

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 (%s) ORDER BY RSCORE DESC
```

RSCORE is the documents calculated rank score 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.

rankOper Specifies how to get the results 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 OPERATE_FUZZY:

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

2 OPERATE_STRICT:

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

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 come from DB2 according to the SQL statement.
- 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 Japanese" was the search term, and wordCounts contains "1000 210" after the search, this means "history" was found 1000 times, and "Japanese" was found 210 times. 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.

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 database

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 procedure can be identified:

- Line 85 and following: here you can modify the default parameters that are passed to the stored procedure.
- Line 171 and following: here you can see the call statement for the stored procedure. The parameters set in the previous section are passed on.
- Line 203 and following: 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 20.

Using Net Search Extender

The sample source file `NXSampleRank.java` has a similar structure, but only 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 DB2 log.

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`.

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.

Chapter 5. Administration overview

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

- Index monitoring
- Index maintenance
- Index and database backup and restoration

Most tasks require input from the application programmers who develop the database applications that search for and retrieve data.

For OS/390, enter the Net Search Extender commands at the UNIX system service 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.

At the end of this chapter there is a section on optimizing search performance. This section provides guidelines for both administrators and application developers using Net Search Extender to prepare data (that is, to create an index) and to implement search functions.

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 the table indexes created. It shows which table column belongs to that index.

```
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.
```

Administration overview

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 TITLE"
```

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 NPT 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.
```

Maintaining indexes

These are the administration tasks for maintaining existing indexes:

- Disable a text column to delete an index when an index is no longer needed
- Disable a database when all indexes are no longer needed
- **Updating an index**

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

To update an index, run:

```
db2nx "UPDATE INDEX yourIndex"
```

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.

- **Deleting an index**

To delete an index, enter:

```
db2nx "DISABLE TEXT COLUMN INDEX yourIndex"
```

- **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, there are two cases in which you must activate an index explicitly:

- When the index has been deactivated explicitly by the DEACTIVATE INDEX command
- When the system has been rebooted
- When Net Search Extender has been stopped by the UXSTOP command (Windows NT only). You have to reactivate the indexes using UXSTART.

To activate an index, run:

```
db2nx "ACTIVATE INDEX yourIndex"
```

To deactivate an index, run:

```
db2nx "DEACTIVATE INDEX yourIndex"
```

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 either index1 or index2, but never both indexes at the same time. In such cases, you should switch indexes by deactivating one index and activating the other.

See point 3 on page 18, "Calculate the amount of resources needed" for further information.

Backing up and restoring databases

After you have set up the DB2 database for Net Search Extender, it is advisable to do the following:

1. Back up the DB2 database using DB2 commands.
2. Copy the Net Search Extender index files from their index directory to a backup directory from where they can be restored.

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.

Retrieving the result table from memory can be much faster than retrieving it from the original table. "ENABLE TEXT COLUMN" on page 44 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 memory. See 3 on page 18, "Calculate the amount of resources needed" for further information.

- **Limit the search result**

Administration overview

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 20 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 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 24.

- **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.

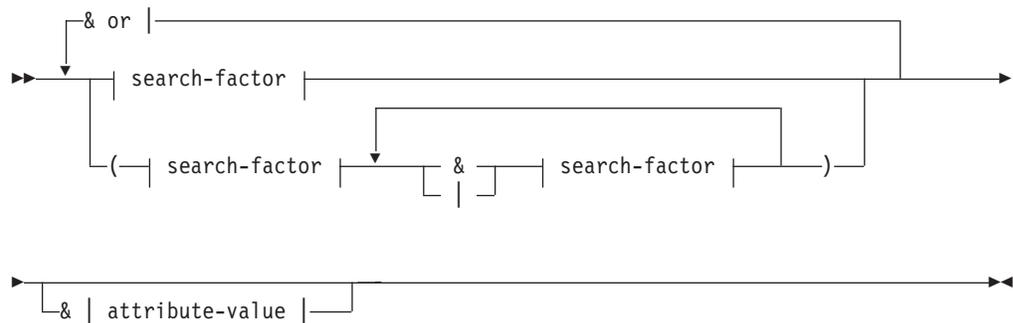
Part 2. Reference

Chapter 6. Search syntax for the searchTerm parameter (SQLMM version)

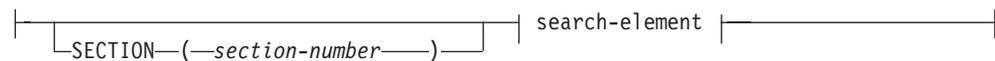
The search syntax described here is for the searchTerm parameter of the Net Search Extender stored-procedure call described in “Chapter 4. Using Net Search Extender” on page 17. This search syntax uses the public SQLMM standard.

Search argument

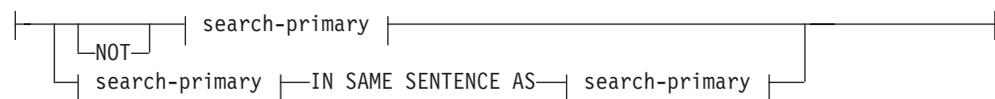
Command syntax



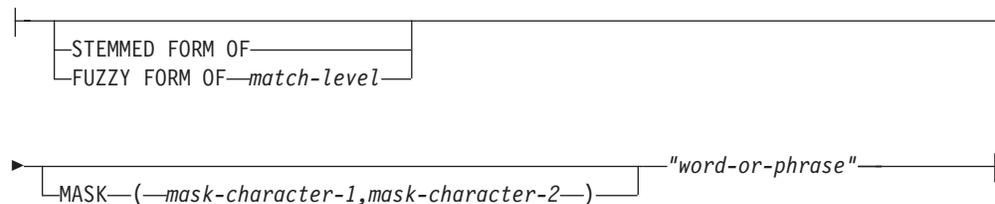
search-factor:



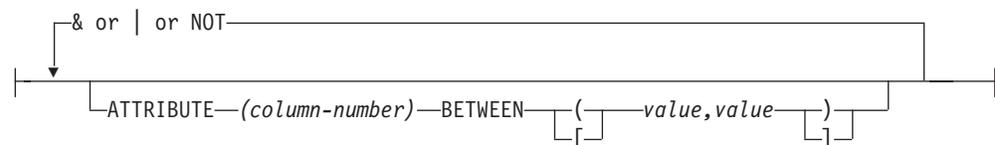
search-element:



search-primary:



attribute-value:



Command parameters

SECTION (*section-number*)

Search syntax (SQLMM)

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 related to 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 text documents from your search 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 form of search is not case-sensitive.

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.

“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.

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.

Search syntax (SQLMM)

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

MASK *mask-character-1,mask-character-2*

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.

attribute value

You can add numeric search terms that will limit the results of the query. However, 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 may be used between numeric search terms. They must be separated from the numeric search terms by at least one blank space.

The brackets define the left and right limits. 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**), parentheses are used.

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

The brackets can be used in any combination ([,], [,), and (,], for example). Note that ranking will only be 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 searches documents containing "book" and all the rows of column 1 containing the value 1234.

"book" & ATTRIBUTE (2) BETWEEN [MIN,20.00]

This searches documents containing "book" and all the rows of column 2 containing values within the range MIN and 20.00

Examples of query strings:

Search for documents containing emergency in section 10, and contain securitystioe (10) "◀;er
MM

Search syntax (SQLMM)

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	39
DEACTIVATE INDEX	Frees memory occupied by data associated with the index	40
DISABLE DATABASE	Deletes all indexes associated with a database; the database is no longer available for use by Net Search Extender	41
DISABLE TEXT COLUMN	Deactivates and deletes an index	42
ENABLE DATABASE	Prepares a database for use by Net Search Extender	43
ENABLE TEXT COLUMN	Prepares a text column for use by Net Search Extender and creates an individual text index for the column	44
GET INDEX STATUS	Displays the index settings and the status of an index	49
GET STATUS	Displays the status of a database	50
HELP	Displays help information	51
UPDATE INDEX	Updates and activates an index	52

This chapter is a reference for the Net Search Extender administration commands. While “Chapter 5. Administration overview” on page 25 provides an overview of Net Search Extender administration, this chapter includes information about the syntax of each command and describes the related parameters.

For OS/390, you enter the Net Search Extender commands at the UNIX system service prompt.

You prefix each command with db2nx except for HELP, NXICRT, NXIDROP, NXSTART and NXSTOP.

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

Environment variables

The environment variables set the default values for the administration commands.

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 database parameter of the Net Search Extender administration commands must match the location name information in the DSNAOINI file.

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

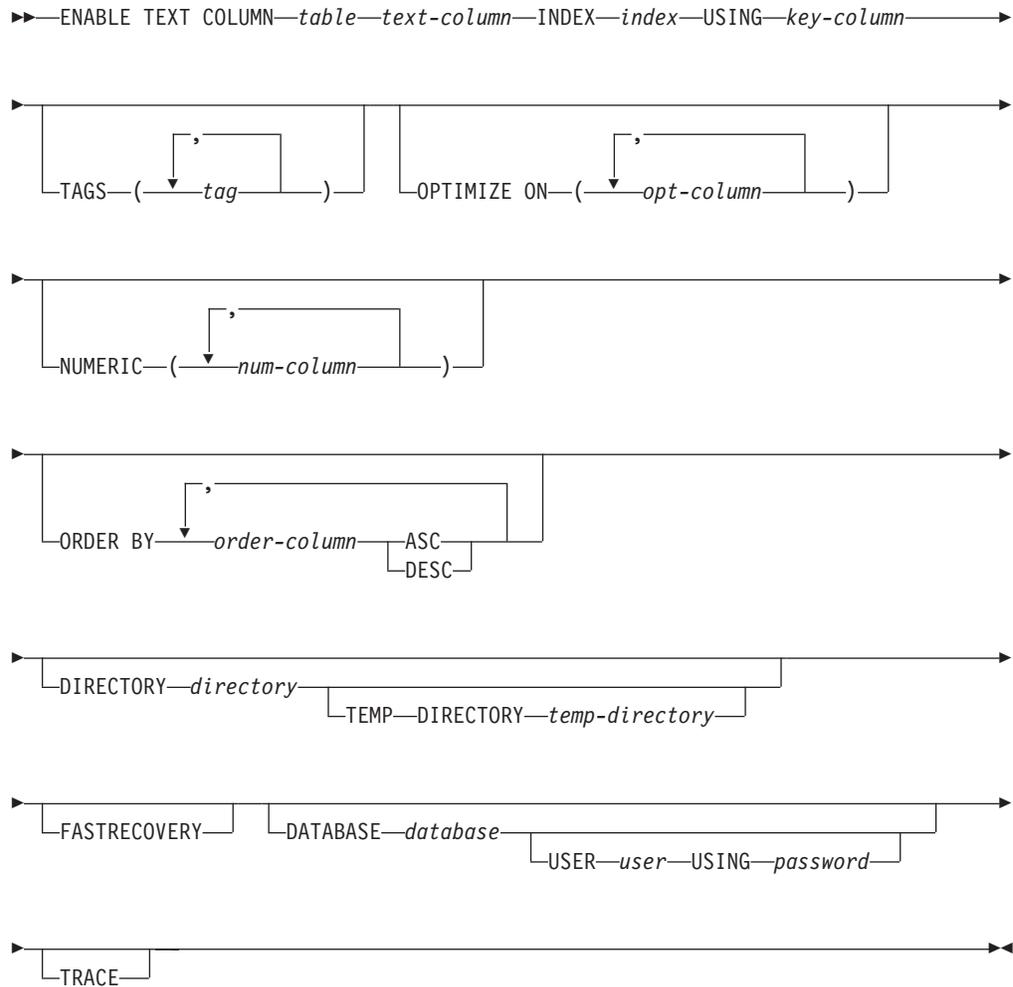
Tracing

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.

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 database that contains the column to be enabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

text-column

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

A name to be given to the index. The name must be unique in the specified directory and not longer than 8 characters.

USING key-column

The name of the key column used to get a relation between the indexed

ENABLE TEXT COLUMN

documents and the database rows. The key column must be of type INT, CHAR, VARCHAR, or TIMESTAMP, and it must be a unique key on the table. However, if you search only on the In-Memory Table, the key does not have to be unique.

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 3 on page 18, "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 this:

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

Note

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

Likewise, 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 for the `searchTerm` parameter (SQLMM version)" on page 31 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 22 columns to be held in an in-memory table of the database 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 database 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 3 on page 18, "Calculate the amount of resources needed" for further information.

ENABLE TEXT COLUMN

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

The numeric column `num-column` can be used to limit text search results to between a range, or equal to a value. For example, search on the word 'dog' but only return results between (19981201 < date < 20001201). Alternatively, search on 'history' but only return books with price < \$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 database 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, etc. numeric fields of the search term.

Also note that the JULIAN_DAY scalar function maybe 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

The columns used to specify a sequence during indexing so that documents can be retrieved from the index in 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

If order mask columns have NULL values, then according to SQL standards these will always rank highest in any given sort order.

DIRECTORY directory

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

TEMP DIRECTORY directory

The directory where temporary index files are to be stored. In Unix systems, if you don't specify a directory, the default directory `/tmp` is used. If you don't specify a directory in Windows NT, the index directory is used.

FASTRECOVERY

Use this 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 point 3 on page 18, "Calculate the amount of resources needed" for further information. Also note that indexing and search performance is slower when this keyword is set.

DATABASE database

The name of the database you want to work with. The `DSNA0INI` environment variable is always used.

USER user

The user ID of the DB2 instance with `DBADM` authority for the database that contains the table.

USING password

The password for the DB2 instance.

TRACE

Activates tracing.

ENABLE TEXT COLUMN

Note

The ENABLE TEXT COLUMN command writes progress information to stdout indicating the amount of data indexed with time stamps to track performance information. Sample output looks like the following:

```
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

```

▶▶—GET INDEX STATUS—index—————▶
|
| DATABASE—database—|
| | USER—user—USING—password—| TRACE—|
| | | | | | | | | | | | | | | | |

```

Command parameters

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

DATABASE *database*

The name of the database 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 database you want to work with.

USING *password*

The password for the DB2 instance.

TRACE

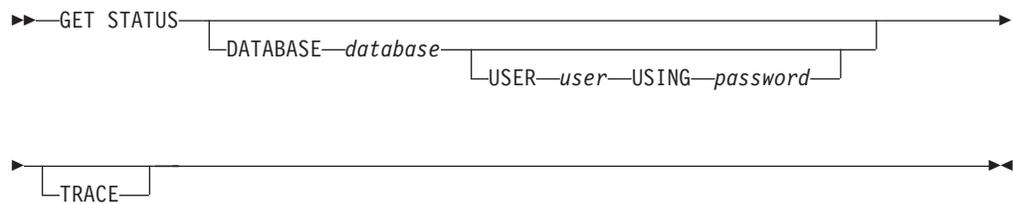
Activates tracing.

GET STATUS

GET STATUS

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

Command syntax



Command parameters

database

The name of the database 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 database for which you are getting the status.

USING password

The password for the DB2 instance.

TRACE

Activates tracing.

HELP

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

Command syntax

▶—HELP—▶

Command parameters

None.

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

```
db2nx db2nx-command
```

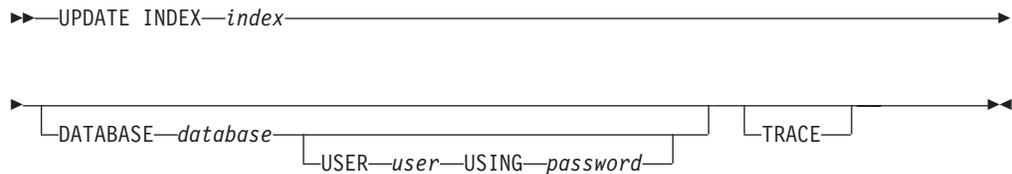
```
List of db2nx commands
```

```
HELP
ENABLE DATABASE      [database [USER user USING password]]
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}} ... ]]
                   [FASTRECOVERY]
                   [DIRECTORY directory [TEMP DIRECTORY temp-directory ]]
                   [DATABASE database [USER user USING password]]
DISABLE TEXT COLUMN INDEX index [DATABASE database [USER user USING password]]
UPDATE INDEX        index [DATABASE database [USER user USING password]]
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]]
```

UPDATE INDEX

This command starts indexing using the same settings defined during enable text column. It brings the index and the in-memory table up to date to reflect the current contents of the text column with which the index is associated.

Command syntax



Command parameters

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

DATABASE database
The name of the database 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 database you want to work with.

USING password
The password for the DB2 instance.

TRACE
Activates tracing.

Note

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

Usage notes

Because updating an index makes it unavailable for search, your application should not allow 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.

Chapter 8. Messages

Each message has a message identifier that consists of a prefix (DES), the message number, and a suffix letter. The suffix letter indicates how serious the occurrence is that produced the message:

- I Information message
- W Warning message
- N Error (or “negative”) message
- C Critical error message

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: 'nmn'.

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 | 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: 'nmn'.

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.

Messages

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: No storage could be reserved for the application.

What to do: Increase the paging space.

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

Explanation: No shard 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. If this doesn't help, contact your IBM representative.

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. If this doesn't help, contact your IBM representative.

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

Explanation: An unexpected token was found.

What to do: Check the command syntax.

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 Index *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.

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 43.

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.

DES7220N 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.

Messages

DES7230W DB2 Net Search Extender will expire in %d1 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

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 Parser syntax error on 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 representative.

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* is 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.

Messages

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.

DES7800I The command completed successfully.

Explanation: The specified command completed successfully.

What to do: No action required.

DES7801N The command failed.

Explanation: The command was not executed successfully.

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 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.

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
J74/G4
555 Bailey Avenue
P.O. Box 49023

San Jose, CA 95161-9023
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:

AIX	DB2 Universal Database	OS/390
DB2	IBM	z/OS
DB2 Extenders	Net.Data	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.

Intel is a registered trademark of Intel.

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

Index

A

ACTIVATE INDEX command
 syntax 39
activating an index 26
administration
 ACTIVATE INDEX command 39
 command line processor 37
 command summary 37
 DB2NX command 37
 DEACTIVATE INDEX command 40
 DISABLE DATABASE command 41
 DISABLE TEXT COLUMN
 command 42
 ENABLE DATABASE command 43
 ENABLE TEXT COLUMN
 command 44
 GET INDEX STATUS command 49
 GET STATUS command 50
 HELP command 51
 sample program 11
 UPDATE INDEX command 52
administration overview 17, 25
attributes 34

B

backing up databases 27
Boolean search argument 33

C

CCSIDs 17
code pages 17
command line
 DB2NX command 37
commands
 ACTIVATE INDEX 39
 DB2NX 37
 DEACTIVATE INDEX 40
 DISABLE DATABASE 41
 DISABLE TEXT COLUMN 42
 ENABLE DATABASE 43
 ENABLE TEXT COLUMN 44
 GET INDEX STATUS 49
 GET STATUS 50
 HELP 51
 summary 37
 UPDATE INDEX 52
compatibility 5
creating an index
 ENABLE TEXT COLUMN
 command 44
 how to 17
Customization
 DB2 related tasks 8
 OS/390 5
 Workload Manager tasks 7

D

database
 backing up and restoring 27

database (*continued*)
 DISABLE DATABASE command 41
 displaying status 25
 ENABLE DATABASE command 43
 enabling 17
 GET STATUS command 50
dataSource 22
DB2 command line processor
 syntax 37
DEACTIVATE INDEX command
 syntax 40
deactivating an index 26
deleting an index 26
DISABLE DATABASE command
 syntax 41
DISABLE TEXT COLUMN command
 syntax 42
disabling an index 26

E

ENABLE DATABASE command
 syntax 43
ENABLE TEXT COLUMN command
 syntax 44
enabled status of databases
 displaying 25
 GET STATUS command 50
error log 24

F

Fastrecovery 47
features 3
FUZZY FORM OF keyword 33

G

GET INDEX STATUS command
 syntax 49
GET STATUS command
 syntax 50

H

hardware requirements 5
HELP command
 syntax 51

I

IN SAME SENTENCE AS keyword 33
index
 ACTIVATE INDEX command 39
 activating 26
 DEACTIVATE INDEX command 40
 deactivating 26
 deleting 26
 DISABLE TEXT COLUMN
 command 42

index (*continued*)

 disabling 26
 displaying the status 26
 GET INDEX STATUS command 49
 HELP command 51
 maintaining 26
 optimizing 27
 UPDATE INDEX command 52
 updating 26
indexDirectory 21
indexname 21

J

Java sample program
 Creating a search function 23
 running 13

M

MASK keyword 34
masking characters 33
match level 33
maxHitCount 21
maxIntermediateHitCount 21
maxRowsToReturn 21
memory requirements 5
messages 53
Monitoring the environment 12

N

Net.Data sample program
 Creating a search function 20
 running 13
 Search function example 23
 Using the search function 20
NOT keyword 33
Notices 61
NXSample.java sample program
 running 13
nxsample sample program
 running 11

O

optimizing search performance 27
outTable 23

P

performance of searches 27

R

rankoper 22
requirements 5
restoring databases 27

S

sample programs
 administration 11

- sample programs (*continued*)
 - Java 13
 - Net.Data 13
 - nxsample 11
 - nxsample.d2w 13
 - NXSample.java 13
 - running 11
- search performance 27
- search syntax 32
- searching in a database
 - using Net.Data 17
- Searching the index 19
- searchTerm 21
- searchTerm parameter syntax
 - SQLMM 31
- SECTION keyword 32
- shared memory considerations 26
- shared memory status 26
- software requirements 5
- sqlStatement 21
- startRow 21
- status of database
 - displaying 25
- status of index
 - displaying 26
 - GET INDEX STATUS command 49
- STEMMED FORM OF keyword 33
- stored procedure
 - calling from a Java program 13
 - for searching using Net.Data 17
 - for standard search 14
 - for standard search with ranking 15
- stored procedure with ranking
 - OS/390 signature 15
- stored procedure without ranking
 - OS/390 signature 14
- Stored procedures for searching 14
- system requirements 5

T

- tag number 32
- textSearch
 - calling from a Java program 13
 - for searching using Net.Data 17
 - solving problems 24
- tmpDirectory 21
- totalDocs 21
- tracing 24
- txfpnetd.d2w sample program
 - running 13

U

- UPDATE INDEX command
 - syntax 52
- updating an index 26

V

- version compatibility 5

W

- wordCounts 23

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

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

Publication No. SC27-1171-00

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

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.



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



Program Number: 5675-DB2

SC27-1171-00

