IBM DB2 10.5
for Linux, UNIX, and Windows

Net Search Extender Administration and User's Guide

IBM

SC27-5526-00

IBM DB2 10.5
for Linux, UNIX, and Windows

Net Search Extender Administration and User's Guide

IBM

SC27-5526-00

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 305.

Edition Notice

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at http://www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide/

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

Contents

Part 1. Net Search Extender overview and concepts

Chapter 1. Net Search Extender key concepts

Net Search Extender offers users and application programmers a way to search full-text documents stored in DB2® databases, other databases, and file systems using SQL queries.

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

To fully understand the capabilities of Net Search Extender, it is necessary to understand key terms and the various options available. It is also necessary to have a basic understanding of DB2 database concepts and terms.

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

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

The documents can be stored in various formats. The formats include either unstructured plain text, structured text such as HTML or XML, or proprietary document formats such as PDF or Microsoft Office document formats. For the latter, additional filtering software that might have to be licensed separately is required.

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

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

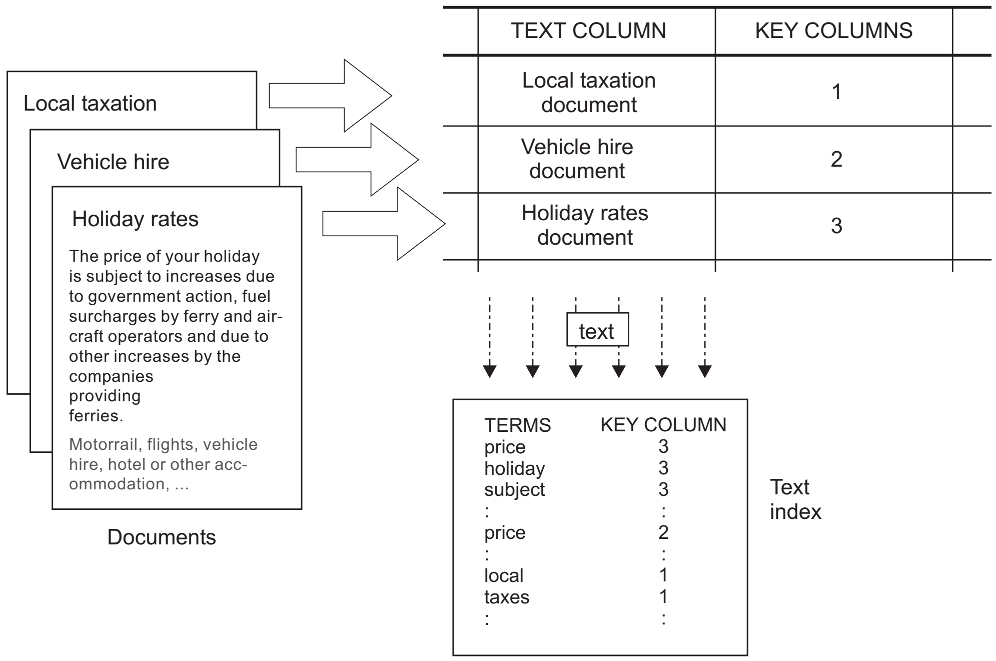

TEXT COLUMN
KEY COLUMNS
Local taxation document
1
Vehicle hire document
2
Holiday rates document
3
Local taxation
Vehicle hire
Holiday rates
The price of your holiday is subject to increases due to government action, fuel surcharges by ferry and air-craft operators and due to other increases by the companies providing ferries.
Motorrail, flights, vehicle hire, hotel or other acc-ommodation, ...
Documents
text
TERMS KEY COLUMN
price 3
holiday 3
subject 3
price 2
local 1
taxes 1
Text index

Figure 1. Creating a text index

Text index creation is the process of defining and declaring the properties of the index, such as the location of the index. After creation, the text index does not as yet contain any data. *Index update* is the process of adding data about terms and documents to the text index. The first index update adds information about all text documents from the text column to the index. The first update is known as the *initial update*.

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

Net Search Extender supports two options to synchronize the text index with its source table. The basic synchronization in Net Search Extender is based on triggers that automatically store information about new, changed, and deleted documents in a log table. There is one log table for each text index.

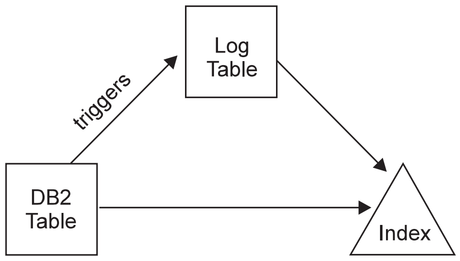


Figure 2. Incremental update process with triggers

The basic option is based only on triggers; updates that are not recognized by triggers will therefore be ignored, for example, loading data with the **LOAD** command, attaching or detaching ranges for a range-partitioned table. An extended synchronization option enables capturing such changes through integrity processing by adding a text-maintained staging table to store information about new and deleted documents, while the log table stores information about changed documents captured through a trigger.

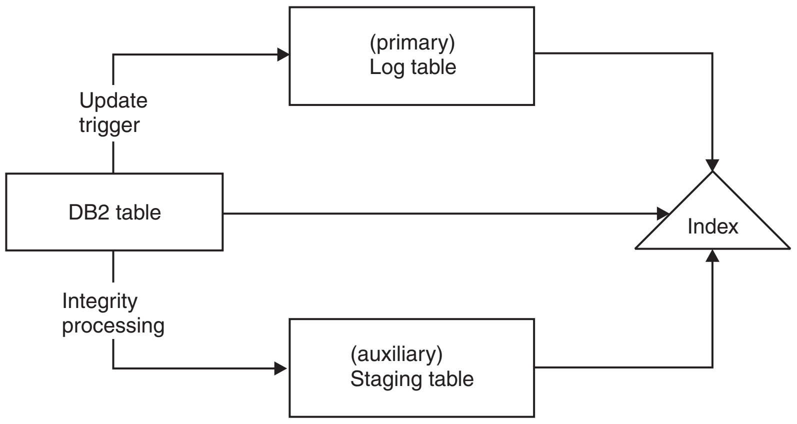


Figure 3. Incremental update with triggers and integrity processing

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

Note that neither of these options synchronizes the text index within the scope of a transaction that updates, deletes, and inserts text documents. Net Search Extender's asynchronous text indexing improves performance and concurrency. The update is applied within a separate transaction to a copy of a very small part of the index. The index is only locked for read access during a very short period of time when the copy is put in place of the original. This is invisible to search operations, see Chapter 22, "Net Search Extender instance services," on page 77 for information.

A text index has certain properties, such as index file location and automatic update properties. If necessary, you can change some of the properties. This is known as *altering* the index. Altering the index does not modify any index data.

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

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

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

Usually the first update after creating a text index is an initial update, and the following updates are incremental. However, when working with presorted indexes you want to keep the order in case of updates. This is addressed by the RECREATE INDEX ON UPDATE option, which totally rebuilds the index each time an update is performed.

After the text index is updated, you can search using one of the following options:
- An SQL scalar search function
- A stored procedure search
- An SQL table-valued function

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

SQL scalar search function overview

Net Search Extender offers three scalar text search functions (CONTAINS, NUMBEROFMATCHES, and SCORE) that are seamlessly integrated within SQL.

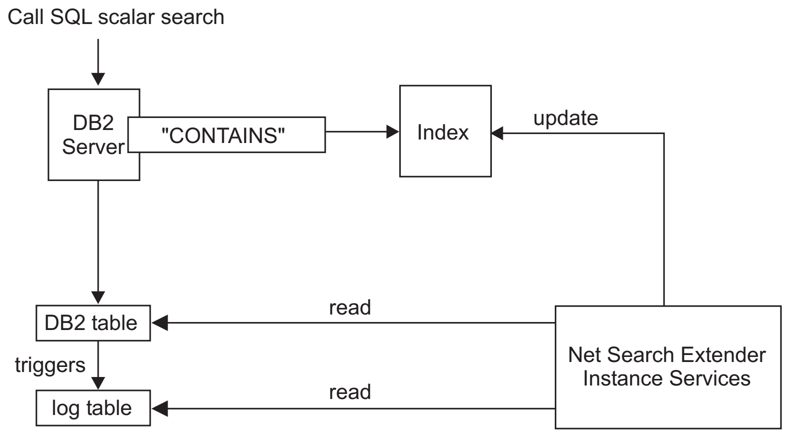


Figure 4. Using an SQL scalar search function for searching

You can use the search functions in the same places that you would use standard SQL expressions within SQL queries. Typical queries are:

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

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

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

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

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

Stored procedure search overview

The main use of the stored procedure search is in high performance and high scalability applications that are interested in text-search-only queries, that is, in queries that do not need to join text search results with the results of other complex SQL conditions.

Usually, presenting search results to the end user involves a call to the search function itself that is followed by a join operation against the user table and possibly a sorting of the result data. This can be a costly operation. However, there are situations in which an application can avoid costly disk operations by performing the join operations on presorted data that is stored in memory. These situations include:

- The subset of data to present to the user is small
- The subset of data is known in advance
- The intended sort order is fixed and known in advance
- A ranked subset of the search results is sufficient

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

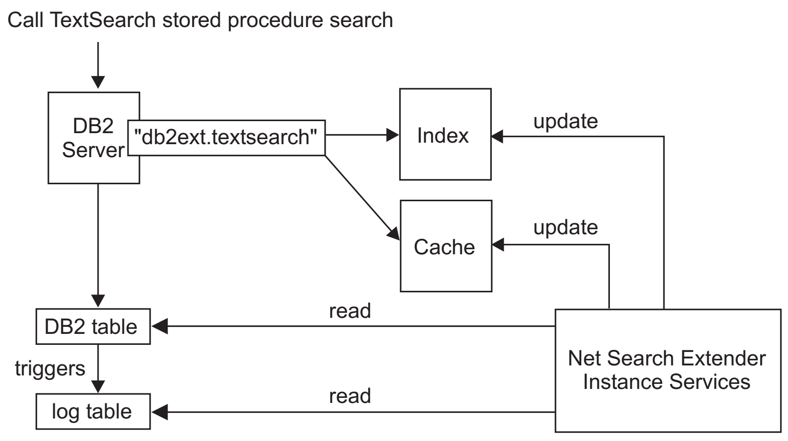


Figure 5. Using a stored procedure search

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

The decision to use the stored procedure for search requires careful memory calculations, such as how much memory is required and how much free memory should be left for index updates. For defaults, see Chapter 15, "Stored procedure search memory requirements," on page 61.

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

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

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

SQL table-valued function overview

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

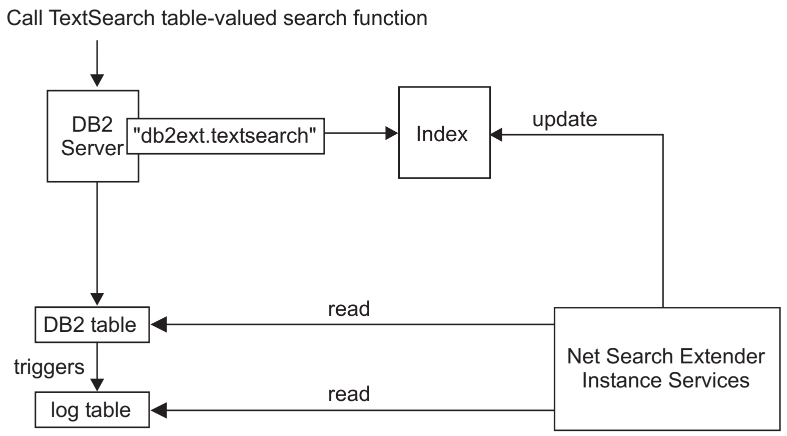

Call TextSearch table-valued search function
DB2 Server
"db2ext.textsearch"
Index
update
DB2 table
read
triggers
log table
read
Net Search Extender Instance Services

Figure 6. Using an SQL table-valued function for searching

The main functional differences to the stored procedure search are:

- No cache is necessary (and no cache is exploited).
- The table-valued function can be used in arbitrary SQL statements.
- Large amounts of memory are not required to pre-store cache table content.

The main functional difference to the SQL scalar search functions is:

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

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

Additional concepts

Column transformation function

You can use your own function to convert an unsupported format or data type into a supported format or data type. By specifying a User Defined Function (UDF), you can get the original text document as input.

The output from the UDF must be a supported format that can be processed during indexing.

You can also use this feature for indexing documents that are stored on external data stores that are not directly supported. In this case, the DB2 column contains document references and the function returns the document contents that have the relevant document reference.

DB2 Net Search Extender instance services

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

DB2 Net Search Extender instance services on Windows

When a new DB2 instance is created, the DB2EXT service for the first partition is automatically created.

DB2EXT services will be added subsequently when the **`db2ncrt`** command is used to add new partitions. DB2EXT services will also be added when the **`db2start add dbpartitionnum`** command is executed. Similarly, the **`db2ndrop`** and **`db2nchg`** also drop or modify DB2EXT services appropriately.

Externally stored data

Text documents that are stored externally, such as in other databases, are supported. For documents that are stored on other databases, use DB2 nickname tables to create a text index.

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

You can also use the column transformation function for data that are stored in unsupported external data stores.

Administration tables and views

The Net Search Extender tables and views provide text index and property information.

There are several tables and views available in Net Search Extender.

Partitioned database support

NetSearch Extender functionality might be affected by additional factors when deployed in a partitioned database environment.

When enabling and administering NSE in a partitioned database environment, consider the following factors:

- Ensure that the DB2 setup is complete as described in the DB2 documentation. The NFS mount must be configured with root access and setuid.
- If a problem occurs during **`db2text start`**, no detailed message is returned stating which of the available partitions is affected. If you issue **`db2text start`** a second time, the system tries to start the service on each of the partitions. The **`db2text start`** command is successful if the following message is displayed: CTE0185 The update and locking services are already active.
- The fenced user ID should be the same as the instance owner ID .
- You cannot insert a new partition number or delete an existing one from `db2nodes.cfg` while the NSE Instance Services are running. This applies to any command that might result in changes to `db2nodes.cfg`.

- On Windows platforms, while using NSE with partitioned database environment, `db2nodes.cfg` should not use IP addresses as well as host names for the same host.
- To avoid unexpected results and error messages, drop existing text indexes before performing data redistribution. If the redistribution operation is executed without first dropping the text indexes in that database, they can still be dropped after the redistribute operation completes.

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

- The stored procedure search and the SQL table-valued function can operate only on tables that are local to the coordinator partition on a partitioned database.
- The SQL scalar search functions (CONTAINS, NUMBEROFMATCHES, and SCORE) can be used on tables spanning multiple partitions except in rare cases.

You should be aware of the following considerations when conducting searches in a partitioned database environment:

- The RESULT LIMIT is evaluated on every partition during search. This means that if you specify a RESULT LIMIT of 3 and use 4 partitions, you might get up to 12 results.
- The SCORE value reflects the document's relevance when compared to the SCORE value of all documents from a single partition even if the query accesses multiple partitions.

Indexes on nicknames in a federated database

You can create a text index on nicknames in a federated database that points to tables in a remote database.

In this case, the role of the log table (for incremental index updates) differs from its role for an index on a regular table. Unlike regular tables, DB2 triggers can not be created on nicknames, so that change information about documents cannot be inserted into a log table using triggers, nor can the extended text-maintained staging infrastructure be used to capture changes. Therefore, there are two different ways for incremental updates to create an index on a nickname:

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

Native XML support

By fully supporting the SQL XML data type in databases with code page UTF-8, all Net Search Extender search functions can be used on XML documents that are stored natively in the database. Note that the text search on SQL XML data types in databases with non-UTF-8 is not supported and might not return any results.

The structural text search on XML documents by sections (see “Search parameters” on page 248 for more information about searching in **sections**) can be extended by powerful XQuery processing of the search results. Net Search Extender text search functionality can be leveraged using the DB2 database server's XQuery language support to provide optimal processing of XML documents.

By using full text search within the `db2-fn:sqlquery()` XQuery input function, it is possible to search in XML documents, and process the resulting XML documents using XQuery:

```
FOR $dept in db2-fn:sqlquery('select Department from MyTable
    where contains(Department,''sections(/dept/employee/resume) "DB2 XML" '')
      = 1')/dept
RETURN $dept/employee/name
```

In the example, column "Department" is of data type "XML". See Part 12, “Working with structured documents,” on page 171 for more information.

A sample is available that shows you how you can query XML data. Refer to *sqllib*/samples/extenders/db2ext. Call xmlsample *database* to populate the database, and create and update the indexes. After you have connected to the database, you can perform searches on the data by issuing **db2 -tvf xmlsearch**.

Partitioned table support

You can create a text index for range-partitioned tables or tables that use the multidimensional clustering feature in single-partition and multi-partition database environments.

Text indexes are supported for any partitioning feature combination.

Note: For Version 9.7, the text index will be partitioned according to the partitioning of the table across multiple database partitions. Other partitioning features, like range-partitioning or multidimensional clustering, have no impact on the partitioning of the text index

Incremental update based on integrity processing

By using the **AUXLOG** option for the Net Search Extender **CREATE INDEX** command, you can control whether an auxiliary log infrastructure (a text-maintained staging table) is used for a text index.

This auxiliary staging table captures information about new and deleted document through integrity processing, while document updates are captured through an update trigger on the base table column referenced in the **INDEX CREATE** command.

The following restrictions apply to using the option:
- The object for which you created the text index must be a base table, not a view or a nickname.
- You cannot specify the **CACHE** option
- You cannot use the **RECREATE INDEX ON UPDATE** option.
- You cannot use replication to control updates.

By default this configuration option is set to ON for range-partitioned tables and the configuration option is set to OFF for non-partitioned tables.

Capturing changes for an incremental update of the text index through integrity processing might require you to perform additional administrative tasks after performing a database operation on the base table. You might have to perform post-processing tasks for the command or preprocessing tasks for a text index update to set integrity for the base table or its dependent tables.

```
db2 "create table test.simple (pk integer not null primary key,
comment varchar(48))"
```

```
db2 "insert into test.simple values (1, 'blue and red')"
```

```
db2text "create index test.simpleix for text on test.simple(comment) index
configuration(auxlog on) connect to mydb"
```

```
db2text "update index test.simpleix for text connect to mydb"
```

```
db2 "load from loaddata4.sql of del insert into test.simple"
```

After the load operation, the base table is blocked. For example, a select operation will result in SQL0668N Operation not allowed for reason code "1" on table "TEST.SIMPLE". SQLSTATE=57016.

The staging table is accessible, but does not yet contain the information about the changed data.

```
db2 "set integrity for test.simple immediate checked"
```

returns SQL3601W The statement caused one or more tables to automatically be placed in the Set Integrity Pending state.SQLSTATE=01586.

At this point the staging table is blocked and modifying operations for the base table are rejected.

```
"insert into test.simple values(15, 'green')"
```

returns DB21034E The command was processed as an SQL statement because it was not a valid command line processor command. During SQL processing it returned: SQL0668N Operation not allowed for reason code "1" on table "SYSIBMTS" ."SYSTSAUXLOG_IX114555". SQLSTATE=57016.

```
db2text "reset pending for table test.simple for text connect to mydb"
```

```
db2text "update index test.simpleix for text connect to mydb"
```

For more details see Part 7, “Planning considerations,” on page 55

Separate fenced user support (Linux and AIX)

Net search extender requires a fenced user account for running its user-defined functions (UDFs) and stored procedures outside of the address (memory) space used by the DB2 database server.

A separate fenced user ID is now supported for Net Search Extender. This ID is different from the DB2 instance owner. A common group is added as secondary group for both the instance owner and fenced user.

You should be aware of the following considerations when using the fenced user ID:

- If the instance owner and fenced user are different, the Net Search Extender admin commands will fail when run by a table owner or a user with control privilege on the table. For example, if the instance owner and fenced user are different and do not share a common secondary group, **db2text START** will return the following error message:

  ```
  CTE0312E No common secondary group exists for fenced user and instance owner
  ```

  A common primary group can be used but a common secondary group is suggested for security reasons. The system, root or admin group should not be used as a common secondary group due to the security risk of NSE files ownership by that group. A new secondary group should be created for this purpose.

Note: NSE services will not start if the common secondary group of the instance owner and fenced user is set to zero.

- A fenced user should have access to index files and thesaurus files.

  Note: The new secondary group helps achieve this. Make sure the umask restrictions allow for group read and write access for the fenced user. The umask value should be set to 0002.
- The following administrative commands can only be issued by the instance owner:
  - CREATE INDEX
  - UPDATE INDEX
  - ALTER INDEX
  - DROP INDEX
  - ACTIVATE CACHE
  - DEACTIVATE CACHE
  - RESET PENDING
  - CLEAR EVENTS
  - DB2EXTTH
  - HELP
  - COPYRIGHT
- No external files are present in the NSE index directory other than NSE index files.
- On HPUX, the thesaurus definition file needs to be recompiled for thesaurus searches to work.
- The **`NUMBER_DOCS`** and **`REORG_SUGGESTED`** routines will not be supported in a partitioned database environment.

Example

If instance owner's ID is db2inst1 and fenced user's ID is db2fenc1, create a new group called 'video'. Make this group the secondary group for both the instance owner and the fenced user.

It can be verified with the following steps:

```
>id db2inst1
uid=44049(db2inst1) gid=204(search) groups=33(video)


>id db2fenc1
uid=44048(db2fenc1) gid=100(users) groups=33(video)
```

Chapter 2. Key features of DB2 Net Search Extender

Net Search Extender Version 9.7 has key features which involve indexing, searching and search results.

- Indexing
  - Fast indexing of very large data volumes
  - Dynamic updating of indexes
  - Optional: Storing table columns in main memory at indexing time to avoid expensive physical read operations at search time
  - Support for structured text formats, for example HTML and XML
  - Support of third-party filtering software "Outside In"
  - Nickname table support
  - Support of presorted text indexes
  - Partitioned database support
  - Native XML support
  - Support for range-partitioned tables and clustered tables (MDC)
- Search
  - Boolean operations
  - Proximity search for words in the same sentence or paragraph
  - "Fuzzy" searches for words having a similar spelling as the search term
  - Wildcard searches, using front, middle, and end masking, for whole words and single characters
  - Free-text searches. For documents containing specific text, the search argument is expressed in natural language
  - A highlight function to show why a particular document qualified as a search result
  - Thesaurus support
  - Restrict searching to sections within documents
  - Numeric attribute support
  - High-speed searching through a large number of text documents with many concurrent users
  - Integration into XQuery processing using the `db2-fn:sqlquery()` function.
- Search results
  - You can specify how the search results are sorted at indexing time
  - You can specify search result subsets when searching large data volumes and expecting large result lists
  - You can set a limit on search terms with a high hit count
  - Built-in SQL functionality combined with the DB2 Optimizer automatically selects the best plan according to the expected search results

Restriction:

- Net Search Extender does not support the IBM® DB2 pureScale® Feature.
- Net Search Extender does not support default table organization by column.

Chapter 3. Introducing the db2text commands

You can access the NetSearch Extender functionality using the **db2text** command.

About this task

Here is an example of a Net Search Extender command:

```
db2text ENABLE DATABASE FOR TEXT
```

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

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

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

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

```
db2text ?
```

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

```
db2text ? command
```

Example

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

```
db2text ? CREATE INDEX
```

db2text returns 0 if the command has been processed successfully, and 1 if the command has not been processed. Note that if there are document errors but the index still updates, the **db2text** command returns 0 with a warning message. Information about document errors can be found in the event table of the index.

What to do next

Depending on your operating system and your active command shell, the system interprets special characters such as ?, (,), *, !, and ". Therefore, if the command contains these characters, use quotation marks or an escape character.

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

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

Part 2. Installation

Chapter 4. Installing Net Search Extender in the DB2 client/server environment

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

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

If you plan to administer Net Search Extender from a remote client, you have to have one of the DB2 server editions and Net Search Extender itself installed on the client side.

Chapter 5. Net Search Extender installation system requirements

The minimum hardware and software requirements for Net Search Extender-supported platforms for DB2 Version 9.7.

You need to install DB2 Version 9.7 before you can run Net Search Extender.

Net Search Extender is supported on the following platforms:
- AIX® (64-bit) platforms
- Linux x86 (32-bit) platforms
- Linux x64 (64-bit) platforms
- Linux on zSeries® (64-bit) platforms
- HP-UX on Itanium-based HP Integrity Series (64-bit) platforms
- Solaris UltraSPARC (64-bit) platforms
- Windows on x86 (32-bit) platforms
- Windows on x86 (64-bit) platforms

Note: Partitioned database environment is not supported by Net Search Extender for (32-bit) platforms.

The minimum disk space for a typical Net Search Extender installation is 50 MB. Additional hardware requirements might be required depending on the amount of data you are planning to index.

Chapter 6. Installing Net Search Extender on a partitioned DB2 server

After installing DB2, you must install Net Search Extender on each partition.

About this task

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

You must ensure that NSE is correctly installed and configured on every partition.

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

Chapter 7. Installing Net Search Extender on UNIX

To install Net Search Extender on UNIX, install the product and update the DB2 instance.

About this task

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

Procedure

To install Net Search Extender on UNIX, follow these steps:

1. Install the product components.
   a. Log on at the target machine as the root user.
   b. Change to the correct directory for your platform.
      - `cd /`*`cdrom`* where *cdrom* is your CD-ROM driver path.
      - `cd` *`platform`*
   c. Uninstall any older version of NSE, if present .
   d. Call **`./nsesetup.sh`** and follow the instructions displayed on the screen.

      After you accept the license agreement, you are presented with a list of possible installation paths. The installation paths that are eligible for you to use depend on the Net Search Extender version you want to install and the installed copies of DB2 database products. After you select a path for the installation, the product is installed at this path. An installation log file is written to the `/tmp` directory and prefixed `db2nsei`.
2. Update the DB2 instance.
   a. Ensure that you are active as the root user.
   b. Use the following command to change your working directory to the path where you installed Net Search Extender:

      ```
      cd path/instance
      ```

      path is the path of the DB2 copy where you installed Net Search Extender.
   c. Run **`db2iupdt`**. Use `./db2iupdt` *`db2instance`*, where *db2instance* is the name of an existing DB2 instance user ID that you would like to use with Net Search Extender.

Chapter 8. Installation on Windows

This method uses some command-line options. The command-line options that require a parameter must be specified with no space between the option and its parameter.

About this task

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

Procedure

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

1. Use the `*cdrom*:\windows\install\setup.exe` to transfer the files from the package to the target machine.

   Note that for every DB2 instance, you must enter a user ID and password to create the correct Net Search Extender services.

   After you accept the license agreement, you are presented with a list of possible DB2 copy names. The DB2 copy names that are eligible for you to use depend on the Net Search Extender version you want to install. After you select a DB2 copy name, the product is installed at the path where this DB2 copy has been already installed.

   For silent install, you can call `setup.exe` in two different modes:

   RECORDMODE
   : Creates a silent install response file.

     A predefined response file called `setup.iss` is located in the installation source directory (this is not used in the case of silent installation). If you want to create a new response file, run `setup.exe -r`. The new `setup.iss` is created in your Windows directory. Copy this `setup.iss` into your installation source directory. Make sure that you have made a backup of your old response file.

     Example: `setup.exe -r -f1"d:\some_directory\setup.iss"`

   SILENTMODE
   : Silent installation.

     Make sure that the file `setup.iss` is located in your installation source directory. Run `setup.exe -s`. The installation is successful if `ResponseResult` is set to `0` in the `setup.log` file located in your installation source directory.

     Example: `setup.exe -s -f1"d:\some_directory\setup.iss" -f2"d:\another_directory\mysetup.log"`

   Installation parameters:

   - **`/r`**: Record mode (records a response file for silent installations)
   - **`/s`**: Silent install (runs installation silently)

- **`/f1`**: Specify alternative response file name (full path)
- **`/f2`**: Specify alternative setup log file name (full path)

2. Reboot the system after the installation.
3. Call `db2text start` to start the DB2 Net Search Extender Instance Services.

Results

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

For a partitioned instance, a set of Windows services are created, one per partition. The DB2EXT instance services for every partition must run under the same user account as the DB2 instance services.

Net Search Extender does not support Microsoft Cluster Server.

Chapter 9. Net Search Extender directory names and file names

Net Search Extender commands require specified names.

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

Chapter 10. Installing the Outside In libraries

To use Net Search Extender with the Outside In software from Stellent, you must set up the required libraries on your platform.

About this task

The Outside In software from Stellent is available for several platforms. For details, see http://www.oracle.com.

Procedure

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

- On Windows, ensure that the directory where the libraries are located is added to the path environment variable.
- On UNIX, add the Outside In libraries into the DB2 `lib` install directory.

Chapter 11. Installation verification

Installation verification on UNIX

You must verify that Net Search Extender is correctly installed to prevent incorrect query results and errors.

Procedure

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

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

Results

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

In a partitioned database, use the following verification sample:

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

Installation verification on Windows

You should verify that your copy of NetSearch Extender is properly installed to prevent some maintenance issues.

Procedure

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

- Non partitioned environment:
  1. Set up the sample text indexes as follows:
     a. Call **`db2cmd`** to open a DB2 command window.
     b. Change to `sqllib\samples\extenders\db2ext`
     c. From the DB2 command window, call `nsesample.bat` *`yourdb`*. Note that this command creates the database if it does not already exist.
     d. Check the generated output file `nsesample.log` in the current directory.
  2. Call the following sample queries to execute in the DB2 command window:
     a. Connect to your database using `db2 connect to` *`yourdb`*

   b. Execute the sample queries using `db2 -tvf search`
   c. Check the results of the queries contained in the script. Note that every query should return one or more hits.

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

- Partitioned database environment: Set up the sample text indexes as follows:
  1. Call `db2cmd` to open a DB2 command window.
  2. Change to `<sqllib>\samples\extenders\db2ext`
  3. From the DB2 command window, call `nsesample_partitioned.bat` *<yourdb>*.
  4. Check the generated output file `nsesample_partitioned.log` in the current directory.

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

Chapter 12. Uninstalling Net Search Extender

To permanently remove Net Search Extender from your system and remove all Net Search Extender indexes, you must first disable each database that contains Net Search Extender indexes and then only remove Net Search Extender.

Uninstalling Net Search Extender on UNIX

To uninstall Net Search Extender on UNIX operating systems, you must disable the database, stop the instances, and issue the **`db2nse_deinstall`** command.

Procedure

Complete the following steps to uninstall Net Search Extender correctly on UNIX operating systems:

1. For each DB2 instance from which you want to uninstall Net Search Extender:
   a. Switch to the user ID of the DB2 instance.
   b. If you no longer intend to use Net Search Extender on this instance, you should drop the indexes and disable the database before uninstalling Net Search Extender:
      ```
      db2text disable database for text connect to databasename
      ```
   c. Stop the DB2 Net Search Extender instance.
   d. Stop the DB2 instance.
2. Ensure that you are active as the root user.
3. Change your working directory to the DB2 path where you want to remove Net Search Extender. For example, `cd /opt/IBM/db2/V10.5/install`.
4. Issue the command **`./db2nse_deinstall`**. For details on the command syntax, see "db2nse_deinstall command" on page 242.

Uninstalling Net Search Extender on Windows

To uninstall Net Search Extender on Windows operating systems, disable the databases, stop the instances, and remove the program.

Procedure

Complete the following steps to uninstall Net Search Extender correctly on Windows:

1. For each database run `db2text disable database for text connect to` *`databasename`*.
2. Stop the DB2 instance.
3. Select **Settings** > **Control Panel** > **Add or Remove Programs**. From the list, select the Net Search Extender *COPYNAME* entry that correlates with the DB2 *COPYNAME* that Net Search Extender was assigned during installation.
4. Click **Remove**.

Part 3. Configuring Net Search Extender (NSE) for high availability (HA)

DB2 Net Search Extender can be configured to support high availability by sharing the indexes between the high availability nodes, as well as Net Search Extender index backup and restore operations. Net Search Extender full-text indexes consist of data stored in a DB2 database, and in some external files located on the file system. Only Net Search Extender data within the database are recovered during failover by configuring DB2 for high availability. The NSE specific external files need to be shared with the fail-over node, using a file sharing technology applicable to the user scenario and the platform in use. The external files are not restored if any index update operation are interrupted, causing the index files to get corrupted. The files need to be backed-up to restore them manually.

Interrupting an index update can irreparably and unpredictably corrupt the index. The fatality of the corruption depends on affected index files and the index operation phase at the time of interruption. Some of the index files are also updated directly, instead of their copies, making rollback recovery more difficult. So if failover occurs during an index update, corrupted index files need to be restored from the last successful index update operation, which are saved as index directory snapshots.

High availability configurations prevent index files that are located on shared storage from getting into an inconsistent state if an index update is interrupted during a failover. Database objects found on the failover system can be used to revert the index files back to a consistent state.

If the file snapshot is not supported by any platform, any corresponding file system sharing technology applicable to that platform shall be considered for NSE shared index folder/drive.

Index directory snapshots

1. All Net Search Extender index files must be stored on dedicated file systems to backup and restore the latest index files. No other data should be stored on the file system.
2. Every index must reside on its own file system. Alternately, indexes can share file systems, but the update schedules for indexes sharing a common file system are serialized in such a way that no two updates occur at the same time. The number of distinct file systems for Net Search Extender indexes are then be adapted to the number of parallel update processes that the system is capable of handling.
3. The space used by a snapshot is initially very small, but tends to increase as file system content is changed. Ensure there is sufficient file space for snapshot on the index file system. Monitor file space usage to ensure ample space exists for snapshots.

Preparing for failover

Indexes are located on shared storage between the high availability nodes. Every index update and scheduled update should immediately be followed by a snapshot of its index directory. These instructions can be encapsulated in a scrip and executed by an external scheduler, as shown in the following steps:

1. Verify whether the index files are located in the shared location between the high availability nodes.
2. Check DB2 Net Search Extender status from `db2ext.tcommandlock` table and `work` directory.
3. Run snapshot procedure to take Net Search Extender index file system snapshot to the shared storage.
4. Invoke the Net Search Extender **UPDATE INDEX** command
5. Remove self-defined mark after the index update completes.

Note: Since the Net Search Extender native scheduled index update can only invoke the **DB2TEXT UPDATE INDEX** command, disable it by setting **UPDATE FREQUENCY** to `NONE`. Use operating system specific index update scheduling instead, such as the **CRON** command on UNIX and Linux, and **AT** command on Windows operating systems. These commands invoke the wrapper script at the specified interval, with one **crontab** entry for every index that has an automated update schedule. This ensures the existence of current snapshots of all indexes on the file system from the most recent successful update on shared storage.

Index characteristics during failover

Key to index recovery is determining whether or not failover corrupted the index. This requires a fallback to the most recent known good state of that index and can be determined by the following Net Search Extender index update process:

- Every index update is internally encapsulated in a pair of insert and delete operations on the `db2ext.tcommandlocks` table.
- To prevent concurrent admin commands on that index, the index update starts by creating a row in this table, a named index, a timestamp, and the type of operation. The row is removed from the table again before the update terminates, making the index available for new administration commands.
- If no index update occurs during a failover then the `db2ext.tcommandlocks` table contains no rows, and no further action is required. All data stored in the log table are immediately available on the failover system via high availability support, ready for the next regular index update.
- If a failover happens during an index update, the `db2ext.tcommandlocks` table on the failover node will show a row for every index that was involved in an update at the time of the failure. There can be more than one affected index, each corresponding to a single row in `db2ext.tcommandlocks`, so every operation needs to be repeated for each row. Manual recovery then needs to be initiated to restore the snapshot. Every affected index is protected against further (scheduled or manual) updates by the presence of the lock entry on the table.
- Check whether the entries in the log table still persist. Compare the timestamp of the oldest entry in the log table of the index with the most recent `CTE0003` entry in the event table of the index.

  If the oldest log table entry is younger than the most recent `CTE0003`, log table cleanup had been done already before the failover, but the `db2ext.tcommandlocks` entry couldn't be deleted yet. The index is uncorrupted in this case, so do not restore the snapshot, but only manually remove the `db2ext.tcommandlocks` entry and proceed as usual.

  If the oldest log table entry is older than the most recent `CTE0003`, the index should be restored from snapshot.

Restoring Index from snapshot

1. Remove all index files in the index directory of the affected index. Note that searches against that index will fail during that time, so stop Net Search Extender.

   ```
   rm -rf /myWORK/NODE0000/TMP_IX300608/*
   ```

2. Replace the empty directory with the content of the snapshot. This takes time since it requires a physical copy of the files.

   ```
   rm -rf /myINDEX
   mount -o snapshot /dev/fslv06 /mnt/
   cp -pR /mnt/* /myINDEX
   ```

3. After the restoring the index directory content, manually remove the row corresponding to the index from `db2ext.tcommandlocks` table.

   ```
   db2 "delete from db2ext.tcommandlocks"
   ```

4. Repeat the previous steps for all affected indexes
5. When done, restart Net Search Extender. Regular operation can proceed now on the failover node.

The Net Search Extender content of the log table remains intact and a new call to **`DB2TEXT INDEX UPDATE`** will process it as before. Some manual cleanup in the event table may be necessary, since it can contain entries created during the original index update operation.

Part 4. Upgrading to DB2 Net Search Extender

Before upgrading Net Search Extender to DB2 Net Search Extender, you must have successfully upgraded your DB2 server and instance (on Linux and UNIX) including all databases to DB2 Version 10.5

Before you begin

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

- Back up all text index directories and index subdirectories before upgrading. Refer to Chapter 39, “Backing up and restoring text indexes,” on page 125.

About this task

Upgrade to Net Search Extender is supported from Net Search Extender Version 9.5 or Version 9.7.

Procedure

To upgrade to DB2 Net Search Extender:

1. Upgrade your DB2 server where Net Search Extender is installed using any of the following tasks:
   - Upgrading DB2 servers (Windows) found in the *Upgrading to DB2 Version 10.5*
   - Upgrading DB2 32-bit servers to 64-bit systems (Windows) found in the *Upgrading to DB2 Version 10.5*
   - Upgrading a DB2 server (Linux and UNIX) found in the *Upgrading to DB2 Version 10.5*

   Upgrading your database is part of these tasks. If you have external unfenced routines on Linux or UNIX that have no dependency on the DB2 engine libraries, the **UPGRADE DATABASE** command redefines your external routines as FENCED and NOT THREADSAFE. See Upgrading C, C++, and COBOL routines in Upgrading to DB2 Version 10.5 for details on how to safely run your routines in the new multithreaded database manager. The Net Search Extender functions with schema name DB2EXT that were altered during the database upgrade are redefined as NOT FENCED and THREADSAFE by the `db2extmdb` script in step 6.
2. Install DB2 Net Search Extender .

   Unlike DB2 database, DB2 Net Search Extender does not support an 'upgrade installation'.

   If your installed DB2 copy was moved to by using the DB2 'upgrade installation' option, the installed DB2 Net Search Extender copy is still at the previous version level.

   If you try to install DB2 Net Search Extender on top of an earlier version of DB2 Net Search Extender, you receive an error message that the existing DB2 Net Search Extender installation has to be removed first. In this case, uninstall the earlier version of DB2 Net Search Extender before installing DB2 Net Search Extender . On Windows operating systems, reboot the machine after uninstallingDB2 Net Search Extender.

3. A DB2 Net Search Extender instance upgrade is only applicable on Linux and UNIX. This step has to be ignored on Windows operating systems. To upgrade the instance, log on as root and run the **db2extimigr** script using the following syntax:

   ```
   DB2DIR/instance/db2extimigr [-h|-?] InstanceName
   ```

   Where *DB2DIR* is the directory where you installed your DB2 Version 10.5 copy.
4. On Linux and UNIX, after a successful DB2 Net Search Extender instance upgrade, verify the installation before you continue with the database upgrade. On Windows operating systems, you can verify the installation immediately.

   Note: Do not apply the DB2 Net Search Extender sample scripts to a database that is not upgraded to DB2 Net Search Extender . A secure way is to create a database to carry out the verification. See “Installation verification on UNIX” on page 35 and “Installation verification on Windows” on page 35 for more details.
5. Upgrade every database that had been enabled for Net Search Extender in a pre- release. Step a) and c) are currently needed to resolve a known problem with **db2extmdb** (duplicate entries in the view DB2EXT.DBDEFAULTS after database upgrade). To perform the database upgrade steps:
   a. Log on to the DB2 server as the instance owner.

      You must be able to successfully stop or start the Net Search Extender instance services, you need DBADM with DATAACCESS authority on the database to be upgraded. On Windows operating systems, the instance user must be part of the Local Administrator group.

      On Windows operating systems, you need to proceed from a DB2 command window that is running with full administrative privileges. See User Access Control feature for more details.

      On the Windows operating system, the **db2extmdb** command does not work if it is started from a command window running with standard user rights. ("CTE0228 The user has insufficient access rights at the operating system level").
   b. Run the **db2extmdb** script to upgrade the database that you enabled for Net Search Extender using the following syntax:

      ```
      db2extmdb database-name
      ```

      While you are running this script, avoid changing user tables with text indexes. You can repeat the command for every database that has Net Search Extender indexes.

      All upgrade steps are logged in the file called `db2extm`*`database-name`*`.log` located in one of the following directories:
      - *`INSTHOME`*`/sqllib/db2ext/` on Linux and UNIX operating systems
      - **DB2PATH**\db2ext\ on Windows operating systems

      Where *INSTHOME* is the instance home directory and **DB2PATH** is the location where you installed your DB2 Version 10.5 copy.
6. If you upgraded from pre-Version 10.57 DB2 32-bit server to a Version 10.5 64-bit server, you must drop your text indexes and re-create your text indexes. Refer to Chapter 37, “Dropping a text index,” on page 121 and Chapter 28, “Creating a text index,” on page 93. In Net Search Extender, you cannot use in

a 64-bit instance text indexes that you created in a 32-bit instance. The search engine returns error CTE0101 Reason code: "17".

7. If you want to use text indexes that you created under the installation directories of your pre-Version 10.5 DB2 copies before DB2 server upgrade on Windows operating systems, restore the text index directories that you backed up.

   Refer to Chapter 39, “Backing up and restoring text indexes,” on page 125. Restoring the text index directories is required if you chose a DB2 copy with the upgrade action in the **Work with existing** window during DB2 Version 10.5 installation or you uninstalled your pre-Version 10.5 DB2 copies after upgrading.

   The text index configuration contains the location of these text index directories before migration. Queries and index administration operations that use these text indexes fail if you do not restore the text index directories.

Part 5. DB2 Net Search Extender index migration tool from 32-bit to 64-bit

Net Search Extender indexes built under 32-bit DB2 instances are not compatible with 64-bit instances.

Trying to search on an index, or to update an index in a 64-bit DB2 instance built under a 32-bit instance always results in an error message preventing the operation from completion. The reason for this is a word size-specific format of a number of small files that are part of the index.

Currently, the only recommendation for upgrading from 32-bit to 64-bit instances, is to drop and re-create the NSE indexes that you had built in your 32-bit instances. If you have indexes of small or moderate size, where the textual content of the table is accessible directly, you can follow this recommendation with low or moderate technical effort. As indexes grow, the rebuilding of the NSE indexes takes considerable amount of time, even though the DB2 database system is able to process up to 12 GB/hr on well-tuned systems.

To avoid the need to completely rebuild your indexes, you can use a tool to migrate the indexes offline. The current version of tool supports only AIX and Solaris operating systems.

For Content Manager Systems, the content indexed by Net Search Extender is not local to the table on which the index is defined for all non-attribute indexes. Attributes are metadata stored on the library server and their content is local. For the CM systems, the text documents are not local to the Net Search Extender database table enabled to full text search. Instead the documents are kept in a separate document server and sent to Net Search Extender for indexing. The content is retrieved and filtered from a remote location (in CM lingo: the Resource Manager) with considerable latency penalties. This slows down the indexing itself, and thus makes the rebuilding of NSE indexes prohibitively expensive.

If you are currently operating on 32-bit instances of DB2 and are still running DB2 in the Version 8 release, on AIX and Solaris platforms you have to update the DB2 instances from 32-bit to 64-bit before you migrate from the Version 8 release to releases Version 9.1, Version 9.5 or Version 9.7. For all these releases, 32-bit instances no longer exist on these two platforms. The situation is also applicable on Linux and Windows, where 64-bit instances are recommended for larger scale systems due to the absence of memory usage restrictions under a 64-bit regime.

Index migration (as described in the following section) has to be performed before a 32-bit to 64-bit migration of the DB2 instance and databases.

Index Migration Procedure

1. Download the Index migration tool from the FTP site.
2. Extract the compressed file to any directory on your system.
3. Index migration tool can be executed by running the shell script **`ctemigridx.sh`**. Execute the shell script (`./ctemigridx.sh`) as described in the following section.

For DB2 Version 8 32-bit AIX environments:

1. Log in as the instance owner
2. Extract the archive `NSE_32_64_Idx_Migr_Tool_AIX_SOL.tar.gz`.
3. Navigate into the `NSE_32_64_Idx_Migr_Tool_AIX_SOL` directory.
4. Check if the DB2 database manager is up and running and NSE is stopped.
5. Run **`/usr/sbin/slibclean`** to clear the AIX library cache. This is necessary, to avoid conflicts between installed NSE libraries and libraries of the same name in the `NSE_32_64_Idx_Migr_Tool_AIX_SOL` directory. Depending on your system configuration, you might need root user authority to be able to run **`slibclean`**.
6. Run the shell script **`ctemigridx.sh`**. The script can be executed in one of the two available modes.

   The first mode takes a database name as argument, and automatically determines all existing indexes in that database and offers to migrate them all at once, or selectively

   Mode 1: This mode establishes a database connection and queries all data that is needed from the NSE database tables. After that it shows a list of indexes that are ready to get migrated. You can choose one index or all listed indexes. Adding the **`-check`** parameter to the command causes it to perform all the necessary steps without the migration.

   Example: `./ctemigridx.sh -dbname sample`

   Mode 2 can be used to migrate a particular index, if you know the index name already, and want to target the migration of this index. This mode is useful if you have a large number of indexes, and want to avoid going through a long menu of indexes. It also comes handy if you are running repeated tests with an individual index. This mode takes index information and migrates it silently without user interaction.

   Mode 2: `./ctemigridx.sh -i index-name -p index-directory [-showmap]`

   Example: `./ctemigridx.sh -i IX123456 -p /home/user/sqllib/db2ext/indexes`

   The index directory has to be specified in the same way as it is used at index creation time. The index directory always has a `NODE0000` subdirectory which then contains the index itself. If you add the **`-showmap`** flag, the log file shows a dump of the migrated index attribute (sections and attributes) for additional verification purposes.

   ```
   . /ctemigridx.sh -i IX123456 -p /home/user/sqllib/db2ext/indexes
   -showmap
   ```
7. After this, index migration is complete; the user can proceed to perform the instance migration to 64-bit as described in the DB2 documentation.

For DB2 Version 8 32-bit Solaris environments:

All steps are identical to the ones that need to be done for AIX, only the **`slibclean`** step is not needed. **`slibclean`** does not exist on Solaris.

ctemigridx.sh syntax

```
►►──./ctemigridx.sh──┬──────────────────────────────────────────────────┬──────►◄
                     ├─-h───────────────────────────────────────────────┤
                     ├─-H───────────────────────────────────────────────┤
                     ├─-?───────────────────────────────────────────────┤
                     ├─-dbname──database-name──┬────────┬───────────────┤
                     │                         └─-check─┘               │
                     └─-i──index-name──-p──index-directory──┬──────────┬┘
                                                            └─-showmap─┘
```

-h, -H, -?
Shows command help and exits.

-dbname *database-name*
The name of the database.

-check
Performs all the necessary steps without migration (simulation).

-i *index-name*
Name of the index to be migrated, it is always in the form `IX`*nnnnnn*.

-p *index-directory*
The directory where the index is located. It has to be specified in the same way as it is used at index creation time. It is `~/sqllib/db2ext/indexes` in most cases.

-showmap
Causes the log file to show a dump of the migrated index attributes (sections and attributes) for additional verification purposes.

Note:
1. The current migration tool does not support the migration of indexes restored in an already migrated Version 9 instance.
2. The DB2 instance must be started and NSE must be stopped before running the index migration tool.
3. You must have write permission on the current directory before running the tool.
4. The tool backs up index files that will be changed only, it does NOT back up the entire indexes. It is advisable to back up the entire index directory before proceeding with execution of the tool.
5. You must execute the tool as DB2 instance owner but not as root.
6. If the **db2_local_ps** command does not give results, then the DB2 database manager is not started and the tool will not work.
7. Currently the migration tool supports single-node systems only.

Part 6. DB2 Net Search Extender index migration tool from 32-bit to 64-bit (Windows)

DB2 Net Search Extender indexes built under 32-bit DB2 instances are not compatible with 64-bit instances. Trying to search or to update an index that was built under a 32-bit DB2 instance in a 64-bit instance results in an error message being returned.

The index migration tool migrates indexes offline and avoids rebuilding indexes. The migration affects only a small number of metadata files and is performed quickly without a large amount of data movement. To perform this migration execute the batch script which is delivered within this package.

Migrate index from V8 32-bit to V9 64-bit

This migration tool runs only on a 32-bit source Net Search Extender instance which will be migrated to 64-bit. It should not be run on a 64-bit instance. Take a backup of the database and index directory from V8 system. While doing the backup ensure that metafile with extensions .an,.as and .tf are backed-up properly.

Migration steps

1. Prepare the database by dropping `DATALINKS` user defined functions in the DB2 Version 8 database:
   ```
   db2 Drop Specific Function DB2Ext.DataLinkContent1
   db2 Drop Specific Function DB2Ext.DataLinkContent1
   db2 Drop Specific Function DB2Ext.DataLinkContent2
   db2 Drop Specific Function DB2Ext.DataLinkContent4
   db2 Drop Specific Function DB2Ext.DataLinkContent3
   db2 Disconnect all
   ```
2. Migrate the 32-bit index files to 64-bit using the `ctemigridx` migration tool. See steps to run the tool on V9 32-bit machine for details.
3. Take a backup of the database and index files from the 32-bit instance.
4. Move the backed-up files and database to the target machine where 64-bit DB2 Version 9.1 or Version 9.5 is installed.
5. Before migrating Net Search Extender indexes to the target machine, migrate the required Net Search Extender databases to the target machine.
6. Restore the Net Search Extender indexes and the newly generated 64-bit metafile into the target machine by the following procedure:
   a. Issue the following command to stop Net Search Extender:
      ```
      db2text stop
      ```
   b. Restore the backup copies of the index directories to the same path as before.
   c. Issue the command to restart Net Search Extender:
      ```
      db2text start
      ```
7. Migrate the Net Search Extender database to the current release.
   ```
   db2extmdb <database-name>
   ```
8. Perform alter index by mentioning new index and work directories. The instance owner requires adequate permission to access and alter the index files.
   ```
   db2text alter index <index-name> for text index
   directory <new_indexdir> work directory <new_workdir>
   ```

9. Perform searches as before and verify whether the new migrated index is searchable.

Migrate index from V9 32-bit to V9 64-bit

This migration tool runs only on a 32-bit source Net Search Extender instance which will be migrated to 64-bit. It should not be run on a 64-bit instance. Take a backup of the database and index directory from a V9 32-bit system. While doing the backup ensure that metafile with extensions .an,.as and .tf are backed-up properly. The DB2 Version 9 32-bit to Version 9 64-bit migration can be performed in two different ways depending on whether or not the source and target machines are the same.

- Source and target machines are different
  1. Migrate the working V9 32-bit database and Net Search Extender indexes to 64-bit using the `ctemigridx` migration tool. See steps to run the tool on V9 32-bit machine for details.
  2. Take a backup of the database and index files from the 32-bit instance.
  3. Move the backed-up files and database to the target machine where V9 is installed.
  4. Before migrating Net Search Extender indexes to Version 9 64-bit, migrate the required Net Search Extender databases to DB2 Version 9 64-bit.
  5. Restore the Net Search Extender indexes and the newly generated 64-bit metafile into the V9 64-bit setup by the following procedure:
     a. Issue the following command to stop Net Search Extender:
        ```
        db2text stop
        ```
     b. Restore the backup copies of the index directories to the same path as before.
     c. Issue the command to restart Net Search Extender:
        ```
        db2text start
        ```
  6. Migrate the Net Search Extender database to the current release.
     ```
     db2extmdb database-name
     ```
  7. Perform alter index by mentioning new index and work directories. The instance owner requires adequate permission to access and alter the index files.
     ```
     db2text alter index index-name for text index
     directory new_indexdir work directory new_workdir
     ```
  8. Perform searches as before and verify whether the new migrated index is searchable.
- Source and target machines are the same

  The following sequence of steps describe how to migrate a working V9 32-bit database and Net Search Extender indexes to a 64-bit Windows machine:
  1. Migrate the 32-bit index files to 64-bit using the `ctemigridx` migration tool. See steps to run the tool on V9 32-bit machine for details.
  2. Migrate the DB2 instance and database from V9 32-bit to V9 64-bit.
  3. Migrate the Net Search Extender database to the current release, using the `db2extmdb` command

Steps to run the tool on V9 32-bit machine:

1. Logon as the DB2 instance owner.
2. Extract the `NSE_32_64_Idx_Migr_Tool_WINDOWS.zip` archive CD to the `NSE_32_64_Idx_Migr_Tool_WINDOWS` directory. Make sure that DB2 instance is up

and running and Net Search Extender is stopped. Ensure that no other process is attempting to get an exclusive lock on the index files located in the index directory.

3. Run the batch file `ctemigridx.bat` file. The script can be run in two different modes.

   Mode 1 will establish a database connection and queries all data that is needed from the Net Search Extender database tables. This mode takes a database name as argument, and automatically determines all existing indexes in that database and offers to migrate them all at once, or selectively. After that it shows a list of indexes that are ready to get migrated. You can choose one index or all listed indexes. Adding the `-check` parameter to the command will perform all the necessary steps without the migration.

   ```
   ctemigridx -dbname sample
   ```

   Mode 2 can be used to migrate a particular index, if you know the indexname already and want to target the migration of this index. This mode is useful if you have a large number of indexes and want to avoid going through a long menu of indexes. It is also handy if you are running repeated tests with an individual index.

   ```
   ctemigridx -i index_identifier -p index_directory -[showmap]
   ```

   The index directory has to be specified in the same way as it is used at index creation time. The index directory always has a `NODE0000` subdirectory which then contains the index itself. Currently the script only supports single-node-systems. If you add the `-showmap` flag, the logfile will show a dump of the migrated index attribute for additional verification.

   ```
   ctemigridx -i IX123456 -p D:\sqllib\db2ext\indexes -showmap
   ```

4. When the Net Search Extender index migration is done, some `*.32` files can be found in the index directory. According to the indexes that have been migrated these new files can be found as follows:
   - IX123456.as.32 (Backup of the old 32 bit IX123456.as file)
   - IX123456.an.32 (Backup of the old 32 bit IX123456.an file)
   - IX123456.tf.32 (Backup of the old 32 bit IX123456.tf file)

   The migration creates a temporary directory that is created during migration. In case it is not there, a `TMP_IX123456` directory can be found inside the Net Search Extender index directory.

5. After verification of the 64-bit Net Search Extender indexes, the `*.32` files can be removed or transferred to another location for backup.

Index migration is complete. You can now perform the instance migration to 64-bit.

Part 7. Planning considerations

Planning before using features is a good practice to ensure you're getting the most out of the database functionality.

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

To use Net Search Extender in the most effective way, it is essential that some planning occurs before deployment. Planning might involve several user groups including those in database administration, interface and system designers, system architects, and developers.

The following topics provide a guide to the areas that you should consider:
- Directory locations and index storage
- Table, column, and index names
- Document formats and supported code pages
- Preventive measures against Net Search Extender index file corruption
- Outside In filtering software
- User roles
- Extended text-maintained staging infrastructure for incremental update

For more information about developing Net Search Extender based applications, see the following related topics:

Chapter 13. Directory locations and index storage for Net Search Extender

The disk space you need for a Net Search Extender index depends on the amount and type of data you want to index.

As a guideline for indexing single-byte documents, reserve disk space of about 0.7 times the size of the documents you want to index. For double-byte documents, reserve the same disk space as the total size of the documents you want to index. The total size might have to include data stored outside the active database that is retrieved through user-defined functions.

The amount of space needed for the temporary files in the work directory is 1.0 to 4.0 times the amount of space needed for the final index file in the index directory. The default index directory is a subdirectory of the DB2 instance directory which is usually located in the `/home` partition of the system (for Linux and UNIX operating systems) and on the C: drive (for Windows operating systems). The default index might have size limitations as well. See "Views for database-level information" on page 269 for more details.

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

For each index, the corresponding index and work directory should reside on the same file system or drive. If you are not using the default location, then care should be taken during the create index command to specify both the index and work directory locations, such that they are on the same file system. Specifying only the index directory in the command leads to the work directory getting created in the default path (which in most cases may not be on the same file system) and vice versa.

For a partitioned database, a text index still uses a single file system on a physical machine to place the index. For large databases, place the index and work directories on a file system that is on a Redundant Array of Independent Disks (RAID) device. This minimizes the possibility of hitting an I/O bottleneck while using the text indexes.

For creating, updating, and deleting Net Search Extender indexes, use the command line interface.

Chapter 14. Resource considerations for a partitioned DB2 server

When running NSE in a partitioned environment, you should keep the following resource considerations in mind:

- When multiple text indexes are updated in parallel, this can lead to considerable burst disk usage during the I/O intensive phases of the index update. This requires careful consideration during the setup of partitions, the associated NSE index storage, and scheduling of index updates.
- For NSE administration commands, there are processes started for every partition to carry out operations such as index create, drop, and update. The update process can be a longer running process that will consume resources. For scheduled updates of text indexes in the system, several index updates running concurrently can lead to as many processes per partition as the number of indexes being updated at a given time. It would be advisable to minimize the number of concurrent index updates by planning the index update schedule accordingly.
- On Linux or UNIX operating systems, you can use the command **`ulimit`** with the appropriate options to view or change the size of the process resource limit. If you are running DB2 Net Search Extender in a partitioned database environment, use the command **`db2_all ulimit`** with the specific options to see the size that applies to all the database partitions. It is essential to verify the hard operating system ulimits for successfully running Net Search Extender update index commands. For example, insufficient data segment size on any of the partitions can cause failures during index update commands most usually with CTE0105 error logged in the event view of the text index.

Chapter 15. Stored procedure search memory requirements

Different platforms require different amounts of memory when using the cache for a stored procedure search.

Using the cache for a stored procedure search requires a large amount of memory and different memory requirements for the following platforms:
- AIX
- Windows
- Solaris
- Linux

Net Search Extender memory requirements for AIX (64-bit)

Before using the Net Search Extender for AIX (64-bit), you must configure the system limits, shared memory limits and swap space.

Configuring the system limits:
- Check the system limits by issuing the command `ulimit -a`
- If there are values other than "unlimited", use the following steps:
  - Log on as root.
  - Back up the file `/etc/security/limits` and then edit the file to raise the hard limits.
  - Set all values to the "unlimited" (value -1) for the DB2 instance owner used.

Configuring the shared memory limits:
- On AIX, there is no need to configure the shared memory limits.

Configuring the swap space:
- Obtain the system RAM size by issuing the command `lsattr -E -l sys0`
- Obtain the size of the swap space by issuing the `lsps -a` command.
- Set the swap space size to at least 1.5-2 times of either the RAM amount of your system, or use the **`MAXIMUM CACHE SIZE`** parameter you provide in the **`CREATE INDEX`** command. Use the SMIT utility to select the larger number.

Net Search Extender stored procedure memory requirements for Windows (32-bit and 64-bit)

The virtual-memory paging file size must be set according to your Windows machine.

Adjusting the size of the paging file:
- Set the Windows virtual-memory paging file size to at least 1.5 - 2 times of either the RAM amount of your system, or use the `MAXIMUM CACHE SIZE` parameter you provide in the `CREATE INDEX` command. Select the larger number. See the Windows documentation for information about changing the size of the paging file.

  On Windows 32-bit, you are recommended to not exceed a maximum cache size of about 1000 MB (1 GB = 1073741824 bytes).

Net Search Extender memory requirements for Solaris (64-bit)

The system limits, shared memory limits and swap space must be checked and configured according to your Solaris machine.

Configuring the system limits:

- Check the system limits by issuing the command: **`ulimit`** `-a`
- Then use the following steps:
  - Log on as root.
  - Back up the file `/etc/system` and then edit the file to raise the hard limits.
  - Add or check that the following lines are set to at least the minimum values shown:

    `rlim_fd_cur -> Default 64, recommended >= 1024`

    `rlim_fd_cur_max -> Default 1024, recommended >= 4096`

Configuring the shared memory limits:

- Check current settings by issuing the command **`sysdef`** `-i`
- Edit the file `/etc/system` to set the shared memory size limit using: `set shmsys:shminfo_shmmax=0xffffffff`

  You might also have to increase the following parameter values:

  `set shmsys:shminfo_shmmni=512`

  `set shmsys:shminfo_shmseg=128` and then reboot the system.

Configuring the swap space:

- Obtain the system RAM size by issuing the command `/usr/sbin/prtconf`
- Obtain the size of the swap space by issuing the `swap -l` command.
- Set the swap space size to at least 1.5-2times of either the RAM amount in your system, or use the **`MAXIMUM CACHE SIZE`** parameter you provide in the **`CREATE INDEX`** command. Select the larger number.

  Refer to the Solaris system documentation for information about how to add swap space.

  You are recommended not to exceed the maximum cache size of about 2000 MB (2 GB = 2147483647 bytes).

Net Search Extender stored procedure memory requirements for Linux (32-bit and 64-bit)

The current shared resource limits and system limits must be verified for Linux operating systems

Check the DB2 documentation for information about recommended kernel parameters on Linux.

The validation status for new Linux kernels and distributions is frequently updated. To obtain the latest information for supported Linux software levels, refer to: http://www.ibm.com/software/data/db2/linux/validate

To see your current shared resource limits use **`ipcs -l`**. To check the system limits, use the **`ulimit -a`** command.

Chapter 16. Table, column and index name considerations

All table, column and index names are usually case-insensitive.

Net Search Extender allows you to specify these names in mixed case too. On Windows, if you want to specify table, column and index names in mixed case, you must enter the name in a backslash (\) double quote (") character sequence. For example, \"DocTxt\".

Chapter 17. Document formats and supported code pages

Net Search Extender needs to know the format (or type) of text documents that you intend to search.

This information is necessary for indexing text documents.

Net Search Extender supports the following document formats:

TEXT Plain text (for example, flat ASCII), in general, text without any markup

HTML
Hypertext Markup Language

XML Extended Markup Language

Document format XML is the default for column data type XML, and is the only supported document format for that data type.

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

Outside In (INSO)
Use this format if you are using filtering software to extract textual content from PDFs and other common text formatting tools, for example, Microsoft Word.

For the document formats HTML, XML, GPP, and the Outside In filter formats, searching can be restricted to specific parts of a document.

Where Outside In filters can not be used because the format of your document is not supported, you can write a User Defined Function (UDF) that does its own filtering. This UDF must be specified at index creation time and converts the data from the unsupported format to a supported format.

You can index documents if they are stored in one of the supported Coded Character Set Identifiers (CCSIDs). See the DB2 documentation for a list of these code pages.

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

```
db2 GET DB CFG for dbname
```

and take the value written for Database code page.

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

Note that incompatible code pages might cause problems when creating a text index or searching.

Chapter 18. Preventive measures against Net Search Extender index file corruption

A corrupt index file is indicated by error messages that are recorded in the event table of the index, reporting kernel errors with various reason codes, depending on which files were affected by the corruption.

For example: The message similar to the following one would be present in the eventview if an update index is attempted when the index is corrupted.

101 CTE0101 A search engine operation failed. Reason code: "7", "100001", "0", "Kernel return code: 17

You cannot repair a corrupt index; you must drop and re-create it. To avoid having a corrupt index, take the following precautions:

- For production systems, specify an index and work directory in the CREATE INDEX statement instead of using the default directory for Net Search Extender indexes, which is located in the home directory of the instance owner. Use a separate file system for both the index and the work directories, and monitor the file system to ensure that it has sufficient free disk space for update operations. The amount of disk space needed for an update depends on the size of the index (in particular, the size of the secondary index) and the amount of data to be processed during the update. To help estimate the amount of disk space needed, monitor peak usage during update operations.
- To avoid index file corruption caused by Windows access violation errors, take the following steps:
  - Exclude the text index and working directories from automatic backup programs.
  - Exclude the index directories from antivirus scan programs.
  - Turn off the Windows Indexing Service for the drives where you store the index and work files.
- Before you shut down the system, correctly stop Net Search Extender by using the following commands:
  - `db2text control list all locks for database` *`database name`*; repeat this command until no locks are held. For more information, see the description of the "UPDATE INDEX command" on page 234.
  - `db2text stop`

Chapter 19. Outside In filtering software

Net Search Extender supports a third-party document filtering software.

Known as Outside In Transformation Technology from Oracle, you can use the software for extracting the textual contents from PDF files, or from documents written in the proprietary format of common text formatting tools without using native applications. Example formats include Microsoft Word and Lotus® Word Pro®.

Net Search Extender loads the Outside In libraries as plug-ins during **`UPDATE INDEX`**. The libraries are not part of Net Search Extender and need to be installed separately. You need to ensure that Net Search Extender can find the Outside In libraries.

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

To view a list of filtering formats and supported platforms, visit the Oracle website at `http://www.oracle.com`.

Chapter 20. User roles

User roles include DB2 instance owners, database administrators, and text table owners; each having a specific set of administrative rights.

DB2 instance owner

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

Required DB2 authorizations
The SECADM must grant DBADM with DATAACCESS privilege to the instance owner. These authorizations are a prerequisite for the execution of DB2 Net Search Extender administrative commands.

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

Commands for the instance owner
`DB2TEXT START`, **`DB2TEXT STOP`**, **`DB2TEXT CONTROL`** and **`DB2EXTHL`**

The commands are only allowed on the server. In a partitioned database environment, this can be any of the configured nodes. Each command checks if the user running the command is the DB2 instance owner. If you, as the instance owner, decide to use a fenced user ID to run the stored procedure and UDFs, the fenced user must have read and write access to all files in the index directory (with read access to the entire directory path). Be aware that your fenced user ID and instance user ID must be members of the same primary group to grant the instance user ID correct access to files created by the fenced user ID and vice versa. Assign correct group membership and file permissions.

In addition to the instance owner any user with the same primary group as that of the instance owner will also be able to execute **`DB2TEXT START`**, **`DB2TEXT STOP`**, **`DB2TEXT CONTROL`** and **`DB2EXTHL`**

Database administrators

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

Required DB2 authorizations
DBADM

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

Text table owners

The text table owner can create, drop, and change indexes. Note that they must be able to control (by having read and write access) the location of indexes and updates to the full-text indexes.

Required DB2 authorizations and privileges
Owner of text table.

Commands for the text table owner:

`DB2TEXT CREATE INDEX`, **`DB2TEXT DROP INDEX`**, **`DB2TEXT ALTER INDEX`**, **`DB2TEXT ACTIVATE CACHE`**, **`DB2TEXT DEACTIVATE CACHE`**, **`DB2TEXT UPDATE INDEX`**, **`DB2TEXT CLEAR EVENTS`**, and **`DB2EXTTH`**.

Note that the command implementation partially runs under the user ID of the DB2 instance owner. Therefore, grant the instance owner the necessary file system access before creating or altering the text indexes. For details on required permissions as listed for each command see Chapter 57, "Administration commands for the text table owner," on page 209

Chapter 21. Extended text-maintained staging infrastructure for incremental update

A configuration option is available in Version 9.7 to add a staging infrastructure that enables capturing changes that are not recognized through the triggers in the regular log table.

If this option is enabled, updates are captured through a trigger in the regular log table, inserts and deletes are captured in the text-maintained staging table.

This configuration option is enabled by default for range-partitioned tables and disabled for non-partitioned tables. Adding the text-maintained staging infrastructure has a critical impact on the availability and status of the base table for various database operations.

The impact of the text-maintained staging infrastructure is similar to adding a Materialized Query Table (MQT) with deferred refresh. Even though the text-maintained infrastructure does not maintain data in a MQT, the staging table causes corresponding behavior to that shown for a MQT staging table.

For example, after a LOAD insert, the tables require integrity processing to enable subsequent database operations on the base table.

If tables are only updated with database commands that affect all rows in the table, for example with **LOAD REPLACE**, adding the extended staging infrastructure does not provide a benefit, instead, the index should be re-created.

Part 8. Administering Net Search Extender

Chapter 22. Net Search Extender instance services

DB2 Net Search Extender Instance Services consists of locking services and update services.

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

DB2 Net Search Extender Instance Services on Windows are represented by Windows services. On a non-partitioned DB2 instance there is one such service per DB2 instance with a service name:

```
DB2EXT - instance_name
```

On a partitioned DB2 instance there is one such service for each partition of the DB2 instance with a service name:

```
DB2EXT - <instance_name>[-<nodenum>]
```

The following topics explain how to start and stop the DB2 Net Search Extender Instance Services and further discusses Locking Services and Update Services in detail:
- Starting and stopping NSE instance services
- Locking services
- Update services
- NSE information catalogues

Chapter 23. Starting and stopping Net Search Extender instance services using the command line

Before you can maintain text indexes and search your documents, you have to start the Net Search Extender Instance Services.

About this task

For DB2 instances used with partitioned databases, it is highly recommended that the Net Search Extender Instances Services be started and stopped using **`db2text start`** and **`db2text stop`** instead of using the normal Windows methods. This ensures that the Instance Services are started and stopped in the correct order.

Note:

- There must be one Net Search Extender Instance Service per DB2 instance. The locking service maintains the locks for all enabled databases for that instance.
- DB2 Net Search Extender Instance Services on Windows are represented by Windows services. On a partitioned DB2 instance there is one such service for each partition of the DB2 instance.

Procedure

- To start the Instance Services, log on to the DB2 instance owner user ID (UNIX operating systems only) and enter the following command:
  ```
  db2text start
  ```
- To stop the Instance Services, enter the following command:
  ```
  db2text stop
  ```

Chapter 24. Net Search Extender locking services

Net Search Extender locking services are used to prevent reading and writing process from interfering with each other.

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

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

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

Using the locking services

In Net Search Extender, there are different types of locks that control concurrent access to an index.

The different locks depend on whether the text index is only being read, as in the case of a search request, or if changes to the text index need to be computed and subsequently written to files, as in the case of an index update.

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

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

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

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

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

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

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

Changes to the configuration file only take effect when Net Search Extender Instance Services are started during **db2text start**. The user can set the following values:
- The maximum number of databases
- The maximum number of indexes per database

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

The default values of the configuration file are as follows:

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

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

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

        latchTimeout = "80"

/>
```

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

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

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

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

Shared memory usage is proportional to the product of the three 'max' values mentioned previously. To avoid excessive shared memory usage, make sure that the values in use match the actual configuration of your DB2 instance. If you increase the values for maxDbs, maxIdxPerDb, or maxLocksPerIdx beyond the default values in the configuration file mentioned previously, ensure that you have sufficient memory. Pay particular attention to the values for maxIdxPerDb and maxLocksPerIdx if you are using the partitioned database environment because these settings will be used for every partition. This is especially significant in determining memory requirements if a number of logical partitions are defined for a DB2 instance on a physical machine.

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

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

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

latchTimeout

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

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

It is strongly recommended to leave the default values for the wait, attempt and timeout parameters intact. The waiting time is calculated in milliseconds. Note that with each attempt, the latchTimeout value is doubled when added to the overall waiting time.

Viewing a lock snapshot

The lock snapshot can be viewed by using multiple commands. The first time a text index is locked, memory is reserved for the database and the text index in the locking services.

About this task

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

- For a single text index:
  ```
  db2text CONTROL LIST ALL LOCKS FOR DATABASE mydatabase INDEX myindex
  ```
- For all the locked text indexes of a database:
  ```
  db2text CONTROL LIST ALL LOCKS FOR DATABASE mydatabase
  ```
  Note that only indexes that are actually locked are in the list.

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

The command `"db2text CONTROL CLEAR ALL LOCKS"` forces the release of all the locks on a database or index. See “CONTROL command” on page 199 for details on how to use this command. Always use the index specifier when using the **`CLEAR ALL LOCKS`** command. Only use this command if you have checked very carefully that no index update is active on the index where you want to clear the locks. Clearing locks on an index that is currently updated can result in index corruption, and require a complete rebuild of the index. Note that this command does not free any memory allocated for the database or indexes. To free memory, you must either drop the index or disable the database, or restart the Net Search Extender services. Do not release locks during an active index update process.

Chapter 25. Update services

Table changes and index updates are not synchronous. The index update process can be started manually or can be scheduled to begin automatically at given intervals.

The update services provide this functionality and are started during **`db2text start`**.

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

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

In this example, every Monday and Wednesday at 12 p.m. and 12 a.m. the Update Services wake up and check if there is some work to be completed on index `db2ext.title`. Note that in this example there need to be at least five changes to `DB2EXT.TITLE` before the automatic index update will start to synchronize the text index with the database.

In a partitioned database environment, separate update services are started for each of the nodes. When multiple text indexes are updated in parallel, this can lead to considerable burst disk usage during the I/O intensive phases of the index update. This requires careful consideration when scheduling and executing index updates.

Note:

Setting index update processes at very short intervals affects system performance negatively. You must consider the amount of changes that you expect to be processed during each update and the time that this will take, as well as the number of indexes that you want to process during automatic index update. Ensure that the intervals between each index update are large enough to allow for an update to finish before the next scheduled update begins, and that updates on several indexes are not scheduled to begin at the same time.

If the text-maintained staging infrastructure is configured for a text index, ensure that the staging table is not in a pending mode by executing the **`RESET PENDING`** command.

Part 9. Developing: creating and maintaining a text index

There are certain areas of text index creation and maintenance which are important to review before proceeding.

Important: Net Search Extender has been deprecated. Its use is no longer recommended and might be removed in a future release. Use DB2 Text Search as a fast and versatile method of searching full-text documents stored in DB2 databases using SQL and XQuery statements. See the topic about migrating from Net Search Extender to DB2 Text Search for details.

This section provides information about creating and maintaining a text index and covers the following areas:

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

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

Before creating a text index, ensure that you have met the prerequisites in Part 7, "Planning considerations," on page 55. Also make sure that you have started the Net Search Extender Instance Services using the **`db2text start`** command.

Chapter 26. Enabling a database

The **ENABLE DATABASE FOR TEXT** command prepares the database for use by Net Search Extender.

Before you begin

DBADM authority is required.

About this task

Perform this task once for each database that contains tables with columns of text to be searched in.

The **ENABLE DATABASE FOR TEXT** command also registers Net Search Extender search functions and procedures that are described in Chapter 60, “SQL scalar search function and the SQL table-valued function,” on page 253.

When you enable a database, the command additionally creates the following tables and views automatically:

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

db2ext.textindexformats
Stores the list of supported formats and the currently active model files used.

db2ext.indexconfiguration
Stores the index configuration parameters.

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

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

Chapter 27. Disabling a database

When you no longer intend to make text searches on a database, disable it with the **DISABLE DATABASE FOR TEXT** command.

Before you begin

DBADM authority is required on the database.

About this task

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

To disable the connected subsystem, use the following command:

```
db2text DISABLE DATABASE FOR TEXT
```

When you disable a database, the command deletes the following objects and also drops all existing text indexes:

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

If the **DISABLE DATABASE FOR TEXT** command returns an error, but you want to disable at all costs (even if indexes are still in use), use the following command:

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

Note: Disabling a database will fail if there are any text indexes defined in the database. It is recommended to remove these indexes one by one and then check if any problems occur. If you use the **DISABLE DATABASE FOR TEXT FORCE** command, it only guarantees that Net Search Extender catalog tables in the database are removed. The force option cannot be applied if an existing text index makes use of the text-maintained staging infrastructure.

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

- Files in the index, work and cache directory
- Scheduler entries in `ctedem.dat`
- Where an index was created using the replication capture option, the IBMSNAP_SIGNAL, IBMSNAP_PRUNE_SET, and IBMSNAP_PRUNCNTL entries in the tables of the remote database must be deleted manually. These entries can be easily identified using the `APPLY_QUAL='NSEDB2'||`*`instance_name`* `and TARGET_SERVER=` *`database_name`* condition.

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

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

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

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

Chapter 28. Creating a text index

Issue the **CREATE INDEX FOR TEXT** command once for each column that contains text to be searched.

Before you begin

One of the following authority levels is required:
- CONTROL privilege on the index table
- INDEX privilege on the table and either IMPLICIT_SCHEMA authority on database or CREATEIN privilege on index schema
- DBADM authority

About this task

You can create a text index on all data types, although there are different requirements for the following data types:
- Binary data types
- Unsupported data types

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

When you create a text index, Net Search Extender automatically creates the following objects, depending on whether the extended text-maintained staging infrastructure is enabled for the text index or not.

Note: It is mandatory to specify the **ADMINISTRATION TABLES IN** clause if an index is created on a range partitioned table. See CTE0150E for more information.
- With the regular log infrastructure

  A log table
  This keeps track of all changes to rows in the user table. Note that if you select the **RECREATE INDEX ON UPDATE** option or use replication capture tables, the log table is not created.

  An event table
  This collects information about all updates and potential problems during an update of the text indexes.

  Triggers on the user table (added with initial update)
  These add information to the log table whenever a document in the user table is added, deleted, or changed. The information is necessary for index synchronization during the next scheduled or manual index update.

  Note that triggers are only created if you create a log table, and the text index is created on a base table and not on views or nickname tables.
- With the extended log and staging infrastructure:

  A log table
  This keeps track of updates to the documents.

An auxiliary staging table
: This keeps track of inserts and deletes.

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

An update trigger on the user table (added during initial update)
: The update trigger adds the primary key of the affected row to the log table, when a document in the indexed column is updated.

To optimize performance and disk space, the **CREATE INDEX** command has an option to specify a different table space for the tables.

Note: If you use the **LOAD** command to import your documents, the triggers do not fire and incremental indexing of the loaded documents is not possible with the regular infrastructure. In this case, it is preferable to use the **DB2 IMPORT** command as this activates the triggers.

If the extended text-maintained infrastructure is configured for the text index, documents inserted with a load insert operation are captured in the auxiliary staging table, and incremental indexing is possible.

Example

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

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

A primary key must exist on this table.

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

To reverse the changes made by **CREATE INDEX**, use the **DROP INDEX** command. See Chapter 37, "Dropping a text index," on page 121 for this information.

To populate the created index with data from the text column, use the following command:

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

Note that you can only search for documents successfully after the text index is synchronized with the table using a **db2text update** command.

If errors occur during indexing, index update event rows are added to the event table. This happens, for example, when a document queued for indexing can not be found or if the document format is invalid. For additional information, see the description of the "Event view" on page 275.

What to do next

Note: Search summary

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

- The SQL scalar search functions work on all text indexes, except those created on views.

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

Creating a text index on binary data types

When you store data in a column that has a binary data type, for example BLOB or FOR BIT DATA, the DB2 database system does not convert the data.

About this task

The documents retain their original code pages (CCSIDs), which can cause problems when creating a text index as you might have two different code pages. Therefore, you need to determine whether you are using the code page of the database, or the code page specified in the **CREATE INDEX** command.

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

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

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

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

Note that there is no support for documents with different code pages in one text index. For information about how DB2 database products convert document code page settings, go to the Globalization Guide.

Note that the problem does not exist when you create indexes on character data types. For character data types, do not specify the **CCSID** parameter.

Creating a text index on an unsupported data type

You can create indexes on columns contain unsupported and user defined data types by creating a conversion function.

About this task

To create an index, the text columns must be one of the following data types:
- CHAR
- VARCHAR
- LONG VARCHAR
- CLOB
- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC
- DBCLOB
- BLOB
- XML

Procedure

If the documents are in a column of a data type that is not supported, such as a user-defined type (UDT), you must:

1. Provide a conversion function that takes the user type as input and casts it to one of the valid data types as an output type.
2. Specify the name of this conversion function at index creation time. See "CREATE INDEX command" on page 216 for further information.

Example

You intend to store compressed text in a table.

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

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

In a federated database, you can create incremental indexes on nicknames using DB2 Replication. If you create incremental indexes, changes you make to text indexes are automatically recreated in associated tables.

Procedure

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

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

   After this step, nicknames for the replication control tables are available as nicknames under one "capture control schema" on the federated DB2 database. This schema name is important for the **DB2TEXT CREATE INDEX** command.

3. Register the table as a replication source.
4. If the DB2 database system does not automatically create a nickname in the registration step, create a nickname for the replication capture table in the federated database. The replication capture table can either be a Change Data (CD) table or a Consistent Change Data (CCD) table. This nickname is a parameter for the **DB2TEXT CREATE INDEX** command.

   Note that the column names IBMSNAP_OPERATION, IBMSNAP_COMMITSEQ, IBMSNAP_INTENTSEQ, and the names of the primary key columns must not be changed.
5. If you are using DB2 replication source, ensure that your capture program is running. Do not use a cold start for the capture program: if you do, all rows in the IBMSNAP_SIGNAL table for APPLY_QUAL LIKE 'NSE%' have to be reinserted. In the following SQL statement you can see how this is done:
   ```
   INSERT INTO capture_control_schema.IBMSNAP_SIGNAL
   SELECT CURRENT TIMESTAMP, 'CMD', 'CAPSTART', MAP_ID, 'P'
   FROM capture_control_schema.IBMSNAP_PRUNCNTL
   WHERE APPLY_QUAL LIKE 'NSE
   ```
6. You can use the following example to create a text index on a nickname using replication:
   ```
   DB2TEXT
   CREATE INDEX indexname FOR TEXT ON nickname (text_column)
   REPLICATION CAPTURE TABLE capture_nickname
   CONTROL TABLE SCHEMA capture_control_schema
   ```

Creating a text index which the stored procedure search can use

The stored procedure search depends on the cache options that you set. You must create a text index that specifies the type of search results that you want returned.

About this task

If you know in advance what subset of data from your table you want to present to the user and are only interested in the top ranked search results and not the complete results list, you can use the stored procedure search. For the stored procedure search, you need to specify cache options during the **CREATE INDEX** command. Working with a cached index, enables high performance at query time, by moving all the specified data into main memory to avoid costly physical read operations from the table.

Before you update the cached index the first time, ensure that your table already contains documents to avoid updating an index on a non-populated table. This provides better indexing performance and a correct estimate of cache memory requirements.

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

When creating a text index for stored procedure search, you must determine and calculate the following parameters:
- The type of cache (temporary or persistent)
- The type of index update (automatic and incremental, or recreation at each update)

- The maximum amount of memory that Net Search Extender can use, by using MAXIMUM_CACHE_SIZE.
- The amount of free memory necessary for subsequent document updates, by using PCTFREE. Note that this is only for incremental updates.

The following types of cache are available:

A temporary cache
: This is rebuilt with each **DB2TEXT ACTIVATE CACHE** command, and requires reloading the data from your DB2 table to memory. Building the cached index from scratch each time Net Search Extender is restarted or the system is rebooted takes longer than reactivating a persistent cache, especially for large tables. Use a temporary cache only if you are dealing with a small amount of fixed data and do not have to consider the time that it takes to build the cached data.

A persistent cache
: This is maintained on disk and can be quickly mapped to memory by using the **DB2TEXT ACTIVATE CACHE** command. In incremental index update scenarios, the cache must remain activated to allow synchronization between the table and the cached indexed. If this does not occur, the next **DB2TEXT ACTIVATE CACHE** command recreates the cache from scratch.

The following methods of updating a text index are available:

Without the RECREATE INDEX ON UPDATE option
: If the **RECREATE INDEX ON UPDATE** option is not set, automatic index update takes place. The process is triggered by the update index command and the update intervals are determined by the update frequency option. The update process is also known as incremental update.

: Avoid deleting and reinserting a document in the table because slots for a deleted document cannot be reused in the cache. As a consequence, you should avoid changing key columns on an activated index.

With the RECREATE INDEX ON UPDATE option
: This recreates the index on each update. Use `variable` data types in the cache column expressions wherever possible. This will save cache space. Use the corresponding cast expressions in the **CACHE TABLE** clause.

: Use this option if your data is not very stable, that is if you expect to insert more than 50% of your documents after the initial index activation.

Net Search Extender provides two SQL functions to help you determine the **CREATE INDEX** memory parameters. These are MAXIMUM_CACHE_SIZE and PCTFREE.

- MAXIMUM_CACHE_SIZE specifies the maximum size of the cached index. You can obtain the value of MAXIMUM_CACHE_SIZE in megabytes (MB) by using the following UDF function:

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

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

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

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

For additional information, see Chapter 15, "Stored procedure search memory requirements," on page 61.

- PCTFREE specifies what percentage of the cache that is specified in MAXIMUM_CACHE_SIZE to hold free for additional documents. The following UDF function returns the recommended **PCTFREE** value based on the actual and maximum numbers of documents.

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

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

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

  The default is set to 50%. If you are recreating the index on each update, set the **PCTFREE** value to `0`.

Example

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

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

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

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

What to do next

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

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

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

Chapter 29. Creating text indexes on views

You can create text indexes on views for use with the stored procedure or table-valued search functions.

About this task

You cannot, however, use any of the scalar functions (for example, CONTAINS). Another major limitation is that you cannot create triggers on views, so any changes in the underlying base tables are not recognized automatically.

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

Example

1. Create a base table using the following command:

   ```
   db2 "create table DB2EXT.TLOGIX140789
   (key INTEGER not null  PRIMARY KEY,
   name VARCHAR(50) not null, comment VARCHAR(90))"
   ```

2. Add some entries using the following commands:

   ```
   db2 "insert into DB2EXT.TLOGIX140789 values
   (1,'Claus','works in room 301')"
   db2 "insert into DB2EXT.TLOGIX140789 values
   (2,'Manja','is in the same office as Juergen')"
   db2 "insert into DB2EXT.TLOGIX140789 values
   (2,'Juergen','has the longest way to Raiko')"
   db2 "insert into DB2EXT.TLOGIX140789 values
   (3,'Raiko','is sitting in the office besides Claus ')"
   ```

3. Create a view using the following command:

   ```
   db2 "create view sampleview as select key, comment from DB2EXT.TLOGIX140789"
   ```

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

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

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

   When you use the stored procedure search in a partitioned database environment, you must explicitly specify a table space for administration tables on a single partition and explicitly call on this partition. To ensure that you connect to the correct partition, use the **DB2NODE** environment variable.

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

   ```
   db2 "insert into DB2EXT.TLOGIX140789 values
   (4,'Bernhard','is working on the same floor
               as Manja, but not as Claus')"
   db2 "insert into DB2EXT.TLOGIX140789 values
   (5,'Guenter','shares the office with Raiko')"
   ```

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

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

   This is the layout of the log table:

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

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

   ```
   db2 "insert into DB2EXT.TLOGIX140789 values(0,CURRENT TIMESTAMP,4)"
   db2 "insert into DB2EXT.TLOGIX140789 values(0,CURRENT TIMESTAMP,5)"
   ```

   The first value describes the operation (0 = insert, 1 = update, 2 = delete). The second should always be the CURRENT TIMESTAMP, and the last value is the primary key of the row that has been inserted, updated or deleted.
7. Use the following command to update the index again:

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

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

Chapter 30. Creating a text index on range partitioned tables

You can create text indexes on range-partitioned tables with and without the extended text-maintained staging infrastructure that supports incremental index updates.

About this task

To disable the infrastructure for a text index on a range-partitioned table, specify the **CREATE INDEX** command with the **AUXLOG** parameter set to `OFF`, as shown in the following example:

```
db2text create index sampleix for text on sample(comment) administration tables in
mytablespace index configuration(auxlog off) connect to mydb
```

In this case, the primary log table is added, and document changes are recognized through triggers. Note that the **ADMINISTRATION TABLES IN** clause must be used when creating indexes on range partitioned tables otherwise you will encounter an error.

You cannot use an incremental update to process changes related to attaching or detaching ranges or to process documents that you loaded into an added partition by using the **LOAD** command with the **INSERT** parameter. You must re-create the text index to synchronize it with the base table.

When the extended text-maintained staging infrastructure is enabled for the text index, document updates are captured through an update trigger into the primary log table, while document inserts and deletes are captured in the auxiliary staging table through integrity processing. This process is shown in the following sample scenarios:

Example

Scenario 1: Attaching a partition for a table with extended text-maintained staging infrastructure

```
db2 "create table uc_007_customer_archive (pk integer not null primary key,
customer varchar(128) not null, year integer not null,
address blob(1M) not null) partition
by range(year)(starting(2000)ending(2001)every 1)"
```

```
db2text "create index uc_007_idx for text on uc_007_customer_archive (address)
administration tables in mytablespace"
```

```
db2 "select indexname, logviewname, auxstagingname from db2ext.textindexes"
```

```
db2text "update index uc_007_idx for text"
```

```
db2 "create table uc_007_customer_2001 (pk integer not null primary key,
customer varchar(128) not null, year integer not null, address blob(1M) not null)"
```

```
db2 "import from uc_007_2001.del of del lobs from ./data modified by codepage=1208
insert into uc_007_customer_2001"
```

```
db2 "alter table uc_007_customer_archive attach
partition p2001 starting(2001) ending(2002)
exclusive from uc_007_customer_2001"
```

Note that the changes are not visible yet, and an integrity processing is required.

```
db2 "select * from sysibmts.systsauxlog_ix253720"
```

```
PK      GLOBALTRANSID        GLOBALTRANSTIME             OPERATIONTYPE
----- -------------------    -------------------         ----------------
0 record(s) selected.
```

```
db2 "set integrity for uc_007_customer_archive immediate checked"
```

Integrity processing will place dependent tables in a pending mode.

```
db2 "select * from sysibmts.systsauxlog_ix253720"
```

```
PK      GLOBALTRANSID             GLOBALTRANSTIME                OPERATIONTYPE
----- -------------------------  ---------------------          ---------------
SQL0668N Operation not allowed for reason code "1" on table
"SYSIBMTS"."SYSTSAUXLOG_IX253720". SQLSTATE=57016
```

Perform integrity processing for the text-maintained staging table(s). The command will process all text indexes for the table

```
db2text "reset pending for table uc_007_customer_archive for text"
```

```
db2 "select * from sysibmts.systsauxlog_ix253720"
```

```
PK      GLOBALTRANSID           GLOBALTRANSTIME               OPERATIONTYPE
----- ----------------------  ----------------------           ---------
  1   x'000000000002215B'      x'20081020204612500381000000'       1
  2   x'000000000002215B'      x'20081020204612500602000000'       1
  3   x'000000000002215B'      x'20081020204612500734000000'       1
  5   x'000000000002215B'      x'20081020204612500864000000'       1
```

The incremental update will process the data from the newly attached partition

```
db2text "update index uc_007_idx for text"
```

Scenario 2: Detaching a partition for a table with extended text-maintained staging infrastructure

```
db2 alter table uc_007_customer_archive detach partition p2005 into t4p2005
SQL3601W
```

```
The statement caused one or more tables to automatically be placed
in the Set Integrity Pending state.  SQLSTATE=01586
```

```
db2text "reset pending for table uc_007_customer_archive for text"
```

```
db2text "update index uc_007_idx for text"
```

Chapter 31. Performance considerations for indexing

Issues must be considered when attempting to enhance performance during indexing.

To enhance performance during indexing, consider the following issues:

- Use a VARCHAR data type to store the text documents instead of LONG VARCHAR or CLOB.
- Use separate physical disks to store the text index and the database files.
- Use small primary key columns, such as TIMESTAMP and INTEGER instead of VARCHAR types.
- Ensure that your system has enough real memory available for all this data. If there is insufficient memory, the operating system uses paging space instead. This decreases indexing and search performance.
- The update `commitcount` parameter, used during the automatic or manual updating of the index, slows down the indexing performance during incremental indexing. Note that the parameter is not used during the initial update process.
- Performance may decrease during index update if many error and warning messages are written to the event log table.

Part 10. Maintaining text indexes

A number of maintenance tasks must be performed to maintain text indexes and obtain useful information about their status.

This section describes how to maintain text indexes and get useful information about their status. The maintenance tasks are:

1. Updating and reorganizing a text index
2. Altering a text index
3. Clearing (deleting) index update event information
4. Dropping a text index
5. Viewing index status

This section also includes information about how to back up and restore indexes and enabled databases.

Chapter 32. Updating and reorganizing a text index

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

If you specify the **`RECREATE`** option in the **`CREATE INDEX`** command, the index is rebuilt completely for each update. This option creates no log tables or triggers. Use this option with care if you have large tables because rebuilding the complete index can be costly.

If the text index was created without the **`RECREATE INDEX ON UPDATE`** option, information about new, changed, or deleted documents is stored through triggers in a log table. If the text index was configured with the extended text-maintained staging infrastructure (AUXLOG ON), an update trigger stores information about changed documents in the log table, while information about inserts and deletes is stored in the auxiliary staging table through integrity processing.

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

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

You should plan periodic indexing carefully; to index large amounts of text documents can be a time- and resource-consuming task. The time it takes depends on many factors. These include the size of the documents, how many text documents have been added or changed since the previous index update, and how powerful the processor is. Two important tips for planning index updates are:

- When working in a partitioned database environment, pay extra attention to the index update schedule to minimize the number of concurrent index updates. This is because for scheduled updates of text indexes in the system, several index updates running concurrently can lead to as many processes per partition as the number of indexes being updated at a given time.
- You should avoid the combination of a large number of indexes and very high automatic update frequencies, as this might result in deadlock situations. For example, 100 indexes with their update frequency set to every 5 minutes, 24 hours a day, and 7 days a week, generates an internal list of 100*12*24*7=201600 checkpoints in a week that must be administered.

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

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

When a database table is updated, the changes that must be made to the Net Search Extender index are logged in a log table. After these log table entries have been processed, the entries are deleted from the log table. If these delete operations to the log table coincide with updates in the database table that need to be logged, a deadlock situation might result.

If the text-maintained staging table infrastructure is configured for the index, certain database operation on the base table might block access to the auxiliary staging table. Make sure the auxiliary staging table is not in a pending mode before updating the text index.

Chapter 33. Updating a text index

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

About this task

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

Command
UPDATE INDEX

Authorization
The privileges held by the authorization ID of the statement must include at least one of the following:
- CONTROL privilege on the table on which the index is defined
- DATAACCESS authority

The following command updates the index:

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

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

If you specify **AUTOMATIC REORGANIZE** during **CREATE INDEX**, the index will be automatically reorganized when necessary.

If you instead specify **MANUAL REORGANIZATION** and want to determine if manual reorganization is necessary, query the db2ext.textindexes view by using the following command:

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

If you specify **MANUAL REORGANIZATION** and often update a column, bear in mind that the update process becomes slower. To manually reorganize, use the following command:

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

Chapter 34. Altering a text index

Issue the **ALTER INDEX** command when the update frequency or index and work directories have to be changed.

Before you begin

The privileges held by the authorization ID of the statement must include at least one of the following:

- CONTROL privilege on the table on which the index is defined
- DBADM authority

About this task

Use the **ALTER INDEX** command to change the index work directory, the update frequency of an index, or the cache characteristics, principally the **MAXIMUM CACHE SIZE** or **PCTFREE**. If you do not specify an update frequency, the current settings are left unchanged. If an index update or search is running, an error message displays. This states that the index is currently locked and no changes can be made.

Example

The following example changes the update frequency for the index.

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

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

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

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

If you change the index directories using the **ALTER INDEX** command, the index files are moved from the original index directory to the new location, and the index is locked during this process. For large indexes, and changes across file systems, this can take considerable amounts of time. After the copying process has finished, the index is unlocked and can be used again.

Chapter 35. Clearing index events

Issue the **CLEAR EVENTS FOR INDEX** command when you no longer need the messages in an index's event table.

Before you begin

The authorization ID of the statement must include at least one of the following privileges:

- CONTROL privilege on the table on which the index is defined
- DBADM authority

About this task

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

Example

The following example deletes messages from the specified text index:

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

Chapter 36. Validating a text index (Windows, AIX)

Starting with DB2 Version 10 Fix Pack 1, use the Net Search Extender index validation utility (the **checknseindex** command) to ensure that a text index is working correctly.

Before you begin

Ensure that update or delete operations are not in progress for the text index that you are validating.

About this task

You must check text indexes for probable corruption if any the following scenarios arise:

- Search queries return errors but do not have specific error codes or explanations.
- Search results do not reflect newly added or updated text documents.
- An index update fails.
- Kernel error messages are in the event log.

Procedure

To validate a text index issue the **checknseindex** command with at least the **-i** and **-p** parameters. This utility uses a significant amount of system resources and might take a long time to run, depending on the size of the text index.

Results

Example 1: Use the **checknseindex** command to verify the status of a valid text index.

```
C:\SQLLIB\bin\\checknseindex.exe -p "C:\litu_ict\Corrupted Index\NODE0000"
-i IX335811

CTE5265I The NSE index validation utility found the specified index,
"IX335811", to be valid.
```

No further action is required.

Example 2: Use the **checknseindex** command to verify the status of a valid text index that has an incorrect path:

```
C:\SQLLIB\bin\\checknseindex.exe -p "C:\litu_ict\Corrupted Index\NODE00001"
-i IX335811

CTE5254E  The NSE index validation utility failed to validate the specified
index because the utility could not access the
path "C:\litu_ict\Corrupted Index\NODE00001".
```

Fix the index path, and reissue the command again.

Example 3: Use the **checknseindex** command with the **-deepCheck** and **-v** parameters to verify the status of an invalid text index:

```
C:\SQLLIB\bin\checknseindex.exe -p "C:\litu_ict\Corrupted Index\NODE0000"
-i IX335812 -deepCheck -v


==================================================

  checknseindex
  -------------

  Net Search Extender index validation utility

==================================================

Validating NSE mapping indexes...

    Forward mapping data control record : Number of blocks = "1"
    Reverse mapping data control record : Number of blocks = "1"
    Forward mapping index control record : Number of documents = "8"
    Reverse mapping index control record : Number of documents = "8"

Validating NSE internal indexes...

Fri Mar 30 13:39:40 2012 Verify primary index...
          100%
Fri Mar 30 13:39:40 2012 Verify secondary index...
          100%
Fri Mar 30 13:39:40 2012 End

Validating consistency of NSE mapping and NSE internal indexes...

CTE5263E  The NSE index validation utility found the NSE mapping index
and the NSE internal index to be inconsistent.  Reason code: "1".
Diagnostic data: "".
```

Attempt to fix the corrupted index.

What to do next

- If an index is corrupted and does not occupy several gigabytes of hard disk space, then re-create the index. For more information, see the topic about creating a text index
- If the invalid index occupies several gigabytes of hard disk space, call IBM customer support for further assistance.
- If you backed up the Net Search Extender text indexes and are using a document management system, follow the corresponding Net Search Extender text index recovery procedure.

checknseindex command (Windows, AIX)

Starting with DB2 Version 10.1 Fix Pack 1, detects corruption or other abnormalities in Net Search Extender text indexes. This command can check the consistency between mapping indexes and internal indexes to ensure that data is being cataloged correctly.

Authorization

- The authorization ID must have at least read permission for the text index directory.
- An user who can perform operations at the database instance level has the required privileges to run this utility.

Syntax

```
►►──checknseindex──┬───────────────────────────────────┬──┬────┬──────────────────►
                   ├─┤ index-path-clause ├─────────────┤  └─-v─┘
                   ├─┤ index-database-clause ├─────────┤
                   └─┤ table-column-database-clause ├──┘

►──┬────────────┬──┬────┬──────────────────────────────────────────────────────────►◄
   └─-deepCheck─┘  └─-h─┘
```

index-path-clause:

```
├──-p──absolute_index_path──-i──index_id──────────────────────────────────────────┤
```

index-database-clause:

```
├──-indschema──index_schema_name──-indname──index_name──-dbname──database_name────┤
```

table-column-database-clause:

```
├──-tabschema──table_schema_name──-tabname──table_name──-colname──column_name──────►

►──-dbname──database_name─────────────────────────────────────────────────────────┤
```

Parameters

-p *absolute_index_path*
: Specifies the absolute path of an index directory.

-i *index_id*
: Specifies ID of the index to verify.

-indschema *index_schema_name*
: Specifies the index schema for which NSE index is validated.

-indname *index_name*
: Specifies the index name for which NSE index is validated.

-dbname *database_name*
: Specifies the database name in which NSE index is created.

-tabschema *table_schema_name*
: Specifies the table schema for which NSE index belongs to and requires validation.

-tabname *table_name*
: Specifies the table name for which NSE index belongs to and requires validation.

-colname *column_name*
: Specifies the column name of table on which NSE index is created and requires validation.

-v
: Displays verbose tool output.

-deepCheck
: Checks both Net Search Extender mapping indexes and internal indexes to verify that they are in sync. If you run the utility without the **`-deepCheck`**

parameter and the utility reports that the indexes are valid but one of the following conditions applies, rerun the utility with the **-deepCheck** parameter:

- Search queries return errors and do not have specific error codes or explanations.
- Search results do not reflect newly added or updated text documents.
- The index update fails.
- Kernel error messages are in the event log.

If you run the utility without the **-deepCheck** parameter and the utility reports that the indexes are invalid, you do not haveto rerun the utility with the **-deepCheck** parameter.

-h Displays help for the utility.

This utility uses a significant amount of system resources and might take a long time to run, depending on the size of the text index.

Example

Example 1: Issue the **checknseindex** command to check the status of a valid text index:

```
C:\SQLLIB\bin\\checknseindex.exe -i IX335811 -p "C:\myTextIndexes\NODE0000"
CTE5265I The NSE index validation utility found the specified index, "IX335811",
to be valid.
```

Usage notes

- The Net Search Extender index validation utility is in the `SQLLIB/bin` directory. To run the utility on Windows operating systems, use **checknseindex.exe**. On AIX operating systems, use **checknseindex**.
- Modify the following sample SQL statement to obtain the ID of an index on your system:

```
select a.indexidentifier, indexdirectory
from db2ext.ttextcolumns a, db2ext.ttextindexes b
where a.indexidentifier = b.indexidentifier
and a.tablename = 'MYTAB'
and a.schemaname = 'MYSCHEMA' and a.columnname ='MYDATA'"
```

- For a partitioned environment, run the utility on all of the corresponding nodes.

Chapter 37. Dropping a text index

Issue the **DROP INDEX FOR TEXT** command when you no longer intend to make text searches on a text column.

Before you begin

The authorization ID of the statement must include at least one of the following privileges:
- CONTROL privilege on the table on which the index is defined
- DBADM authority

Example

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

When dropping a text index, you also drop the following tables and views:
- The log table and view of the index
- The event table and view of the index
- The log table triggers (if present)
- The text-maintained staging table and view (if present)

Note: Always drop the indexes on the table before dropping the table. If you drop the table first, any text-maintained staging table (if present) is dropped as well, but the indexes with their administrative tables and views still exist.

Chapter 38. Viewing text index status

To get information about the current text indexes within the database, use the Net Search Extender catalog views.

Example

For example, if you want to know about the current database defaults, use the following command:

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

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

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

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

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

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

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

Chapter 39. Backing up and restoring text indexes

You must stop the Net Search Extender services before backing up or restoring enabled databases and text indexes.

Procedure

- To back up enabled databases and text indexes created by Net Search Extender:
  1. Find out which indexes Net Search Extender has created and where they are stored. Call a select statement on the db2ext.textindexes view:
     ```
     db2 "select indschema, indname, indexdirectory from db2ext.textindexes"
     ```
  2. Ensure that no index update is running, and then stop Net Search Extender services. Issue the following command:
     ```
     db2text stop
     ```
  3. After backing up the database, back up the index directories and subdirectories.
  4. Restart Net Search Extender services. Issue the following command:
     ```
     db2text start
     ```
- To restore the enabled databases and text indexes created by Net Search Extender:
  1. Stop Net Search Extender. Issue the following command:
     ```
     db2text stop
     ```
  2. Restore the backup copies of the index directories to the same path as before.
  3. Restart Net Search Extender. Issue the command:
     ```
     db2text start
     ```

Chapter 40. Removing files from the /tmp directory

Certain files must not be deleted from the `/tmp` directory while the Net Search Extender services are running.

While the Net Search Extender services are running, make sure that no scheduled jobs that clean `/tmp` are accidentally removing these files. The following files must exist in the `/tmp` directory and must not be deleted while the Net Search Extender services are running:

- Semaphore and shared memory files:

  ```
  instance_owner.TEXT.0000.LATCH
  instance_owner.TEXT.0000
  instance_owner.CACHE.0000
  instance_owner.SCHEDULER.LATCH
  instance_owner.DEMON.SEM
  instance_owner.DEMON.MEM
  ```

  Note: In a partitioned database environment, there would be additional, similarly named files corresponding to each node: *`instance_owner`*`.TEXT.0001.LATCH`, *`instance_owner`*`.TEXT.0001`, *`instance_owner`*`.CACHE.0001`, and so on.

- During create index, if the cache is temporary, you might see files in `/tmp` similar to the following:

  ```
  database_name.IX123456
  database_name.IX123456.data0
  ```

Part 11. Methods for searching text

The Net Search Extender allows SQL scalar search functions, a stored procedure search function, and an SQL table-valued function to search text.

Net Search Extender provides the following methods for searching text:

SQL scalar search functions
: Text search sub-queries can be embedded in SQL queries. Net Search Extender provides the SQL scalar search functions as extension to the available SQL functions. By including text search sub-queries in SQL queries, it is possible to combine Net Search Extender search functionality with DB2 XQuery processing. Text search queries on XML documents can be used in the `db2-fn:sqlquery()` XQuery input function, and allow for a direct processing of the resulting XML documents with XQuery.

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

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

For SQL scalar search functions, this section describes the following areas:

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

  Refer to Chapter 60, "SQL scalar search function and the SQL table-valued function," on page 253 for a detailed description of the syntax.
- Specifying search arguments by using examples with the CONTAINS function.

  Refer to Chapter 59, "Syntax of search arguments," on page 245 for a complete description of the syntax.

For the stored procedure search function, this section describes the following areas:

- Searching for text using the stored procedure search.
- For specifying search arguments, refer to Chapter 59, "Syntax of search arguments," on page 245 for a description of the parameters.

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

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

  Refer to Chapter 60, "SQL scalar search function and the SQL table-valued function," on page 253 for a description of the syntax.
- For specifying search arguments, refer to Chapter 59, "Syntax of search arguments," on page 245 for a description of the parameters.

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

Before searching, ensure that all the appropriate indexing steps, described in Part 9, "Developing: creating and maintaining a text index," on page 87, involving the different data types have been performed.

Chapter 41. Searching for text using SQL scalar search functions

There are multiple ways to use the SQL scalar search functions by using the `CONTAINS`, `NUMBEROFMATCHES`, and `SCORE` functions.

Using examples, this section describes how to use the SQL scalar search functions in the following ways:
- Using the function `CONTAINS` to issue a query.
- Using the function `NUMBEROFMATCHES` to determine how many matches of the search term are found in a text document.
- Using the function `SCORE` to obtain the relevancy of a found text document.

Refer to Chapter 60, "SQL scalar search function and the SQL table-valued function," on page 253 for a description of the syntax.

Issuing a query

Issuing a query of the CONTAINS function searches for text in columns found in tables.

Example

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

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

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

Searching for "" is not supported. Using two consecutive quotation marks in a search term will result in a syntax error message. Also, a query syntax error occurs if a newline character is used within the search string.

Note:

If you know that the text search alone will return a very large result set, it is beneficial to add restrictive search criteria, for example:

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

Searching and returning the number of matches found

You can use the NUMBEROFMATCHES function to determine how often a search term occurs in a document.

About this task

The NUMBEROFMATCHES function can be used as a conditional statement to find rows that have at least one occurrence of a term. The following query only returns the rows that have at least one occurrence of the word "book".

```
SELECT AUTHOR,TITLE,NUMBEROFMATCHES(COMMENT,'"book"')
        FROM DB2EXT.TEXTTAB WHERE
    NUMBEROFMATCHES(COMMENT, '"book"') > 0
```

NUMBEROFMATCHES returns an integer value for every row.

Searching and returning the score of a found text document

SCORE returns a positive number that indicates how well the document meets the search term relative to other found documents in the same index. The value is calculated based on the number of matches that are found in the document in relation to the document's size.

Example

In the following example, you can get the score of a found document by using the SCORE function:

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

SCORE returns a DOUBLE value between 0 and 1.

The values returned by SCORE are only meaningful if they are compared to other SCORE values returned for the same index. The values cannot be compared to scores returned for other indexes.

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

The SCORE values are different depending on the DB2 database environment:

- In a non partitioned database environment, all the documents are in a single table. The SCORE value is based on a single table, and a document's relationship to all the other documents in the table.
- In a partitioned database environment, all the documents are located on different partitions. During indexing, only the documents that are local on every partition are used to build the text indexes. In this case, the SCORE value is based on the documents relationship to all documents in only one of the multiple partition.

Chapter 42. Specifying SQL search arguments

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

Refer to "Search argument syntax" on page 245 for a complete description of the syntax.

Searching for terms in any sequence

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

Example

For example:

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

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

Searching with the Boolean operators AND and OR

You can combine search terms with other search terms using the Boolean operators "&" (AND) and "|" (OR).

About this task

In this example, you are looking for the terms "author" and "pulitzer". Search results that match either of these terms are returned.

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

You can also combine several terms by using Boolean operators:

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

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

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

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

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

If you want to enforce a different evaluation order of the boolean operators, you must include parentheses:

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

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

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

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

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

Searching with the Boolean operator NOT

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

Example

For example:

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

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

Fuzzy search

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

Example

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

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

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

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

Searching for parts of a term (character masking)

Masking characters, otherwise known as "wildcard" characters, offer a way to make a search more permissive. They increase the number of text documents that are found by a search.Net Search Extender uses two masking characters: percent (%) and underscore (_).

About this task

Net Search Extender uses these masking characters the same way the DB2 predicate LIKE uses them.

- % represents any number of arbitrary characters.

  Here is an example of % used as a masking character in the middle of a search term:

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

  This search term finds text documents containing the word "thriller", "throttle", and "thread-splitter".
- _ represents one character in a search term.

  The following example also finds text documents containing the word "thriller".

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

You can use more than one wildcard character in a phrase (more than one word in the phrase can contain a wildcard), however the terms resulting from the wildcard expansion can only be single terms, not multi-word terms. For example, the wildcard expression "th%er" will not match the phrase "the caller".

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

Note that you cannot combine fuzzy search or thesaurus search with wildcard character search. Also, make sure that the wildcard characters are expandable within the implicit or explicit EXPANSION LIMIT *number* search parameter. For details on EXPANSION LIMIT *number* search parameter see, “Search parameters” on page 248

Searching for terms that contain a masking character

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

Example

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

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

Searching for terms in a fixed order

You can search for terms in a fixed order if you search for "primary key".

Example

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

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

Searching for terms in the same sentence or paragraph

Net Search Extender has a limited capability to search for terms in the same sentence or paragraph.

Example

Here is an example of a search argument that finds text documents in which the search term "web" occurs in the same sentence as the term "disk":

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

You can also search for several words occurring together. In the next example, two phrases are searched for that occur in the same paragraph:

```
SELECT AUTHOR, TITLE
        FROM DB2EXT.TEXTTAB
        WHERE CONTAINS(COMMENT,
      '"linguistic analysis processing" IN SAME PARAGRAPH AS
             "search algorithms"') = 1
```

Searching for terms in sections of structured documents

If you have a document that is divided into sections, you can specify an index search in one or more sections instead of the entire document.

Example

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

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

Note that section names are case-sensitive. Ensure that the section name in the model file and in the query are identical.

Thesaurus search

Thesaurus search is a powerful search-term expansion function in Net Search Extender. The additional terms you search for are taken from a thesaurus that you build yourself, so you have direct control over the terms.

About this task

For example, a thesaurus search for "database", might find terms like "repository" and "DB2" if you decide that these terms are related.

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

Example

The following examples demonstrate the syntax for using thesaurus expansion.

This example takes the term "product" and expands it, adding all related terms of this term found in the thesaurus "nsesamplethes".

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

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

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

Numeric attribute search

You can search for numeric arguments in a text file using NetSearch Extender, such as ranges and boolean conditions.

About this task

You can search on numeric attributes that are stored in a the text index using the following syntax:

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

Free-text search

Free-text search is a search in which you express the search term as free-form text. A phrase or a sentence describes in natural language the subject to be searched for.

About this task

The sequence of words in a free-text query is not relevant. However, at least one of the query terms in the free-text query must occur in the documents to be searched.

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

Example

For example:

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

Chapter 43. Additional search syntax examples

To become familiar with additional search syntax examples, run the **search** script located in the sqllib/samples/extenders/db2ext/ directory.

About this task

This contains examples of Net Search Extender search functions that run against the sample table.

Enter the command as follows:

```
db2 -tvf search
```

There is also a sample that shows you how you can query XML data. After you have connected to the database, you can perform searches on the data by issuing **db2 -tvf xmlsearch**

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

- On UNIX operating systems: **nsesample** in the *instance_owner_home*/sqllib/samples/extenders/db2ext directory.
- On Windows operating systems: nsesample (.bat) in the *sqllib*\samples\extenders\db2ext directory.
- For XML searches, call xmlsample (.bat)*database* to populate the database, and create and update the indexes.

Chapter 44. Searching for text using a stored procedure search

Use the stored procedure search interface if you only need a ranked subset of the text search results, and high query performance.

Do not use the stored procedure if you require all the search results, or you need to index a large number of documents. The main reason for this is that parts of the user table are copied into memory and so a lot of real memory needs to be available.

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

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

- The cache-search-result options have been specified during **CREATE INDEX**.
- The present and future shared memory requirements, possibly involving incremental updates, have been fully considered.
- The cache of the index has been activated using the **db2text activate** command.
- In a partitioned database environment, to be able to use the stored procedure search, the table must use a table space on a single partition and the procedure should be called on the same partition. Otherwise the search will be disallowed and errors will be returned.

The following is an example of a stored procedure search:

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

The first parameter is the search term. The syntax for the search term is the same as in the SQL scalar functions. The next parameters are the index schema and index name. If you have not enclosed the name in quotation marks, it is translated to uppercase. The following two numeric arguments give you the result slice starting point and the number of results in the slice. The next two integer values specify if score and hit information are requested. The final two values are the function return values.

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

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

Chapter 45. Searching for text using an SQL Table-Valued Function

Use the SQL Table-Valued Function if you do not need all of the search results, and if you do not have enough memory to use a cached index as used in the stored procedure search.

There are two SQL table-valued functions available, both called `db2ext.textsearch`. One of them has two additional parameters for use with the `db2ext.highlight` function.

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

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

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

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

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

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

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

Note:

- Only a single primary key column is allowed.
- In a partitioned database environment, to be able to search using the SQL Table-Valued Function, the table must use a table space on a single partition and the function should be called on the same partition. Otherwise the search will be disallowed and errors will be returned.

Using the highlight function

To use the SQL table-valued db2ext.highlight function, you must use the db2ext.textsearch function with the additional **`numberOfHits`** and **`hitInformation`** parameters.

About this task

The highlight function must not be used if you created an index using a transformation function whose implementation might change, else the returned highlight position information might not be correct because of a mismatch between positional information at the indexing time and at the search time.

Example

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

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

The query returns the following result:

```
DOCID HIGHLIGHT

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

1 record(s) selected.
```

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

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

The search argument returns the following result:

```
DOCID HIGHLIGHT

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

1 record(s) selected.
```

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

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

The search argument returns the following result:

```
DOCID HIGHLIGHT

2     York Times <bf>bestseller</bf> about <bf>peacekeeping</bf> ...
```

```
      <bf>peacekeeping</bf> <bf>soldiers</bf> called "Keepers" ... the
      worlds <bf>attention</bf> after their

1 record(s) selected.
```

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

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

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

Chapter 46. Searching on more than one column

In cases where you need to create a text index on more then one column, the easiest way is to use the SQL scalar function and combine the searches on those columns.

Example

You can see this in the following example:

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

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

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

Chapter 47. Using text search in outer joins

If you use an outer join query that uses the CONTAINS() search function, the query might fail and result in the reason code `CTE0129 NULL values are not allowed to be passed as parameters`.

The CONTAINS() predicate must reference the column of a table on the tuple preserving side of the outer join.

For example, T1 is the tuple-preserving side in 'T1 left outer join T2' and T2 is the tuple-preserving side in 'T1 right outer join T2'.

Chapter 48. Performance considerations during search

To enhance performance during search, consider issues when searching within SQL, with the stored procedure, or if you use the NUMBEROFMATCHES or the SCORE function without the CONTAINS function.

To enhance performance during search, consider the following issues:

- When searching within SQL:
  - If you notice a decrease in performance, Use the `explain` statement to check the processing plan of the DB2 Optimizer.
  - Parametric search can make searching faster, especially if you use other search predicates to reduce the result size.
  - Use the result limit keyword if you do not require all of the results.
- When searching with the stored procedure:
  - As the specified cache table expression is copied from the database into memory, ensure that your workstation has enough memory available for this data. If there is insufficient memory, paging space is used, which decreases search performance.
- If you use the NUMBEROFMATCHES or the SCORE function without the CONTAINS function, query performance may decrease. Also, to avoid duplicate processing, ensure that the string in the CONTAINS function exactly matches the string used in NUMBEROFMATCHES or SCORE function.

Chapter 49. User scenarios

The Net Search Extender can be used when running an SQL scalar search, stored procedure search or SQL table-valued function search.

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

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

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

The SQL table-valued function example
The SQL table-valued function example is a variant of the stored procedure search example.

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

Simple example with the SQL scalar search function

You can use the SQL scalar search functions, such as CONTAINS, NUMBEROFMATCHES, and SCORE, to search text indexes using AND, OR, and boolean operators.

Example

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

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

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

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

```
db2 create database sample
```

Enabling a database for text search
You can issue DB2 Net Search Extender commands in the same way as

DB2 commands on the command line of the operating system. For example, use the following command to start Net Search Extender Instance Services:

```
db2text START
```

Then prepare the database for use with DB2 Net Search Extender:

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

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

Creating a table

```
db2 "CREATE TABLE books (isbn VARCHAR(18) not null PRIMARY KEY,
      author VARCHAR(30), story CLOB(100k), year INTEGER)"
```

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

Creating a full-text index

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

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

Loading sample data

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

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

Synchronizing the text index

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

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

Searching with the text index

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

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

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

This query searches for all books containing the term `cat` where the `year` value of the book is greater or equal to `2000`. The query returns the following result table:

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

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

Simple example with cache usage and stored procedure search

You can create a text index in the cache to use with the TEXTSEARCH stored procedure. The search results available are dependent on the cache size, but the time required to use run stored procedures is reduced.

Example

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

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

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

Creating a text index with cache option

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

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

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

Synchronizing the index and activating the cache

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

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

To activate the cache, use the following command:

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

This loads the content of the columns `author` and `story` into the cache.

Searching with the TEXTSEARCH Stored Procedure

You can only use the DB2 Net Search Extender stored procedure in certain cases.

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

This query searches for all books about a `cat`, but only returns the first two results. The result table for a book might be as follows:

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

AUTHOR    STORY
Mike      The cat hunts some mice.

Return Status = 0
```

For more samples about the search syntax, check the following file in the DB2 instance directory: `sqllib/samples/extenders/db2ext/search`

Simple example with the SQL table-valued function

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

About this task

The SQL table-valued function query corresponds to the previously used CONTAINS query. See "Synchronizing the text index" under the topic “Simple example with the SQL scalar search function” on page 153 for information.

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

In this example, NULL is cast to the data type of the primary key.

Chapter 50. Using a thesaurus to expand search terms

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

A thesaurus is a controlled vocabulary of semantically related terms that usually covers a specific subject area.

Net Search Extender lets you expand a search term by adding additional terms from a thesaurus that you have previously created. Refer to Chapter 59, “Syntax of search arguments,” on page 245 to find out how to use thesaurus expansion in a query.

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

This section describes:

- **“The structure of a thesaurus”**

  A thesaurus is structured like a network of nodes linked together by relations. This section describes Net Search Extender's predefined relations and how to define your own relations.
- **“Creating and compiling a thesaurus” on page 159**

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

The structure of a thesaurus

A thesaurus is structured like a network of nodes linked together by relations.

Net Search Extender looks up a term in a thesaurus by starting at the term, then following a path through the term relations and delivering the terms found in the process.

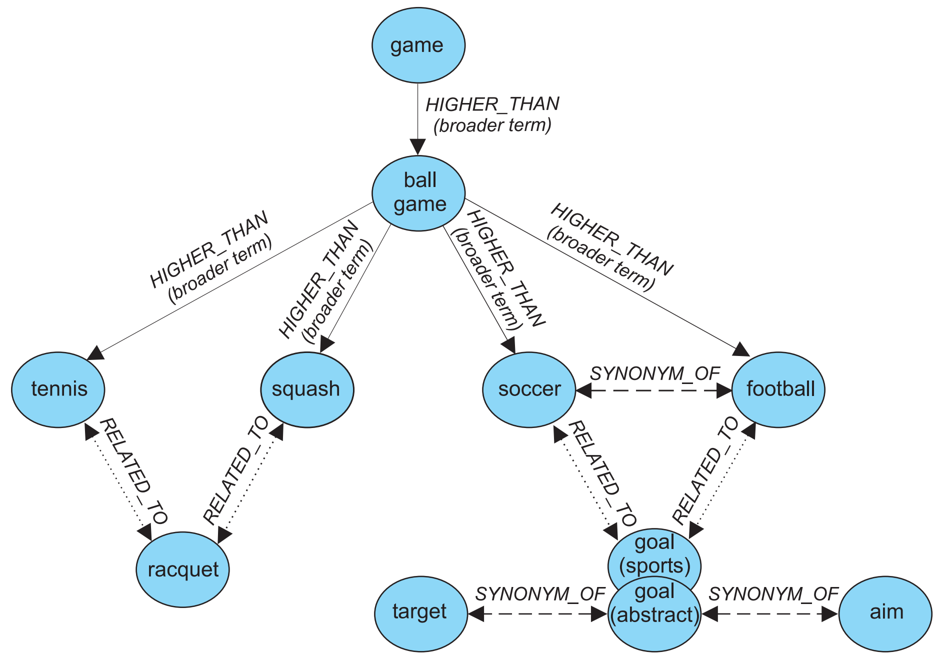


Figure 7. An example of the structure of a thesaurus

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

Predefined thesaurus relations

The Net Search Extender contains predefined relations which include associative relations, synonym relations, and hierarchical relations.

- **Associative relations**

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

  Predefined associative relation: `RELATED_TO`

  Examples:

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

- **Synonym relations**

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

  Predefined synonym relation: `SYNONYM_OF`

  Examples:

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

  The figure in Figure 7 shows two `goal` terms in the same thesaurus. One is specified with the comment `(sports)`, the other with the comment `(abstract)`. Even if terms have the same spelling, synonym relations can connect different word groups. You can model this by using different relations when defining the thesaurus.

- **Hierarchical relations**

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

Predefined hierarchical relations:

- `LOWER_THAN` to model narrowing relations

  `LOWER_THAN` relations are for modelling a sequence of more specialized terms. The deeper you follow a narrowing relation, the more specific the terms become. For example, if you look up the term `ball game` along a `LOWER_THAN` relation, the result could be `squash tennis` and so on, in a list of increasingly specialized terms.
- `HIGHER_THAN` to model broadening relations

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

Defining your own relations

Net Search Extender lets you define your own `RELATED_TO`, `LOWER_THAN`, and `HIGHER_THAN` thesaurus relations.

Because each relation name must be unique, you must qualify such relations names by the addition of a unique number, like this: `RELATED_TO(42)`.

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

Creating and compiling a thesaurus

There are steps that must be followed when creating a thesaurus that can be used by the Net Search Extender functions

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

Creating a thesaurus definition file

There are certain restrictions that apply when creating a thesaurus definition file.

About this task

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

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

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

`sqllib``\db2ext\thes`

On UNIX systems, the thesaurus directory is:

instance_owner_home/sqllib/db2ext/thes

Here are the first few definition groups from that file:

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

:WORDS
   acoustics
 .RELATED_TO signal processing

:WORDS
   aeronautical equipment
 .SYNONYM_OF turbocharger
 .SYNONYM_OF undercarriage

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

Figure 8. An extract from the sample thesaurus definition file

For the syntax of each definition group, see "Thesaurus support."

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

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

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

Compiling a definition file into a thesaurus dictionary

To compile a thesaurus definition file, run the **db2extth** command.

About this task

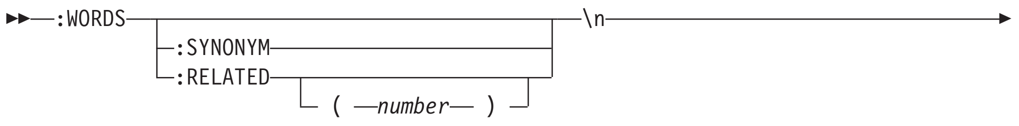
To use a thesaurus dictionary within a partitioned environment, ensure that all the physical nodes can access the created files.

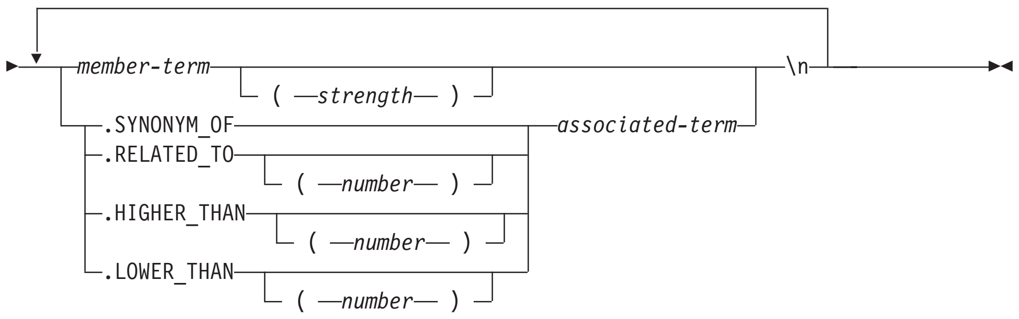
Thesaurus support

When creating your own thesaurus, specific syntax must be used.

Here is the syntax of each definition group when you create your own thesaurus:

Syntax of a thesaurus definition





Note that `\n` is not part of the syntax, but represents the end of a line in the thesaurus definition file.

You can insert comment lines in a thesaurus definition file like this:

```
# my comment text
```

:WORDS
A keyword that begins a group of related words.

:SYNONYM, :RELATED [(*number*)],
A relation name.

Relation names consist of a relation type and a number. If the number is omitted, zero is assumed, which is the system-provided relation name. :SYNONYM is always the system-provided relation name.

Relation names that begin with a colon, such as :SYNONYM, precede a list of words that are related to each other by the same relation. For example:

```
:WORDS
 :SYNONYM
   air steward
   cabin staff member
   flight attendant
```

member-term
A term to be included in the thesaurus dictionary.
- Maximum length is 64 bytes (42 bytes for code page UTF-8).
- Single-byte characters and double-byte characters of the same letter are regarded as the same.
- Uppercase and lowercase characters are not distinguished.
- A term can contain a blank character.
- The single-byte character period "." or colon ":" cannot be used.

This parameter can be useful if you do not want a thesaurus lookup to include words that have a weak relation to the looked up term. Strength is a numeric value from 1 to 100. The default value is 100.

.SYNONYM_OF, .RELATED_TO [(*number*)], .HIGHER_THAN [(*number*)], .LOWER_THAN [(*number*)]

A relation name. The relation name .HIGHER_THAN corresponds to the BROADER query relation and .LOWER_THAN to the NARROWER query relation. Relation names consist of a relation type and a number. If the number is omitted, zero is assumed, which is the system-provided relation name. The relation name .SYNONYM is always the system-provided relation name.

Relation names that begin with a period, such as .SYNONYM_OF, define the relation between one word and another. For example:

```
:WORDS
   air steward
 .SYNONYM_OF cabin staff member
 .SYNONYM_OF flight attendant
```

The optional *number* identifies a user-defined relation. This must be a unique number from the whole thesaurus definition file (currently 1 to 128). For example: `RELATED_TO(42)`.

If you want to use symbolic names for thesaurus relations in your application instead of the relation name and number, your application must handle the name-to-number mapping. For example, if you define the relation `opposite_of` as RELATED_TO(1), your application must map this name to the internal relation name RELATED_TO(1).

associated-term

Each associated term must be preceded by the relation name. The associated term is related to each member term with respect to the specified relation. If all member terms are related to each other, this can be specified using a member relation.

- Maximum length is 64 bytes (42 bytes for code page UTF-8).
- Single-byte characters and double-byte characters of the same letter are regarded as the same.
- Uppercase and lowercase characters are not distinguished.
- A term can contain a blank character.
- The single-byte character period "." or colon ":" cannot be used.

Here is an example of an associated term:

```
:WORDS:SYNONYM
  reject
  decline
    RELATED_TO(1) accept
```

Thesaurus supported CCSIDs

Coded character set identifiers (CCSIDs) are supported by the thesaurus to enable non-English text searches.

The following CCSIDs are supported by the thesaurus:

819	Latin 1
850	PC Data Latin 1
874	Thai
932	Combined Japanese
943	Combined Japanese
949	Combined Korean

950	Combined Traditional Chinese
954	Japanese
970	Combined Korean
1208	UTF 8
1250	Latin 2
1252	Latin 1
1253	Czech
1254	Turkish
1255	Hebrew
1256	Arabic
1258	Vietnamese
1363	Combined Korean
1381	Combined Simplified Chinese
1383	Chinese (simplified), combined SBCS/DBCS
1386	Chinese (simplified), combined SBCS/DBCS
5039	Japanese (combined SBCS/DBCS)

Messages returned by the thesaurus tool

There are multiple error messages which can be returned when using the thesaurus tool.

ADM_MSG_INVALID_CCSID

Invalid CCSID specified.

The requested code page is not supported.

ITL_THES_MSG_BUFFER_OVERFLOW

Buffer overflow.

ITL_THES_MSG_DICT_EXIST

Thesaurus dictionary *dictionary name* already exists.

Cannot be overwritten.

ITL_THES_MSG_DICT_INTEGRITY_ERROR

Integrity of dictionary *dictionary name* is lost.

The thesaurus dictionary file is corrupted.

ITL_THES_MSG_DICT_NOT_EXIST

Thesaurus dictionary *dictionary name* does not exist.

ITL_THES_MSG_DICT_VERSION_ERROR

Dictionary *dictionary name* version error.

The thesaurus dictionary was created with an incompatible earlier version.

ITL_THES_MSG_ERROR_IN_FILE

Error in file *file name*.

ITL_THES_MSG_FILE_ACCESS_ERROR

Could not access file *file name*.

ITL_THES_MSG_FILE_CLOSE_ERROR

Could not close file *file name*.

ITL_THES_MSG_FILE_EOF_ERROR

Unexpected end of file in *file name*.

Error in definition file.

ITL_THES_MSG_FILE_OPEN_ERROR

Could not open file *file name*.

ITL_THES_MSG_FILE_REACHED_END

Unexpected end of file in *thesaurus definition file*.

There is an error in the definition file.

ITL_THES_MSG_FILE_READ_ERROR

Could not read file *file name*.

ITL_THES_MSG_FILE_REMOVE_ERROR

Could not remove file *file name*.

ITL_THES_MSG_FILE_RENAME_ERROR

Could not rename file *file name 1* to *file name* 2.

ITL_THES_MSG_FILE_WRITE_ERROR

Could not write file *file name*.

ITL_THES_MSG_IE_BLOCK_START

No block starting line was found in file *file name* at line *line number*.

ITL_THES_MSG_IE_EMPTY

The thesaurus definition file *file name* is empty.

ITL_THES_MSG_IE_NO_TERM

No terms are defined in *file name* at line *line number*.

ITL_THES_MSG_IE_REL_SYNTAX

Relationship is specified incorrectly in *file name* at line *line number*.

ITL_THES_MSG_IE_STRENGTH_DOMAIN

Strength is out of range.

Valid values are 1 - 100; the default is 100.

ITL_THES_MSG_IE_STRENGTH_SYNTAX

A strength value is specified incorrectly.

Syntax: After the term, type [:20] for a strength of 20.

ITL_THES_MSG_IE_TERM_LEN

A thesaurus term is longer than 64 characters.

ITL_THES_MSG_IE_USER_DEF

Relationship is specified incorrectly in *file name* at line *line number*.

ITL_THES_MSG_IE_USER_DEF_DOMAIN

A relationship number is out of range in *file name* at line *line number*.

ITL_THES_MSG_INPUT_ERROR

Error in the thesaurus definition file *file name* at line *line number*.

ITL_THES_MSG_INTERNAL_ERROR

Internal error.

ITL_THES_MSG_LOCKED

Thesaurus dictionary *dictionary name* is in use.

ITL_THES_MSG_LOCKING_ERROR

Could not lock dictionary *file name*.

ITL_THES_MSG_MEMORY_ERROR

Memory error.

ITL_THES_MSG_NAMELEN_ERROR

Parameter error *file name*. The thesaurus definition file name is too long.

ITL_THES_MSG_NO_TARGET_DIR_ERROR

Parameter error. No target directory specified.

ITL_THES_MSG_NONAME_ERROR

Parameter error. No thesaurus definition file name specified.

ITL_THES_MSG_NORMALIZE_ERROR

Error in normalizing a term.

Error in the thesaurus definition file.

ITL_THES_MSG_OUTFILE_EXIST

Output file *file name* already exists.

ITL_THES_MSG_PARAMETER_ERROR

Internal parameter error.

ITL_THES_MSG_PATHLEN_ERROR

Parameter error *file name*. The thesaurus definition file path is too long. The path length must not exceed the maximum length supported for directory names in the operating system.

ITL_THES_MSG_UNEXPECTED_ERROR

Internal unexpected error.

Chapter 51. Net Search Extender indexing configuration

There are certain configuration options to modify the indexing and search behavior of Net Search Extender

This chapter provides information about some configuration options to modify the indexing and search behavior of Net Search Extender .
- Tokenization
- Stop words
- Configuration

Tokenization

During indexing, Net Search Extender processes document text in the following way, breaking the text up into tokens.

Words

All alphanumeric characters ("a".."z,"A".."Z", "0".."9") are used to create the full-text index. Separation characters are blank characters and the characters described in the following sentence recognition section. Control characters, such as line feed (also known as a new line character) and blank characters, are interpreted as follows: Control characters (less than 0x20) in the middle of the line are regarded as blank characters. Blank characters and control characters before and after a line feed (0x0A) are ignored. Line feed before and after a 1-byte character are regarded as blank characters and 2-byte characters for the same character are always regarded as the same characters. Capital letters and small letters for the same character, for example, "A" and "a", are regarded as the same characters if nothing is specified during search, or as different characters if exact matching is required during search.

Sentences

Net Search Extender recognizes ".", "!", "?" provided the following conditions are met:
- A special character "." has to be followed by a blank space or a new line to be considered the end of a sentence.
- The characters "!" or "?" anywhere in the text will mark the end of a sentence (even without a space or new line).
- If "!", "?", or "." are contained within quotation marks, they are ignored and are not considered to mark the end of a sentence.

Paragraphs

Paragraph recognition is dependent on the document format. In Plain Text format, any two consecutive new line characters (possibly with an intervening carriage return) are recognized as a paragraph boundary. In HTML, the paragraph tag <p> is interpreted as paragraph boundary. The other document formats do not support paragraph recognition.

Stop words

Stop words are words with a high frequency and no relevant content for the text retrieval process.

Usually, all function words (in a linguistic sense) are considered stop words, for example "and", "or", and "in". Searching an index for stop words can reduce the precision of a text retrieval system significantly.

Net Search Extender provides stop word processing for a list of languages. The configuration parameter **`IndexStopWords`** can be set at index creation time and determines if stop words are indexed or not. The default is 1 meaning that stop words are indexed.

If you do not want to index stop words, you must set **`IndexStopWords`** to `0`, and specify the language of your input documents using the language parameter during create index. If stop words are not indexed, the index is smaller and faster. Do not alter this value in the configuration `.ini` file template after you have created the index, as this leads to documents being treated differently depending on the time that they were indexed and consequently results in incoherent stop word handling.

Ignoring stop words during indexing is effective only if all of the documents in your collection are in the same language.

Languages supporting stop words

Stop words are high frequency words that do not provide useful content, such as "and", "or", and "in". Some languages supported by NetSearch Extender can choose not to index stop words using `IndexStopWords`.

The following languages provide stopword processing.

AR_AA	Arabic as spoken in Arabic countries
CA_ES	Catalan as spoken in Spain
DA_DK	Danish as spoken in Denmark
DE_CH	German as spoken in Switzerland
DE_DE	German as spoken in Germany
EL_GR	Greek as spoken in Greece
EN_GB	English as spoken in the U.K.
EN_US	English as spoken in the U.S
ES_ES	Spanish as spoken in Spain
FI_FI	Finnish as spoken in Finland

FR_CA
French as spoken in Canada

FR_FR
French as spoken in France

HE_IL Hebrew as spoken in Israel

IS_IS Icelandic as spoken in Iceland

IT_IT Italian as spoken in Italy

IW_IL Hebrew as spoken in Israel

NB_NO
Norwegian Bokmal as spoken in Norway

NL_BE
Dutch as spoken in Belgium

NN_NO
Norwegian Nynorsk as spoken in Norway

PT_BR
Portuguese as spoken in Brazil

PT_PT
Portugese as spoken in Portugal

RU_RU
Russian as spoken in Russia

SV_SE
Swedish as spoken in Sweden

Configuration

Net Search Extender is able to search for words which may have characters used in different combinations, for example, alphanumerics, numbers, and special characters.

To do this, Net Search Extender provides the following configurations:

Character normalization
Character normalization ensures that words that can be written in two ways can be both searched for. For example, the German word 'Überbau' can also be written as 'Ueberbau'. Normalization ensures that both words can be searched for, by using either 'Überbau' or 'Ueberbau'. The functionality also normalizes accented letters, for example, 'accès' to the matching simple character, for example, 'acces'. Note that the use of this option can have undesired results in languages where for example the character 'Ü' does not have an equivalent standard normalization as 'Ue'

Using specific characters as part of a word
Using specific characters as part of a word ensures that product names which can involve a series of alphanumeric characters, special characters and numbers can be searched on as a single word. For example, by treating the alphanumeric combination 'DT9' as one word, or by enabling the '/' special character, so that OS/390® are searched on as whole words rather than as 'OS' and '390'.

For these configuration settings, switches are available. To customize the switches, change the `.ini` file template before creating an index.

The `.ini` file template is stored in `sqllib/db2ext/cteixcfg.ini`. As you can also make changes to most of the values in this template file using the **CREATE INDEX** command, it is recommended that you only change the following values:

```
AccentRemoval (for character normalization)
UmlautNormalization (for character normalization)
TreatNumberAsWords (for treating numeric characters as part of the word)
AdditionalAlphanumCharacters (for using specific characters as part of a word)
```

AccentRemoval
: This parameter specifies if accented characters are normalized to the matching simple character. For example, événement is also indexed as evenement. The default is true.

UmlautNormalization
: This parameter specifies if an umlaut character is also indexed as two characters with the same meaning. For example, 'Übersee' is also indexed as 'Uebersee'. The default is true.

TreatNumbersAsWords
: This parameter specifies if numeric characters next to a word are part of the word. For example, 'DT9' is treated as one word and not as one word 'DT' and the number '9'.

AdditionalAlphanumCharacters
: The string value of this parameter defines which characters are treated as part of a word. The string of special characters must be a sequence of one or more characters in UTF-8. The default string contains the characters "/-@".

  You are not allowed to use the wildcard characters % and _ in the list of characters that are treated as being part of a word. This results in problems during query execution.

If you want to change any of these configuration values, edit the `.ini` file before you create your index. To activate inactive switches, remove the comment marker ";" from the beginning of the line. For further information, see the `cteixcfg.ini` file.

You are recommended not to alter any of the other values in the `.ini` file.

Part 12. Working with structured documents

Net Search Extender allows you to index and search text or numeric fields, such as title, author, or price in a structured document.

The documents can be in XML, Outside In, HTML format, or contain user-defined tags (GPP).

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

To be able to search in these fields, you must specify a `FORMAT` and a `MODEL` file when you create the text index containing the documents.

Chapter 52. Searching natively stored XML documents

Generally, when creating an index on an XML data column, you are not required to specify a FORMAT. Net Search Extender selects format XML by default when a text index is created on a column of type XML. The format specifiers TEXT and HTML are not allowed on XML data columns.

The following sections address search on natively stored XML documents. You are shown how the concepts of section search can be applied to natively stored XML documents and how to integrate this functionality into XQuery processing.

In the subsequent samples that illustrate the creation and use of a text index on XML columns, the following XML document is used. It is stored in table t1, column c2 of type XML.

```
<?xml version="1.0">
<purchaseOrder orderDate="2001-01-20">
   <shipAddress countryCode="US">
      <name>Alice Smith</name>
      <street>123 Maple Street</street>
      <city>Mill Hill</city>
      <zip>90999</zip>
   </shipAddress>
   <item partNo="123" quantity="1">
      <name>S&B Lawnmower Type ABC-x</name>
      <price>239.90</price>
      <shipDate>2001-01-25</shipdate>
   </item>
   <item partNo="987" quantity="1">
      <name>Multifunction Rake ZYX</name>
      <price>69.90</price>
      <shipDate>2001-01-24</shipdate>
   </item>
</purchaseOrder>
```

Using the default document model

If no document model is specified in the CREATE INDEX statement, Net Search Extender uses the default document model.

About this task

One characteristic of the default document model is that section names are in XPath notation specifying the absolute path to each element and attribute. Note that section names in the search query are not XPath expressions that are evaluated during query execution. Instead they are names referring to specific parts (elements and attributes) within structured documents.

If you are not using a model file, define a text index for XML documents as follows:

```
db2text CREATE INDEX i1 FOR TEXT ON t1(c2) CONNECT TO mydbname
```

As column c2 is of datatype XML, you can omit the FORMAT specification. The FORMAT specification is set to XML by default in this case.

When no document model is specified, each XML element is assigned a name automatically depending on its absolute XPath within the document. For example, element price can be accessed by the section name /purchaseOrder/item/price in the search query. The attribute countryCode can be accessed by using the section name /purchaseOrder/shipAddress/@countryCode.

After the index is updated by using the **db2text update** command, a possible SQL expression using SECTION search with the scalar search function might be as follows:

```
SELECT c2 FROM t1
WHERE CONTAINS(c2, SECTIONS("/purchaseOrder/item/name") "Rake") = 1
```

The query returns the sample XML document shown previously.

Using a customized document model

If you want to define customized section names, you must specify a model file that assigns user defined names to certain parts of a document. An advantage of using a document model is that you can specify which parts of an XML document you want to index and use XPath expressions to specify these parts.

About this task

A model file for the previously mentioned XML document might look as follows:

```
<?xml version="1.0"?>
<XMLModel>
   <XMLFieldDefinition
      name="itemName"
      locator="/purchaseOrder/item/name" />
   <XMLFieldDefinition
      name="customerName"
      locator="//shipAddress/name" />
   <XMLAttributeDefinition
      name="partNumber"
      type="NUMBER"
      locator="/purchaseOrder//item/partNo" />
   <XMLFieldDefinition
      name="none"
      locator="/purchaseOrder/orderDate"
      exclude="yes" />
</XMLModel>
```

Note that the document model assigns the name itemName to the element /purchaseOrder/item/name which is referenced in the previous search query.

The index definition, using the model file, is:

```
CREATE INDEX i1 FOR TEXT ON t1(c2) DOCUMENTMODEL XMLModel IN
   /mydir/myfilename/xmlmodel.xml CONNECT TO mydbname
```

The document model name (using the parameter DOCUMENTMODEL) specifies the root element in the model file. This is XMLModel for XML document models. The path /mydir/ ... points to the file that defines the model.

The document model syntax supports a subset of the W3C XPath syntax which allows for the convenient identification of elements.

After creating the text index using the model file shown previously, and updating the index using the **db2text update** command, it is possible to search for the element /purchaseOrder/item/name as follows:

```
SELECT c2 FROM t1
WHERE CONTAINS(c2, SECTIONS("itemName") "Rake") = 1
```

Note the difference to the search query where no document model was specified. Both queries return the same sample XML document mentioned previously.

The XML document model also defines an attribute `partNumber` on the XML attribute `partNo` of element `item`. The data type of Net Search Extender attribute definitions must always be NUMBER.

The attribute definition in the sample model file shown previously, allows searching on value ranges like:

```
SELECT c2 FROM t1  WHERE CONTAINS
  (c2, ATTRIBUTE "partNumber" BETWEEN 300 AND 500) = 1
```

XQuery support

While searching for XML documents in the database, it is also possible to process the search results by using XQuery. By exploiting the DB2 database server's hybrid database engine, an SQL text search query can be combined with XQuery processing.

About this task

This is done by using the db2-fn:sqlquery() input function in the XQuery context. In order to use the XQuery input function, you must switch from SQL to XQuery by using the **`set language XQuery`** command, or the query has to be prefixed with the keyword `XQuery`. This is an important indicator to the parser that it is working with an XQuery expression and must follow the case sensitivity rules and syntax rules that apply to the XQuery language.

The db2-fn:sqlquery() function takes a string literal that represents a full-select. The db2-fn:sqlquery() function returns an XML sequence that represents the concatenation of the XML column values that are selected by the full-select.

The following expression can be used to combine text search and XQuery processing on natively stored XML documents:

```
XQUERY db2-fn:sqlquery('SELECT c2 FROM t1
           WHERE CONTAINS(c2,
           ''SECTIONS ("/purchaseOrder/item/name") "Rake" '')
     = 1 ')//shipAddress/name
```

This query returns all the `name` elements under the `shipAddress` element in XML documents that contain a purchasing order item named "Rake". You must explicitly select the XML column (in our case c2) in the SELECT statement.

The previous sample can be extended by a FLWOR construct as follows, and embedded in your application:

```
XQUERY FOR $item in db2-fn:sqlquery('SELECT c2 FROM t1
           WHERE CONTAINS(c2, '' SECTIONS ("/purchaseOrder/item/name") "Rake" '')
       = 1 ')
           WHERE $item[@partNo > "800"]
RETURN $item/price
```

Note that the full-select of the db2-fn:sqlquery() input function always returns the complete XML document in which a hit occurs.

Consider the following XML document that is stored natively in the database:

```
<?xml version="1.0" ?>
<dept bldg="101">
  <employee id="901">
     <name>Sabine</name>
     <resume>DB2 programmer</resume>
  </employee>
  <employee id="902">
     <name>Holger</name>
     <resume>XML expert</resume>
  </employee>
</dept>
```

Searching for an employee in your department where the term "XML" is contained in the resume might look as follows:

```
SELECT c2 FROM t1  WHERE CONTAINS(c2, SECTIONS("/dept/employee/resume") "XML")=1
```

This select statement returns the complete XML document. Embedding the search query in XQuery as follows:

```
XQUERY db2-fn:sqlquery('SELECT c2 FROM t1
         WHERE CONTAINS(c2,
         ''SECTIONS ("/dept/employee/resume") "XML" '') =1') //employee/name
```

returns the following two results:

```
<name>Sabine</name>
<name>Holger</name>
```

Notice that although the employee Sabine does not have the term "XML" in her resume, she appears in the resulting sequence of this XQuery. This happens because the full-select returns the whole document, that is, it returns the complete XML document that has at least one employee with the term "XML" in the resume.

If you want the query to return only the result `<name>Holger</name>`, issue the following XQuery statement:

```
XQUERY for $d in db2-fn:sqlquery('SELECT c2 FROM t1
         WHERE CONTAINS(c2,
         ''SECTIONS ("/dept/employee/resume") "XML" '') =1')
   return §d/dept/employee/name[contains(parent::employee/resume,"XML")];
```

Net Search Extender filters out all XML documents that have the term XML in the section `/dept/employee/resume` by using a structure sensitive full-text index on the XML column. Base on the returned subset of XML documents, the return statement `return §d/dept/employee/name[contains(parent::employee/resume,"XML")]` returns only those <name> elements that have XML in their sibling element called <resume> by navigating the XML document using the XPath axis.

Chapter 53. Structured document support

How a document model describes structured documents

Documents in HTML or XML format are examples of structured documents; they contain tags that identify text fields and document attributes. Text fields can contain information like the title, author, or a description of the document.

The following is an extract from a structured plain-text document. It contains text that is delimited by HTML-like tags.

```
[head]Handling structured documents
[/head]

[abstract]This document describes the concept of structured documents
and the use of document models to...
[/abstract]
:
:
```

When Net Search Extender indexes structured documents, it has to recognize the structure so that it can index the text field and the attributes, and store them together with a unique name. This enables Net Search Extender to selectively search in a particular text field or to find documents that have a particular attribute by using the SECTION or ATTRIBUTE clause.

To enable Net Search Extender to understand the structure of a particular document format, you must pass Net Search Extender a definition of the structure in a *document model*. Alternatively, you can use the default document models provided by Net Search Extender.

You specify the name of the document model as an argument when you call the **CREATE INDEX** command to index the documents. For example, `CREATE INDEX i1 FOR TEXT ON t1(c2) DOCUMENT MODEL GPPModel IN mymodel.xld CONNECT TO db`

The parameter **GPPModel** refers to the type of document model you are using.

Before you can index documents using a document model, you must first define a document model and then make the document model known to the index.

Note: If XML documents use indexes that are not well-formed, the indexing process will stop at the point where the problem is encountered in the document. This means that only a part of the document will be indexed. If you do not correct the document, you will only be able to search in the parts of the document that were indexed. This will occur only if your table column type is not XML.

An example of a document model

A document model consists of text field definitions and attribute definitions.

You must define one document model for each document format that you intend to index. Here is a simple document model for plain-text structured documents. Note that GPP in the example stands for General Purpose Parser.

```
<?xml version="1.0"?>
<GPPModel>                  - the GPP document model begin here

  <GPPFieldDefinition       - a field definition begins here
  name="Head"               - the name you assign to this field
   start="[head]"           - the boundary string at the beginning of the field
  end="[/head]"             - the boundary string at the end of the field
  exclude="YES" />

  <GPPFieldDefinition       - the next field definition begins here
  name="Abstract"
  start="[abstract]"
  end="[/abstract]"
  exclude="NO" />
:
:
</GPPModel>
```

Document models are specified in the XML language using tags as defined in Chapter 54, "Document model reference," on page 191. A document model consists of text field definitions and attribute definitions. The previous example illustrates only text field definitions defined in `GPPFieldDefinition` elements. In a similar way, you can use `GPPAttributeDefinition` to define document attributes.

The first line `<?xml version="1.0"?>` in the example specifies that the document model is written using XML tags. Each of the text field definitions specifies boundary strings to identify the `start` and `end` of the field definition in the source document. So, whenever a document contains the sequence of characters `[head]` followed by some text and the sequence of characters `[/head]`, the text between those boundary strings is taken to be the content of the text field that is identified by the name `head`.

You assign a field name to each field definition. This field name is the means by which a query can restrict search to the content of a text field using a SECTION clause in the CONTAINS function. The name of the field can be either fixed or can be derived by a rule from the structural unit's content. Such a name could be, for example, the tag name of an XML entity, or the name of an XML attribute.

Document models

A document model primarily controls what parts of a document's structure need to be indexed and how they are indexed.

Its purpose is to:
- Identify text fields that should be distinguished in the source document
- Determine the type of such a text field
- Assign a field name to the text field

When the document model identifies text as belonging to a text field, the text is considered to be part of the textual content of the document, and terms are extracted and stored in the index.

The elements of a document model vary depending on the parser used for that document format:
- For HTML format, a document model uses the HTML tag names to define which tags should be indexed, and how to handle meta-tag information.

- For XML format, there is no predefined set of tags, so a document model must first define which tags are of interest. XML elements of the same name can also be distinguished based on what other elements they are embedded in.
- For GPP (general purpose parser) format, the document model interacts even more deeply with the parser, because it has to determine the boundaries of the text fields. Here the field definition must specify strings for detecting the boundaries of fields.
- For Outside In formats, a document model uses tags similar to HTML tag names to define which tags should be indexed, and how to handle meta-tag information. Note that the Outside In Transformation Technology is also known as INSO.

Text fields

A document model lets you identify document parts or sections as either belonging to a particular text field or as being a document attribute, or both.

The text of a document is fully indexed regardless of whether or not it is part of a text field. Meaningful terms are extracted and stored in the index. This means that unrestricted text searches include a search of that text.

However, by defining text fields, you can search for text selectively in a particular field. For example, you can search for documents that contain the word `structure` in the text field `Abstract`. For example, `SELECT doc from my_docs WHERE CONTAINS (doc, SECTIONS(Abstract) "structure" = 1`.

A text field can occur several times in a document. For instance, you can define a text field that contains all figure captions. A text field may also overlap with another text field.

If you want to avoid indexing the content of certain text fields, you can specify a field definition that contains `exclude="YES"`. You can find a list of limitations for text fields and document attributes in “Limitations for text fields and document attributes” on page 193.

Document attributes

Document attributes contain short, formatted information of type `number`.

In contrast to text fields, you can use value ranges to search documents containing such attributes.

Attributes are not stored with indexed text, but in a separate item index. So, to search for documents by content of an attribute, you must make an attribute search explicitly on the attribute. For example, `SELECT doc FROM my_docs WHERE CONTAINS (doc,ATTRIBUTE "year" BETWEEN 2001 AND 2005) = 1`.

Number attributes

Net Search Extender provides a parser that recognizes floating-point numbers.

The following are examples of correct and incorrect formats for attribute values.

Table 1. Supported formats for attribute values

Correct format	Incorrect format
1000 1 000 1.000 - where the period is a decimal character	1,000
100 000 100 000.00123	1 000 000 - two spaces between 1 and 0

Note that space characters are not allowed in the decimal fraction of a number. For example, 1 000.000 100 is treated as two numbers, 1000.000 and 100.

Language-specific separators and language-specific monetary formats are not supported.

Default document models

When using one of the default document models, you must remember that all fields are indexed, and no special information is extracted, and no numeric attribute is indexed.

For HTML, XML, and Outside In filtered documents, Net Search Extender provides default document models that are used if you do not specify a document model during index creation. For structured plain text documents, you must provide and specify a document model.

If you use one of the default document models:

- All fields are indexed, and no special information, such as meta information, is extracted.
  - For HTML and INSO formats, each field is assigned the name of the corresponding tag.
  - For XML, all XML nodes of an XML document are mapped to overlapping fields which are identified by the fully qualified element paths of the corresponding nodes. For example, the path `/play/role/name`.
- No numeric attribute is indexed (as no numeric attribute is defined in the default document model).

Table 2. Behavior of the default document models for the supported document formats

Document type	Behavior of the default document model
HTML	Accepts these as text fields: <a> <address> <au> <author> <h1> <h2> <h3> <h4> <h5> <h6> <title>. Field name is the tag name, for example "address".
XML	Accepts all tags as text fields. The field name is the fully qualified element path name, for example "/play/title".
Structured plain text (GPP)	No default document model.
Outside In (INSO)	Accepts as text fields, the document properties shown in "Definition of a document model for Outside In filtered documents" on page 188 as returned by the Outside In filters. The Field name is the name of the document property used by Outside In, for example: "SCCCA_TITLE".

For each type of document, a default document model is defined. As each model is different, an example and explanation is provided for each one in the following sections.

Note:

Although the default document models do correctly process documents, for better indexing and search you should define your own document models.

With the default document model, the text of a document is fully indexed regardless of whether or not it is part of a text field. This means that unrestricted text searches include a search of that text.

Definition of a document model for structured plain-text documents

This topic contains the parameters of the document model elements and restrictions.

name You assign a name to the text field or document attribute for each definition. The names enable you to restrict a search query to the content of a specific text field or document attribute. Using the previous examples, you could search for documents containing the word `structure` in the text field named `Abstract`.

start A boundary string in code page UTF-8 that marks the beginning of the text field or document attribute. There are no rules for specifying boundary strings; they can be any arbitrary UTF-8 string. Here are some examples: start="introduction:", start="note!", start="$$...".

Nonprintable characters and the special XML characters "<" and "&" must be specified using the default XML character entries ("&lt;" for "<", and "&amp;" for "&").

end Optional. A boundary string in code page UTF-8 that marks the end of the text field or document attribute. If you do not specify an end tag, the next found start tag is assumed to be the end of the field. If no subsequent start tag is found, the field extends to the end of the document, and no further fields are identified.

type The type of document attribute must always be "NUMBER". The parameter does not apply to field definitions.

exclude
YES or NO. A parameter that determines whether the text in a field definition should be excluded and therefore, not indexed. This parameter does not apply to attribute definitions.

In the example, the field definition "head" would be excluded, but definition "abstract" included.

Restriction:

- There must not be two field definitions or attribute definitions with the same start tag. However, a field definition and attribute definition may have the same start tag and end tags.
- A start tag must not be a proper prefix of another. For example, you cannot have a start tag "author" and a start tag "authority".
- Start tags and end tags must not be empty strings.

What happens when a GPP document is indexed

The general-purpose parser scans the document looking for one of the start boundary strings. When it finds a start boundary string, it parses the subsequent field until it finds the corresponding end boundary string.

The content of the field is then indexed according to the definition term, that is, as a text field or document attribute. If the text field and document attribute have the same start and end boundary strings, the content of the field is indexed as both a text field and a document attribute.

No nesting of fields is allowed; if a new start boundary string is found in a field before the end boundary string has been reached, the new start boundary string is interpreted as normal text.

If no corresponding end boundary string is found, the field is assumed to extend to the end of the document, and an appropriate reason code is reported.

If no end boundary string is specified in the document model, the new start boundary string signals the end of the previous field.

Definition of a document model for HTML documents

The HTML parser converts the text to code page UTF-8. It performs HTML tag recognition and classifies them into tag classes.

NetSearch Extender uses the following tag classes:
- Tagged information to be ignored, such as font information
- Tags that provide positional information, such as <p>; for new paragraph
- Tags that provide structural information, such as <Title>

It recognizes all character entity references defined in HTML 4, like "&auml;" (ä) and resolves them to the corresponding code points in UTF-8.

It recognizes meta tags and parses the meta tag text.

Here is an example of an HTML document:

```
<HTML>
<HEAD>
<META NAME="year" CONTENT="2002">
<TITLE> The Firm </TITLE>
</HEAD>
<BODY>
<H1>Synopsis</H1>;


<H1>Prologue</H1>;:
:
</BODY>
```

Here is an example of an HTML document model:

```
<?xml version="1.0"?>
<HTMLModel>

 <HTMLFieldDefinition
 name="subtitle"
 tag="title"
 exclude="NO" />
```

```
<HTMLFieldDefinition               - This is the start of text field
name="header1"
tag="h1"
exclude="YES" />                   - This is the end of the text field

<HTMLAttributeDefinition           - This is the start of the document
name="year"                               attribute
tag="meta"
meta-qualifier="year"
type="NUMBER" />                   - This is the end of the document
                                       attribute
</HTMLModel>
```

The first line, `<?xml version="1.0"?>`, specifies that the document model is written using XML tags. Note that this model is not written for XML format documents.

Each field is defined within a `HTMLFieldDefinition` or `HTMLAttributeDefinition` tag, which contain element parameters.

All the text field definitions must be contained within the `<HTMLModel>` tag. The tag name is passed as a parameter during index creation: `CREATE INDEX iA FOR TEXT ON T1(C2) DOCUMENTMODEL HTMLModel IN myModel.xml CONNECT TO db`.

These are the parameters of the document model elements:

name
You assign a name to the text field or document attribute for each definition. The names enable you to restrict a search query to the content of a specific text field or document attribute. Using the previous examples, you could search for documents containing the word `firm` in the text field named `subtitle`.

tag
Identifies an element whose start and (implied) end tags mark the text field or document attribute. The text that is inside an element of that name makes up the content of the defined field.

The case of the tag is ignored.

Using the previous examples, the text following any `H1` tag is indexed as being part of the field "header1". Based on the sample document, "synopsis" and "prologue" would be indexed.

meta-qualifier
This tag has to be used with the **tag** element. By specifying tag="meta", the value of the content that matches the meta-qualifier is extracted.

In the HTML document example, the meta tag has the following elements:

```
<META NAME="year" CONTENT="2002">
```

In the document model example, meta-qualifier="year". Therefore, the content "2002" is indexed as the value of the attribute "year".

type
The type of document attribute must be "NUMBER". The parameter does not apply to field definitions.

exclude
YES or NO. A parameter that determines whether the text in a field definition should be excluded and therefore, not indexed. This parameter does not apply to attribute definitions.

In the example, the field definition "header1" would be excluded, but definition "subtitle" included.

All other text of a document is indexed, but not as part of any field.

Element parameters

These are the parameters of the document model elements:

name
You assign a name to the text field or document attribute for each definition. The names enable you to restrict a search query to the content of a specific text field or document attribute. Using the previous examples, you could search for documents containing the word `firm` in the text field named `subtitle`.

tag
Identifies an element whose start and (implied) end tags mark the text field or document attribute. The text that is inside an element of that name makes up the content of the defined field.

The case of the tag is ignored.

Using the previous examples, the text following any `H1` tag is indexed as being part of the field "header1". Based on the sample document, "synopsis" and "prologue" would be indexed.

meta-qualifier
This tag has to be used with the **tag** element. By specifying tag="meta", the value of the content that matches the meta-qualifier is extracted.

In the HTML document example, the meta tag has the following elements:

```
<META NAME="year" CONTENT="2002">
```

In the document model example, meta-qualifier="year". Therefore, the content "2002" is indexed as the value of the attribute "year".

type
The type of document attribute must be "NUMBER". The parameter does not apply to field definitions.

exclude
YES or NO. A parameter that determines whether the text in a field definition should be excluded and therefore, not indexed. This parameter does not apply to attribute definitions.

In the example, the field definition "header1" would be excluded, but definition "subtitle" included.

All other text of a document is indexed, but not as part of any field.

Definition of a document model for XML documents

A document model for XML documents allows you to define how an element found in an XML document is mapped to a field, a document attribute, or both.

Here is an example of an XML document:

```
<?xml version="1.0"?>
<purchaseOrder orderDate="2001-01-20">            [4]
    <shipAddress countryCode="US">               [1]
        <name>Alice Smith</name>                 [2]
        <street>123 Maple Street</street>
        <city>Mill Hill</city>
        <state>CA</state>
        <zip>90999</zip>
    </shipAddress>
    <item partNo="123" quantity="1">          [3]
```

```
        <name>S&B Lawnmower Type ABC-x</name>
        <price>239.90</price>
        <shipDate>2001-01-25</shipDate>
    </item>
    <item partNo="987" quantity="1">     [3]
        <name>Multifunction Rake ZYX</name>
        <price>69.90</price>
        <shipDate>2001-01-24</shipDate>
    </item>
</purchaseOrder>
```

Here is an example of an XML document model that matches the previous sample document:

```
<?xml version="1.0"?>
<XMLModel>

<XMLFieldDefinition                  [1]
name="addresses"
locator="/purchaseOrder/shipAddress" />

<XMLFieldDefinition                 [2]
name="customerName"
locator="//shipAddress/name"
exclude="yes" />

<XMLAttributeDefinition                 [3]
name="partNumber"
type="NUMBER"
locator="/purchaseOrder//item/@partNo" />

<XMLFieldDefinition                 [4]
name="none"
locator="/purchaseOrder/@orderDate" />

</XMLModel>
```

The first line, `<?xml version="1.0"?>`, specifies that the model is written using XML. Each field is defined within a `XMLFieldDefinition` or `XMLAttributeDefinition` tag, which contains element parameters.

Note that all the text field definitions must be contained within the `<XMLModel>` tag. This tag name is passed as a parameter during index creation: `CREATE INDEX i1 FOR TEXT ON T1(C2) DOCUMENTMODEL XMLModel in myModel.xml CONNECT TO db`.

The fields and attributes in the sample are marked with numbers that correspond to the definitions in the example model file.

Nesting of fields is allowed, for example, if the XPath location of one specification selects a node that lies inside an XML element selected by another attribute definition. Nested fields is shown in the previous sample XML document. The field `addresses` selects a node in the XML document that dominates the node selected by field `customerName`. The content of that embedded node therefore, logically belongs to both fields. Although text fields may be overlapping, the text inside those fields is indexed only once. In this example, when searching with a field restriction, `Alice Smith` is found in `addresses` as well as in `customerName`. However, due to the matching semantics of the locator expression, it is not possible to map one and the same XML node to multiple fields.

Net Search Extender does not attempt to detect the code page of an XML document. The DB2 code page is taken.

The content of fields is determined by the following rules:

- For a field whose locator matches a comment, a processing instruction, or an XML attribute, the field content is the actual comment text, processing instruction text, or attribute value text.
- For a field that matches an XML element or the root node, the field content consists of any text from any embedded element except for elements that are matched by fields having the specification `exclude="YES"`.

The document must contain well-formed XML, but it is not necessary for a DTD to be specified in the XML document. No DTD validation or external entity resolution is carried out; Net Search Extender only matches the XML document against the document model. Internal entities are substituted as required by XML.

Element parameters

These are the parameters of the document model elements:

name You assign a name to the text field or document attribute for each definition. These names enable you to restrict a search query to the content of a specific text field or document attribute.

You can use one of the following variables in a name. The variable is replaced by a string generated from the matching element in the source document.

Variable
Value

$(NAME)
The actual qualified name (QName) of the XML element that matched the XPath.

$(LOCALNAME)
The actual local name (without prefix) of the XML element that matched the XPath.

$(PATH)
The actual absolute path as a sequence of slashes and tags of the XML element that matched the XPath (used as name in the default document model).

type The type of document attribute must be "NUMBER". The parameter does not apply to field definitions.

locator
Expressions in the XPath language that select the parts of the source documents to be used as search fields.

When writing a XML Document Model file, the qualified names, known as QNames, inside a `locator` must be identical to some tags in the XML document, otherwise no fields will be recognized and the queries on fields will not return a result.

The following are examples of locators.

`purchaseOrder | salesOrder`
All `purchaseOrder` elements and `salesOrder` elements

`shipAddress`
All `shipAddress` elements

***** All elements (this is the abbreviation of `child::*` - see the syntax for further information)

name/item
All `item` elements that have a `name` parent

purchaseOrder//item
All `item` elements that have a `purchaseOrder` ancestor

/ The root node

comment()
All comment nodes

processing-instruction()
All processing instructions

attribute::* (or @*)
All attribute nodes

A literal is a string enclosed either in single or double quotation marks. For an exact definition of terminal tokens see the XML recommendations.

The XPath locators that are supported by the Net Search Extender document model are similar to XML Stylesheet Language Transformation (XSLT) patterns. They comprise exactly the subset of XSLT patterns that do not contain any predicates nor the functions 'id' and 'key' nor the node tests 'text()' and 'node()'.

ignore YES or NO. Use the parameter to make exceptions to the locator.

Sometimes you may want to specify a general locator, such as *, to match the nodes you want to index. But you may also specify that some nodes matching a more specific locator should not be indexed.

To formulate this, include a field definition with the more specific locator for the nodes to be ignored during indexing. You then give this locator a higher priority than the one with the general locator (see following section), and specify `ignore="yes"`. This indicates to the indexer that it must not generate field information for the matching nodes.

Note that when such an ignored node is embedded in a field-generating node, the content of the ignored node gets indexed, because it also belongs to the contents of the field-generating node.

priority
A floating point number between -1 and +1 that specifies the priority to be given to a definition found by a particular locator.

If you do not specify a priority, the default priorities are used:

- Multiple alternatives separated by `|` are treated as a set of definitions, one for each alternative.
- Locators that match by a single name; that is, locators of either of the following forms have default priority 0:
  - `ChildOrAttributeAxisSpecifier QName`
  - `ChildOrAttributeAxisSpecifier processing-instruction(Literal))`
- Locators of the form `ChildOrAttributeAxisSpecifier NCName:*` have default priority -0.25.
- Other locators of the form `ChildOrAttributeAxisSpecifier NodeTest` have default priority -0.5.
- Any other locator has default priority 0.5.

Note that the more specific the locator is, the higher the default priority. For example, the unspecific locator * gives a low priority to the found definition, whereas a name is a more specific locator and gives a higher priority.

Also note that when a node is matched by more than one locator, you can determine which of the definitions are chosen by assigning priorities to them. The definition with the highest priority is chosen. If two definitions have the same priority, the latest is chosen.

This conflict resolution is the same as that used in XML Stylesheet Language Transformation (XSLT).

exclude
YES or NO. A parameter that determines whether the text in a field definition should be excluded and therefore, not indexed. This parameter does not apply to attribute definitions.

In the example, the field definition "customerName" would be excluded, but definition "addresses" included.

Note: If there are more than one field definition with same locator value but different names, then only the last field definition is effective. All previous field definitions are ignored.

The following model file contains one field definition.

```
<XMLFieldDefinition
name="from"
exclude="NO"
locator="/document/email/from" />
```

After indexing the documents are found when searching in section "`from`".

Next, another field definition is added to the end of the model file.

```
<XMLFieldDefinition
name="from_last"
exclude="NO"
locator="/document/email/from" />
```

After recreating and updating the index, the documents are found when searching in section "`from_last`" but no documents are found when searching in section "`from`".

Definition of a document model for Outside In filtered documents

Document models for the Outside In format are very similar to HTML document models in that they allow you to map structural elements identified by a given set of tags toNet Search Extender text fields and document attributes.

Assume you have a set of Microsoft Word documents and want to index the document properties "title", "subject", and "keyword" as fields, and the document properties "author" and "category" as document attributes. The following example for an Outside In document model will achieve this mapping:

```
<?xml version="1.0"?>
<INSOModel>

<INSOFieldDefinition
name="title"
tag="SCCCA_TITLE"/>
```

```
<INSOFieldDefinition
name="title"
tag="SCCCA_SUBJECT"/>

<INSOFieldDefinition
name="title"
tag="SCCCA_KEYWORDS"/>

<INSOAttributeDefinition
name="author"
tag="SCCCA_AUTHOR"
type="STRING"/>

<INSOAttributeDefinition
name="category"
tag="SCCCA_CATEGORY"
type="STRING"/>

</INSOModel>
```

Element parameters

These are the parameters of the document model elements:

name
A name that you assign to the text field or document attribute. You assign a field name to each field definition, and an attribute name to each attribute definition. These names are the means by which a query can restrict search to the content of a certain text field and can search for documents having a certain attribute.

tag
Identifies a tag whose begin and end or implied-end elements mark the text field or the document attribute. The text that is inside an element of that name makes up the content of the defined field or attribute. The case of the tag is disregarded. Possible values are described in the following paragraph.

type
The type of document attribute can be either "NUMBER", "DATE", or "STRING". This parameter does not apply to field definitions.

exclude
YES or NO. A parameter that determines whether the text in a field definition should be excluded and therefore, not indexed. This parameter does not apply to attribute definitions.

Outside In document models consist of field and attribute definitions that each define a name and a tag. For attribute definitions a type is also required, whereas field definitions have an optional "exclude" flag. As with HTML models, the name attribute of such a definition defines the name of the Net Search Extender field or attribute that the document part is to be mapped on. This can be an arbitrary UTF-8 text string. For additional information, see the Outside In Content Access Specification, Version 7.5.

For a list of possible values for the tag attribute relating to the Outside In begin, end and document property tags.

What happens when an Outside In document is indexed

By default, all the text is indexed as not belonging to any field.

Whenever a begin tag that appears in the stream of text matches a definition item in the document model that is currently active, the text between the begin tag and its corresponding end tag is treated according to that definition term. For example, as an indexed field, an excluded field, or as an attribute or both.

If no matching definition exists, the begin tag and its corresponding end tag are ignored.

As Outside In filters automatically recognize the format and code page of the document, the CCSID specification has no effect. If the Outside In filters are unable to determine the correct format and code page, the document is treated as an ASCII file.

Chapter 54. Document model reference

Net Search Extender provides information regarding supported document models.

DTD for document models

The following topic contains a formal description of the syntax of document models in the form of a document type definition (DTD).

```
<!ELEMENT GPPModel (GPPFieldDefinition|GPPAttributeDefinition)+>
<!ELEMENT HTMLModel (HTMLFieldDefinition|HTMLAttributeDefinition)+>
<!ELEMENT XMLModel (XMLFieldDefinition|XMLAttributeDefinition)+>

<!ELEMENT GPPFieldDefinition EMPTY>
<!ATTLIST GPPFieldDefinition name CDATA #REQUIRED>
<!ATTLIST GPPFieldDefinition start CDATA #REQUIRED>
<!ATTLIST GPPFieldDefinition end CDATA #IMPLIED>
<!ATTLIST GPPFieldDefinition exclude (YES|NO) NO>

<!ELEMENT GPPAttributeDefinition EMPTY>
<!ATTLIST GPPAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition start CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition end CDATA #REQUIRED>
<!ATTLIST GPPAttributeDefinition type NUMBER #REQUIRED>

<!ELEMENT HTMLFieldDefinition EMPTY>
<!ATTLIST HTMLFieldDefinition name CDATA #REQUIRED>
<!ATTLIST HTMLFieldDefinition tag CDATA #REQUIRED>
<!ATTLIST HTMLFieldDefinition meta-qualifier CDATA #IMPLIED>
<!ATTLIST HTMLFieldDefinition exclude (YES|NO) NO>

<!ELEMENT HTMLAttributeDefinition EMPTY>
<!ATTLIST HTMLAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST HTMLAttributeDefinition tag CDATA #REQUIRED>
<!ATTLIST HTMLAttributeDefinition meta-qualifier CDATA #IMPLIED>
<!ATTLIST HTMLAttributeDefinition type NUMBER #REQUIRED>

<!ELEMENT XMLFieldDefinition EMPTY>
<!ATTLIST XMLFieldDefinition name CDATA #REQUIRED>
<!ATTLIST XMLFieldDefinition locator CDATA #REQUIRED>
<!ATTLIST XMLFieldDefinition ignore (YES|NO) NO>
<!ATTLIST XMLFieldDefinition priority CDATA #IMPLIED>
<!ATTLIST XMLFieldDefinition exclude (YES|NO) NO>

<!ELEMENT XMLAttributeDefinition EMPTY>
<!ATTLIST XMLAttributeDefinition name CDATA #REQUIRED>
<!ATTLIST XMLAttributeDefinition locator CDATA #REQUIRED>
<!ATTLIST XMLAttributeDefinition ignore (YES|NO) NO>
<!ATTLIST XMLAttributeDefinition priority CDATA #IMPLIED>
<!ATTLIST XMLAttributeDefinition type NUMBER #REQUIRED>
```

Semantics of locator (XPath) expressions

An XPath expression needs to be interpreted with respect to a context node, and denotes a set of nodes. When used as Net Search Extender selector patterns, the context node is free, that is, a relative path pattern `p` is interpreted as `//p`.

According to the XML data model, XML documents are viewed as trees containing these kinds of nodes:

- The root node

- Element nodes
- Text nodes
- Attribute nodes
- Namespace nodes
- Processing instruction nodes
- Comment nodes

The links between those nodes, that is the tree-forming relationship, reflect the immediate containment relationship in the XML document.

The *root node* can appear only at the root and nowhere else in the tree. It contains, as its children, the document element and optional comments and processing instructions.

Element nodes can contain any kinds of nodes except for the root node. The other kinds of nodes are only allowed as leaf nodes of the tree.

There are three kinds of *containment links*: 'child', 'attribute', and 'namespace'. The 'attribute' and 'namespace' containment links must lead to attribute and namespace nodes. So, to access the children of an element node (in terms of graph theory) you need to follow 'attribute' links to find all contained attributes, follow 'namespace' links to find all contained namespace declarations, and follow 'child' links to find contained elements, text nodes, processing instructions, and comments.

These are the Net Search Extender XPath selector patterns:

- `Pattern '|' LocationPathPattern` in context N denotes the union of the nodes matched by Pattern and LocationPathPattern, both in context N.
- `'/'RelativePathPattern` in context N denotes whatever this RelativePathPattern denotes in the context of the root.
- `'//'RelativePathPattern` in context N denotes the union of the denotations of this RelativePathPattern interpreted in any context that is a descendant (on the child axis) of the root.
- `RelativePathPattern '/' StepPattern` matches a node in context N, if and only if that node is matched by StepPattern in the context of its parent, and its parent node is matched by RelativePathPattern in context N.
- `RelativePathPattern '//' StepPattern` matches a node in context N, if and only if that node is matched by StepPattern in the context of its parent, and it has an ancestor node that is matched by RelativePathPattern in context N.
- `'child'::NodeTest` (abbreviated syntax: NodeTest) in context N matches a node that is a child of N (on the child axis) and that satisfies NodeTest.
- `'attribute'::NodeTest` (abbreviated syntax: @NodeTest) in context N matches a node that is an attribute of N and that satisfies NodeTest.
- `NodeType '(' ')'` is satisfied for a node if and only if it is of the specified type.
- `'processing-instruction' '(' Literal ')'` is satisfied for any processing-instruction-type node that has Literal as its name.
- `'*'` is satisfied for any element or attribute node (name mask for element name).
- `NCName ':' '*'` is satisfied for any element node that has NCName as its name prefix.
- `QName` is satisfied for any node with the specified name.

Note

A NodeTest of the form NameTest assumes the node to be of the principal type on the selected axis, which is attribute type on the attribute axis and child type on the child axis. Consequently, NameTest cannot be used to choose comments or processing instruction nodes, but only child and attribute nodes. Moreover, the patterns allow for the selection of any kind of node, except for namespace nodes, because the axis specifier 'namespace' is not allowed.

Examples of patterns:

- `chapter | appendix` denotes all chapter elements and appendix elements
- `table` denotes all table elements
- `*` denotes all elements (note that this is the abbreviation of child::*)
- `ulist/item` denotes all item elements that have a ulist parent
- `appendix//subsection` denotes all subsection elements with an appendix ancestor
- `/` denotes the singleton set containing just the root node
- `comment()` denotes all comment nodes
- `processing-instruction()` denotes all processing instructions
- `attribute::* (or @*)` denotes all attribute nodes

This is the syntax of the locator element:

```
Locator     ::= LocationPathPattern
          | Locator '|'  LocationPathPattern
 LocationPathPattern  ::= '/' RelativePathPattern ?
          | '//'? RelativePathPattern
 RelativePathPattern  ::= StepPattern
          | RelativePathPattern '/' StepPattern
          | RelativePathPattern '//' StepPattern
 StepPattern    ::= ChildOrAttributeAxisSpecifier NodeTest
 ChildOrAttributeAxisSpecifier ::=
          ('child' | 'attribute') '::'
          | '@'?
 NodeTest    ::= NameTest
          | NodeType '(' ')'
          | 'processing-instruction' '(' Literal ')'
 NameTest    ::= '*' | NCName ':' '*' | QName
 NodeType    ::= 'comment' | 'processing-instruction'
```

NCName and QName are as defined in the XML Names Recommendation:

NCName
An XML name containing no colons

QName
A NCName that can be preceded by a NCName followed by a colon. For example: `NCName:NCName`

Limitations for text fields and document attributes

There is a list of limitations for the text field and document attributes, as well as a list of tags that can be included in an HTML document model.

The following lists the limitations for text fields and document attributes:

- Maximum number of fields in an index: 32767

- Maximum number of values for one attribute of type STRING in one document: 1024
- Maximum number of attributes of type STRING: 253
- Number of characters in a STRING attribute value is truncated to 128
- Maximum number of attributes of types DATE and NUMBER: 32766
- Number of characters in a DATE or NUMBER attribute value is truncated to 128
- For NUMBER attributes, a double precision floating point number is accepted as a value.
- Maximum number of values that can be specified for one attribute of type DATE or NUMBER in one document: unlimited

These are the tags that can be included in an HTML document model:
- `<A>`
- `<ADDRESS>`
- `<AU>`
- `<AUTHOR>`
- `<H1>`
- `<H2>, <H3>, <H4>, <H5>`
- `<H6>`
- `<TITLE>`

Tags like `<HEAD>` and `<BODY>` that can contain other tags, cannot be specified in an HTML document model as a text field.

Outside In tag attribute values

The following topic contains a list of possible values for the tag attribute relating to Outside In document property tag types.

```
SCCCA_ABSTRACT
SCCCA_ACCOUNT
SCCCA_ADDRESS
SCCCA_ATTACHMENTS
SCCCA_AUTHORIZATION
SCCCA_BACKUPDATE
SCCCA_BASEFILELOCATION
SCCCA_BILLTO
SCCCA_BLINDCOPY
SCCCA_CARBONCOPY
SCCCA_CATEGORY
SCCCA_CHECKEDBY
SCCCA_CLIENT
SCCCA_COMPANY
SCCCA_COMPLETEDDATE
SCCCA_COUNTCHARS
SCCCA_COUNTPAGES
SCCCA_COUNTWORDS
SCCCA_CREATIONDATE
SCCCA_DEPARTMENT
SCCCA_DESTINATION
SCCCA_DISPOSITION
SCCCA_DIVISION
SCCCA_DOCCOMMENT
SCCCA_DOCTYPE
SCCCA_EDITMINUTES
SCCCA_EDITOR
SCCCA_FORWARDTO
SCCCA_GROUP
SCCCA_KEYWORD
```

```
SCCCA_LANGUAGE
SCCCA_LASTPRINTDATE
SCCCA_LASTSAVEDBY
SCCCA_MAILSTOP
SCCCA_MANAGERSCCCA_MATTER
SCCCA_OFFICE
SCCCA_OPERATOR
SCCCA_OWNER
SCCCA_PRIMARYAUTHOR
SCCCA_PROJECT
SCCCA_PUBLISHER
SCCCA_PURPOSE
SCCCA_RECEIVEDFROM
SCCCA_RECORDEDBY
SCCCA_RECORDEDDATE
SCCCA_REFERENCE
SCCCA_REVISIONDATE
SCCCA_REVISIONNOTES
SCCCA_REVISIONNUMBER
SCCCA_SECONDARYAUTHOR
SCCCA_SECTION
SCCCA_SECURITY
SCCCA_SOURCE
SCCCA_STATUS
SCCCA_SUBJECT
SCCCA_TITLE
SCCCA_TYPIST
SCCCA_USERDEFINEDPROP
SCCCA_VERSIONDATE
SCCCA_VERSIONNOTES
SCCCA_VERSIONNUMBER
```

Possible values for the tag attribute relating to Outside In begin and end tag subtypes:

```
SCCCA_ALTFONTDATA
SCCCA_ANNOTATIONREFERENCE
SCCCA_CAPTIONTEXT
SCCCA_CHARACTER
SCCCA_COMPILEDFIELD
SCCCA_COUNTERFORMAT
SCCCA_CUSTOMDATAFORMAT
SCCCA_DATEDEFINITION
SCCCA_DOCUMENTPROPERTYNAME
SCCCA_ENDNOTEREFERENCE
SCCCA_FONTANDGLYPHDATA
SCCCA_FOOTNOTEREFERENCE
SCCCA_FRAME
SCCCA_GENERATEDFIELD
SCCCA_GENERATOR
SCCCA_HYPERLINK
SCCCA_INDEX
SCCCA_INDEXENTRY
SCCCA_INLINEDATAFORMAT
SCCCA_LISTENTRY
SCCCA_MERGEENTRY
SCCCA_NAMEDCELLRANGE
SCCCA_REFERENCEDTEXT
SCCCA_STYLE
SCCCA_SUBDOCTEXT
SCCCA_TOA
SCCCA_TOAENTRY
SCCCA_TOC
SCCCA_TOCENTRY
SCCCA_TOF
SCCCA_VECTORSAVETAG
SCCCA_XREF
```

Note that the tables include any document property, as well as all the tag subtypes recognized by the INSO filters. There are two subtype exceptions: SCCCA_DOCUMENTPROPERTY and SCCCA_BOOKMARK.

Part 13. Reference

Chapter 55. Administration commands for the instance owner

Instance owner administration consists of checking the status of Net Search Extender locking and update services, and starting and stopping these services.

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

The commands are subcommands of the **`db2text`** command and allow for the administration of Net Search Extender Services that are specific to a DB2 instance.

Command	Purpose
"CONTROL command"	Lists and deletes full-text index locks. Also lists the cache states.
"START command" on page 201	Starts the Net Search Extender instance services.
"STOP command" on page 202	Stops the Net Search Extender instance services.
"DB2EXTHL command" on page 208	Changes the maximum size of the input parameter of the highlight UDF.

CONTROL command

Lists and deletes full-text index locks managed by the Net Search Extender Instance Services.

Purpose

If the locking and update services are running, you can view their status as well as information about the activated cache.

In a partitioned database environment this only affects the current partition. The user is responsible for invoking the DB2 command **`db2_all`** for the required partitions.

Authorization

To issue the command successfully, the user must be the DB2 instance owner with DBADM and DATAACCESS authority.

Required connection

This command must be issued from the DB2 database server.

Command syntax

```
►►─CONTROL──┬─CLEAR──|set-of-locks|──────────────────┬──────────────────►◄
            ├─LIST──|set-of-locks|───────────────────┤
            ├─SHOW-CACHE-STATUS-FOR──|index-specification|─┤
            └─STATUS─────────────────────────────────┘
```

set-of-locks:

```
├──ALL-LOCKS-FOR──┬─│database-specification│─┬──────────────────────────────────┤
                  └─│index-specification│────┘
```

index-specification:

```
├──│database-specification│──INDEX──┬────────────────────┬──index-name──────────────┤
                                    └─index-schema-"."───┘
```

Database-specification:

```
├──DATABASE──database-name───────────────────────────────────────────────────────────┤
```

Command parameters

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

Do not use the CLEAR command if any index administration commands, like an index update, are still active on the index you apply it to, because this could corrupt your index, requiring a complete rebuild.

`LIST` Use **`LIST`** to get information about the current locks held for a specific index or database. If there is an update lock, the command also prints information about the number of documents that have been processed so far.

Note this is only for the time that the lock is holding the index.

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

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

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

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

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

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

Usage notes

When an administration command error message indicates that there is a locking problem, ensure that no conflicting task is running. For example, attempting an **ALTER** command while an **UPDATE** command is running.

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

START command

Starts a daemon that controls the locking of full-text indexes and the automatic updating of full-text indexes on the DB2 server.

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

Authorization

The instance owner must hold DBADM with DATAACCESS authority for the current DB2 instance.

Required connection

This command must be issued from the DB2 database server.

Command syntax

```
►►—START————————————————————————————————————————————————————►◄
```

Command parameters

None

Usage notes

On Windows, the command starts a service `DB2EXT - `*`instance_name`*` [-`*`nodenum`*`]`. You can also start the service(s) using the Control Panel or the **NET START** command. However, you cannot start Net Search Extender through a Terminal Service Client.

For DB2 instances used with partitioned databases, it is highly recommended that the Net Search Extender Instances Services be started using **db2text start** instead of using the normal Windows methods.

Using the services management console you can manually start or stop each of the DB2EXT services for a DB2 instance. However, to keep Net Search Extender in a proper running state it is necessary to start all DB2EXT services and shutdown all DB2EXT services associated with a DB2 instance. Also, during manual start and stop it is necessary that the following sequence must be followed:

Starting
Start the NSE (DB2EXT) service beginning with the lowest-numbered

partition on a host and working your way up to the highest-numbered partition, before starting other DB2EXT services

Stopping
: Stop the NSE (DB2EXT) service beginning with the highest-numbered partition on a host and work your way down to the lowest-numbered partition.

However when you use **`db2text start`** and **`db2text stop`** , this required sequence is automatically followed by the system.

Startup type "Automatic" is not supported for the DB2EXT services. The DB2EXT services have to be started manually by executing **`db2text start`** or by executing **`net start`** for each of the DB2EXT services for the DB2 instance. Do not attempt to run text search operations when the system is not fully started.

If **`START`** fails, there might still be obsolete entries in the scheduler referring to indexes that not longer exist. Edit the file `../sqllib/db2ext/ctedem.dat` and remove any obsolete entries. Rerun the **`START`** command.

After successfully starting Net Search Extender, the process **`ctelock`** (**`ctelock.exe`** on Windows) is active on your system. Several shared resources (shared memory and semaphores) are created and stored in the `/tmp` directory on UNIX machines. These files are required by Net Search Extender and must not be deleted as long as the instance is running. However, if after a **`STOP`**, the **`START`** command is not successful, make sure that all of the old anchor files in the `/tmp` directory have been removed. The `/tmp` directory must be read, write and execute accessible by all.

STOP command

Stops the locking and update services of Net Search Extender.

Authorization

The instance owner must hold DBADM with DATAACCESS authority for the current DB2 instance.

Required connection

This command must be issued from the DB2 database server.

Command syntax

```
>>--STOP--+-------+-------------------------------------------><
          '-FORCE-'
```

Command parameters

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

Usage notes

Stopping the Net Search Extender instance services disables any further use of specific Net Search Extender commands. When restarting the services, you must activate the temporary cache again if you previously used an activated cache with your index.

For DB2 instances used with partitioned databases, it is highly recommended that the Net Search Extender Instances Services be stopped using **`db2text stop`** instead of using the normal Windows methods.

Using the services management console you can manually stop each of the DB2EXT services for a DB2 instance. However, to keep NSE in a proper running state it is necessary to shut down all DB2EXT services associated with a DB2 instance. Also, during a manual stop it is necessary that you stop the NSE (DB2EXT) service beginning with the highest-numbered partition on the host and working your way down to the lowest-numbered partition. However when you use **`db2text stop`** , this required sequence is automatically followed by the system.

After successfully stopping Net Search Extender, the process **`ctelock`** (**`ctelock.exe`** on Windows) is terminated. The shared resources and the anchor files in the `/tmp` directory on UNIX machines are deleted.

Chapter 56. Administration commands for the database administrator

The database administrator can run multiple commands to set up databases for Net Search Extender use.

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

Only the **ENABLE DATABASE** and **DISABLE DATABASE** commands are a variation of the **DB2TEXT** command, although all these commands allow for administration on the database level.

Command	Purpose
"ENABLE DATABASE command"	Enables the current database to create full-text indexes.
"DISABLE DATABASE command" on page 206	Resets preparation work completed by Net Search Extender for a database.
"DB2EXTHL command" on page 208	The **DB2EXTHL** command changes the maximum size of the input parameter of the highlight UDF.

Tip

If no database connection information has been specified as part of the **db2text** command, the **db2text** executable causes an implicit connection to be made to the default subsystem specified in the environment variable **DB2DBDFT**.

ENABLE DATABASE command

Enables a database to create and use full-text indexes on text columns. The **ENABLE DATABASE** command creates the Net Search Extender infrastructure in the database, such as administration tables, views, user-defined functions (UDFs), and stored procedures for searching.

Authorization

A user must have DBADM privilege to execute the **ENABLE DATABASE** command.

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **ENABLE DATABASE** command.

Command syntax

```
►►─ENABLE DATABASE FOR TEXT─┬──────────────────────────────────────┬─►◄
                            └─┤ AUTOGRANT ├─┤ connection-options ├─┘
```

connection-options:

```
├──┬────────────────────────────────────────────────────────────┬────────────┤
   └─CONNECT TO──database-name──┬──────────────────────────────┬─┘
                                └─USER──userid──USING──password─┘
```

Command parameters

CONNECT TO *database-name*
: The name of the database that is a target for this command. You can omit this parameter, if the environment variable **DB2DBDFT** is set and the user is running the command under a user ID with the necessary DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database.

AUTOGRANT
: If this option is specified, an attempt is made to grant DBADM with DATAACCESS privileges to the instance owner, in case the instance owner misses these privileges for this database. For a successful grant of privileges, the user must have SECADM privilege for the database and cannot be the instance owner (a user cannot grant privileges to herself/himself).

  Note: This option is not supported in the stored-procedure interface.

Usage notes

This command prepares the connected database for use by Net Search Extender. It is a mandatory step before you can create a Net Search Extender index on tables/columns in the database.

You can view the database defaults established after running the command by using the DB2EXT.DBDEFAULTS catalog view.

Changes to the database
: This command grants DBADM authority to the DB2 instance owner associated with the DB2 instance of the enabled database.

  The **ENABLE DATABASE** command creates various database objects in the schema DB2EXT, such as Net Search Extender catalogs, UDFs, and stored procedures. After running the command, the following catalog views are available:

```
db2ext.dbdefaults
db2ext.textindexes
db2ext.textindexformats
db2ext.indexconfiguration
```

  These tables are located in the default table space of the database, known as IBMDEFAULTGROUP. This table space is distributed over all the nodes defined in `db2nodes.cfg`

Changes to the file system
: None.

DISABLE DATABASE command

Undoes Net Search Extender changes to a database.

Authorization

A user must have DBADM privilege to execute the **DISABLE DATABASE** command.

Prerequisite

Instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. SECADM must explicitly grant DBADM with DATAACCESS to instance owner before running the **DISABLE DATABASE** command.

Command syntax

```
►►──DISABLE DATABASE FOR TEXT──┬───────┬──┬──────────────────────┬──────────────►◄
                               └─FORCE─┘  └─|connection-options|─┘
```

connection-options:

```
├──┬──────────────────────────────────────────────────────────────┬──────────┤
   └─CONNECT TO──database-name──┬─────────────────────────────┬───┘
                                └─USER──userid──USING──password─┘
```

Command parameters

CONNECT TO *database-name*
: The name of the database that is a target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command under a user ID with the necessary DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database.

FORCE
: Forces the dropping of all Net Search Extender indexes in the database.

Usage notes

This command resets the connected database, so that it can no longer be used by other Net Search Extender commands. If full-text indexes exist in the database, this command fails unless the **FORCE** option is used.

This command does not remove DBADM authority from the DB2 instance owner.

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

However, if some of the indexes can not be completely dropped, there may still be resources that need to be manually cleaned up. These include:
- Files in the index, work and cache directory
- Scheduler entries in `ctedem.dat`
- Where an index was created using the replication capture option, the IBMSNAP_SIGNAL, IBMSNAP_PRUNE_SET, and IBMSNAP_PRUNCNTL entries in the tables of the remote database must be manually deleted. These

entries can be easily identified using `APPLY_QUAL="NSE"||<instance name> and TARGET_SERVER=` *`database_name`* command.

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

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

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

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

Changes to the database

The following modifications made in the database to enable Net Search Extender are deleted:

- The Net Search Extender catalog views in the database.
- All the database objects created by Net Search Extender.

Changes to the file system and shared memory

If you use the **`FORCE`** option, the index files are deleted.

If you use the **`FORCE`** option, the cache is deleted for any activated cache of indexes.

DB2EXTHL command

Changes the maximum size of the input parameter of the highlight UDF.

Purpose

By default, the highlight UDF takes as input a document up to a maximum size of 100 KB and returns a 200 KB CLOB. Depending on the size of the largest document in the database, you can increase the input value to a maximum size of 1 GB.

Authorization

To issue this command successfully, the user must be a DB2 instance owner with DBADM and DATAACCESS authority.

Required connection

This command must be issued from the DB2 database server and requires a **`DB2DBDFT`** environment variable.

Command syntax

```
►►—db2exthl—new-highlight-input-size——————————————————————————►◄
```

Command parameters

new-highlight-input-size

The new result size of the highlight UDF in kilobytes. This is a positive integer < 1048576.

Chapter 57. Administration commands for the text table owner

The table owner can run commands to modify text indexes in tables.

This section describes the syntax of administration commands for the text table owner.

The commands are subcommands of the **DB2TEXT** command. These allow the owner of a table to create and manipulate full-text indexes on columns of the table.

Command	Purpose
"ACTIVATE CACHE command"	Activates the cache so that search operations using the stored procedure are possible
"ALTER INDEX command" on page 211	Changes the characteristics of an index
"CLEAR EVENTS command" on page 215	Deletes index events from an index event table used during index update
"CREATE INDEX command" on page 216	Creates a full-text index
"DEACTIVATE CACHE command" on page 230	Deactivates the cache so that search operations using the stored procedure are no longer possible
"DB2EXTTH command" on page 233	Compiles the thesaurus definition file
"DROP INDEX command" on page 231	Drops a full-text index for a text column
"RESET PENDING command" on page 233	Refreshes text-maintained dependent tables for base tables with the extended staging infrastructure after executing a SET INTEGRITY command.
"UPDATE INDEX command" on page 234	Starts the indexing process based on the current contents of the text columns
"HELP command" on page 238	Displays the list of **DB2TEXT** command options
"COPYRIGHT command" on page 238	Displays the Net Search Extender product and copyright information

Tip:

If no database connection information has been specified as part of the **db2text** command, the **db2text** executable causes an implicit connection to be made to the default subsystem specified in the environment variable **DB2DBDFT**.

ACTIVATE CACHE command

Activates the cached table from either the DB2 user table or the persistent cache files. After completion, search operations using the stored procedure are possible.

This command is only available if the index was created with a **CACHE TABLE** option.

Authorization

The authorization ID of the statement must include at least one of the following privileges:

- CONTROL privilege on the table or nickname on which the text index is defined
- DBADM authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **ACTIVATE CACHE** command.

Command syntax

```
►►──ACTIVATE CACHE FOR INDEX──┬──────────────────┬──index-name──FOR TEXT──────────►
                              └─index-schema-"."─┘

►──┬──────────┬──┬──────────────────────┬──────────────────────────────────────────►◄
   └─RECREATE─┘  └─|connection-options|─┘
```

connection-options:

```
├──┬──────────────────────────────────────────────────────────────┬──────────────┤
   └─CONNECT TO──database-name──┬──────────────────────────────┬──┘
                                └─USER──userid──USING──password─┘
```

Command parameters

index-schema
: The schema of the text index, as specified in the **CREATE INDEX** command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
: The name of the text index, as specified in the **CREATE INDEX** command.

RECREATE
: Applies only to indexes using a persistent cache; an existing cache is deleted. If an update without activation has completed, the persistent cache is automatically reconstructed from the database.

CONNECT TO *database-name*
: The name of the database that is target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

You cannot issue the command if one of the following commands is currently running on the index:

- **UPDATE INDEX**
- **ALTER INDEX**
- **DROP INDEX**
- **CLEAR EVENTS**
- **DEACTIVATE CACHE**

Note: Activation of a cached table may require its recreation from scratch, even though a persistent cache was used. This occurs if an update operation was performed while the persistent cache was deactivated.

The amount of memory used to build the cache is dynamically calculated from the current number of documents and the size of the result columns. Use the **PCTFREE** value to increase the calculated minimal amount of memory by a factor of 100/(100-PCTFREE). The **PCTFREE** value is specified in the **CREATE INDEX** or **ALTER INDEX** command.

Thereby, **PCTFREE** describes the percentage of the allocated cache that is reserved for insert operations while the cache is activated. Note that for each **ACTIVATE CACHE** command, the actual memory size is re-evaluated.

Changes to the file system
Files for implementing the persistent cache are created.

ALTER INDEX command

The ALTER INDEX command changes the characteristics of a full-text index.

Purpose

The command changes the characteristics of a full-text index, for example, the update options and the storage options.

Authorization

The authorization ID of the statement must include at least one of the following privileges:
- CONTROL privilege on the table or nickname on which the text index is defined
- DBADM authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **ALTER INDEX** command.

Command syntax

```
►►──ALTER INDEX──┬─────────────────────┬──index-name──FOR TEXT──────────────────►
                 └─index-schema-"."─┘

►─┬───────────────────────────┬──┬───────────────────┬──────────────────────────►
  └─|update-characteristics|─┘  └─|storage-options|─┘
```

```
►─┬────────────────────────┬──────────────────────────────────────────────►◄
  └─|connection-options|─┘
```

storage-options:

```
├──┬─────────────────────────────┬──┬──────────────────────────────────┬──────►
   └─INDEX DIRECTORY──directory─┘  └─WORK DIRECTORY──workdirectory─┘

►─┬──────────────────────────────────────────────────┬──┬─────────────────────┬──►
  └─CACHE TABLE──┬─PERSISTENT──┬─────────────────┬─┬─┘  └─PCTFREE──percentage─┘
                 │             └─IN──directory─┘ │
                 └─TEMPORARY─────────────────────┘

►─┬───────────────────────────────┬──────────────────────────────────────────┤
  └─MAXIMUM CACHE SIZE──memsize─┘
```

update-characteristics:

```
├──┬──────────────────────────────────────────────────┬───────────────────────►
   └─UPDATE FREQUENCY──┬─NONE──────────────────────┬─┘
                       └─|update-frequency|─┘

►─┬────────────────────────────────┬──┬──────────────────────────────────┬──┤
  └─UPDATE MINIMUM──minchanges─┘  └─COMMITCOUNT FOR UPDATE──count─┘
```

update-frequency:

```
                                                        ┌─,──────┐
├──D──(──┬─*──────────┬──)──H──(──┬─*───────────┬──)──M──(──▼─0...59─┴──)──────┤
         │ ┌─,─────┐  │           │ ┌─,──────┐  │
         └─▼─0...6─┴──┘           └─▼─0...23─┴──┘
```

connection-options:

```
├──┬──────────────────────────────────────────────────────────────────┬────────┤
   └─CONNECT TO──database-name──┬─────────────────────────────────┬─┘
                                └─USER──userid──USING──password─┘
```

Command parameters

index-schema
: The schema of the text index as specified in the **CREATE INDEX** command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
: The name of the text index as specified in the **CREATE INDEX** command.

INDEX DIRECTORY *directory*
: The directory path where the text index is stored. As the directory will contain index data, ensure that the directory has read, write, and execute permissions for the DB2 instance owner user ID.

Note that in a partitioned database environment, this directory has to exist on every partition. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical partitions of a server. Any index files from the previous index directory are deleted.

WORK DIRECTORY *workdirectory*
: Stores temporary files during search and administration operations. You can change the separate work directory independently of a new index directory.

If the directory does not exist, it is created using the DB2 instance owner user ID. If it exists, ensure that the directory has read, write and execute permissions on UNIX platforms for the instance owner.

Note that in a partitioned database environment, this directory has to exist on every partition. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical partitions of a server. Any temporary index files from the previous index directory are deleted.

CACHE TABLE PERSISTENT IN *directory*
: Specifies that the cached table in **CREATE INDEX** is persistent even after a deactivation or system reboot. In either case, this allows for a fast **ACTIVATE CACHE** execution. The persistent cache is stored in the specified directory.

The previously created persistent cache is moved to a new location. This operation always requires a deactivated index.

CACHE TABLE TEMPORARY
: Specifies that the cached result table is now temporary and any previously existing persistent cache has been deleted. Note that this change operation requires a deactivated index.

MAXIMUM CACHE SIZE *memsize*
: Specifies the new maximum size of the cached table to be built during **ACTIVATE CACHE**. Specify the *memsize* parameter in megabytes as a positive integer.

If *memsize* is too small, the **ACTIVATE CACHE** command fails. The actual cache size is calculated during the **ACTIVATE CACHE** command. This change requires a deactivated index.

PCTFREE *percentage*
: Specifies the percentage of the cache held free for additional documents. The *percentage* must be an integer value less than 100 and greater or equal to 0. Note that the previous persistent cache is deleted and that this change requires a deactivated index.

UPDATE FREQUENCY
: Using the following parameters, the index update frequency determines when the update occurs:

- **D.** The day(s) of the week when the index is updated: * (everyday) or 0..6 (0=Sunday)
- **H.** The hour(s) when the index is updated: * (every hour) or 0..23
- **M.** The minute(s) when the index is updated: 0..59
- **NONE.** No further index updates occur. This is intended for a text column in which no further changes are made, or where only manual update are run in the future.

If you do not specify the **UPDATE FREQUENCY** keyword, the frequency settings are left unchanged.

UPDATE MINIMUM *minchanges*

The minimum number of changes allowed for text documents before the index is incrementally updated. If you do not specify the **UPDATE MINIMUM** keyword, the setting does not change.

Note that you can only change the **UPDATE MINIMUM** if you did not create the index using the **RECREATE ON UPDATE** option.

COMMITCOUNT FOR UPDATE *count*

For update processing, you can specify a commitcount. This applies to both the **UPDATE** command and the **UPDATE FREQUENCY** specification, which schedules update processing.

The COMMITCOUNT value is ignored during initial update.

Note that you can only change COMMITCOUNT if you did not create the index using the **RECREATE ON UPDATE** option.

Also note that you cannot change COMMITCOUNT if you did create the index with the **REPLICATION** clause.

CONNECT TO *database-name*

The name of the database that is target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*

Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

You cannot issue the alter index command if one of the following commands is running on the index:

- **ALTER INDEX**
- **CLEAR EVENTS**
- **ACTIVATE CACHE**
- **DROP INDEX**
- **UPDATE INDEX**
- **DEACTIVATE CACHE**

If you create the index with a cache option, you can not use the **ALTER INDEX** command for the index directory when the index is activated. You must first deactivate the cache.

In a partitioned database environment, a text index with cache options is only allowed on a single-partition table space.

Changes to the database

Change Net Search Extender catalog views.

Changes to the file system

- Creation of `NODE<nr>` subdirectories in the index, and work directories
- Moving of index files
- Creation of persistent cache directories
- Moving of persistent cache files

CLEAR EVENTS command

Deletes indexing events from an index's event view. The name of the event view is found in the EVENTVIEWNAME column of the DB2EXT.TEXTINDEXES view.

Authorization

The authorization ID of the statement must include at least one of the following privileges:

- CONTROL privilege on the table or nickname on which the text index is defined
- DBADM authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **CLEAR EVENTS** command.

Command syntax

```
>>--CLEAR EVENTS FOR INDEX--+---------------------+--index-name--FOR TEXT------------>
                            '--index-schema-"."--'

>--+--------------------+--+------------------------+------------------------------><
   '--COMMITCOUNT--count-'  '--|connection-options|--'
```

connection-options:

```
|--+--------------------------------------------------------------------+--------|
   '--CONNECT TO--database-name--+------------------------------------+-'
                                 '--USER--userid--USING--password-----'
```

Command parameters

index-schema
: The schema of the text index as specified in the **CREATE INDEX** command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
: The name of the text index as specified in the **CREATE INDEX** command.

COMMITCOUNT *count*
: An INTEGER value >=0 displays the number of rows deleted in one transaction by DB2.

CONNECT TO *database-name*
: The name of the database that is target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

When you schedule regular updates using the **UPDATE FREQUENCY** option in the **CREATE INDEX** or **ALTER INDEX** commands, regularly check the event table. Use **CLEAR EVENTS** to clean up the event tables, after you have checked the reason for any events indicating an error and removed the source of the error that is mentioned in the event table.

You cannot issue the clear events command if one of the following commands is running on the index:
- **UPDATE INDEX**
- **ALTER INDEX**
- **ACTIVATE CACHE**
- **DEACTIVATE CACHE**
- **DROP INDEX**

CREATE INDEX command

Creates a full-text index on a text column for use in Net Search Extender full-text queries.

Purpose

In a partitioned database environment, a full-text index is created on every partition of the table space the user table is defined on. Subsequent changes to the distribution of the table space are not allowed and will lead to unexpected behavior in the administration commands and during the search process.

Authorization

The authorization ID of the statement must include at least one of the following privileges:

One of:
- DBADM authority
- CONTROL privilege on the table or nickname on which the text index is defined
- INDEX privilege on the table or nickname on which the text index is defined and one of following:
  - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist
  - CREATEIN privilege on the schema, if the schema name of the index refers to an existing schema

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **CREATE INDEX** command.

Command syntax

```
►►──CREATE INDEX──┬──────────────────┬──index-name FOR TEXT ON──────────────►
                  └─index-schema-"."─┘

►──┬─────────────────┬──table-name──────────────────────────────────────────►
   └─table-schema-"."┘

►──┬─(text-column-name)──────────────────────────────────────────────────────►
   └─(──┬────────────────────────┬──function-name-(-text-column-name-)──)─┘
        └─|function-schema "."|──┘

►──┬───────────────────┬──┬────────────────────────────┬─────────────────────►
   └─|attribute-list|──┘  └─|text-default-information|─┘

►──┬──────────────────────────────┬──┬────────────────────┬──────────────────►
   └─|update-characteristics|─────┘  └─|storage-options|──┘

►──┬──────────────────────────────────┬──┬──────────────────────────────────┬─►
   └─|cache-search-result-options|────┘  └─|index-configuration-options|────┘

►──┬──────────────────────┬──────────────────────────────────────────────────►◄
   └─|connection-options|─┘
```

attribute list:

```
                    ┌─','─────────────────────────────────────────┐
                    ▼                                             │
├──ATTRIBUTES──(────SQL-column-expression──┬─────────────────────┬┴──)──────┤
                                           └─AS-attribute-name───┘
```

text-default-information:

```
├──┬─────────────┬──┬─────────────────────┬──────────────────────────────────►
   └─CCSID─ccsid─┘  └─LANGUAGE─language───┘

►──┬───────────────────────────────────────────────┬─────────────────────────┤
   └─FORMAT─format──┬───────────────────────────┬──┘
                    └─|model-information|───────┘
```

model-information:

```
├──DOCUMENTMODEL─documentmodel-name──IN──modelfilepath──────────────────────►

►──┬────────────────────┬────────────────────────────────────────────────────┤
   └─USING CCSID─ccsid──┘
```

update-characteristics:

```
├──┬────────────────────────────────────────────────────┬────────────────────►
   └─UPDATE FREQUENCY──┬─NONE──────────────────────┬────┘
                       └─|update-frequency|────────┘

►──┬─|incremental-update-characteristics|─┬──────────────────────────────────┤
   └─RECREATE INDEX ON UPDATE─────────────┘
```

incremental-update-characteristics:

```
├──┬──────────────────────────────┬──┬────────────────────────────────┬──────────────────►
   └─UPDATE MINIMUM──minchanges─┘  └─REORGANIZE──┬─AUTOMATIC─┬─┘
                                                 └─MANUAL────┘

►──┬───────────────────────────────────┬─────────────────────────────────────────────────┤
   ├─COMMITCOUNT FOR UPDATE──count─────┤
   └─┤capture-table-characteristics├───┘
```

capture-table-characteristics:

```
├──REPLICATION CAPTURE TABLE──┬──────────────────────────────┬─capture-table-name──────►
                              └─capture-table-schema-"."─────┘

►─CONTROL TABLE SCHEMA──capture-control-schema─────────────────────────────────────────┤
```

update-frequency:

```
                                                          ┌─,───┐
├──D──(──┬─*────────┬──)──H──(──┬─*─────────┬──)──M──(──▼─0...59─┴──)──────────┤
         │  ┌─,──┐  │             │  ┌─,───┐  │
         └─▼─0...6─┴┘             └─▼─0...23─┴┘
```

storage-options:

```
├──┬──────────────────────────────┬──┬──────────────────────────────────┬──────────────►
   └─INDEX DIRECTORY──directory─┘  └─WORK DIRECTORY──workdirectory─┘

►──┬───────────────────────────────────────────────┬─────────────────────────────────┤
   └─ADMINISTRATION TABLES IN──table-space-name────┘
```

cache-search-results-options:

```
                      ┌─','──────────────────────────────────────────┐
├──CACHE TABLE──(───▼─SQL-column-expression──┬──────────────────────┬─┴──)──────────►
                                              └─AS──attribute-name──┘

►──┬─PERSISTENT──┬──────────────────┬─┬──┬──────────────────────┬───────────────────►
   │             └─IN──directory────┘ │  └─PCTFREE──percentage──┘
   └─TEMPORARY────────────────────────┘

►─MAXIMUM CACHE SIZE──memsize──────────────────────────────────────────────────────►
```

```
►──┬─────────────────────────────────────────────────────┬───────────────────────►
   └─INITIAL SEARCH RESULT ORDER──(──SQL-order-by-list──)─┘
```

```
►──┬─────────────────────────────────────────────────────┬──────────────────────┤
   └─KEY COLUMNS FOR INDEX ON VIEW──(SQL-columnname-list)─┘
```

index-configuration-options:

```
                              ┌─,────────────┐
                              ▼              │
├──INDEX CONFIGURATION──(────option-value────┴──)────────────────────────────────┤
```

connection-options:

```
├──CONNECT TO──database-name──┬──────────────────────────────────┬───────────────┤
                              └─USER──userid──USING──password────┘
```

Command parameters

index-schema
: The schema of the text index. Use this as a DB2 schema name for the index-specific administration tables. If no schema is specified, the user ID of the DB2 connection is used. Note that the index schema must be a valid DB2 schema name.

index-name
: The name of the index. Together with the index schema, this uniquely identifies a full-text index in a database.

  Note that the index name must be a valid DB2 index name.

table-schema
: The schema for which of the table, nickname, or view the index is created. If no schema is specified, the user ID of the DB2 connection is used.

table-name
: The name of the text table, nickname, or view in the connected database that contains the column the full-text index is created for.

  Note that when the table name does not refer to a DB2 base table, there are the following restrictions:

  - A view only allows a stored procedure or table-valued function search. Therefore, you must specify the key columns for the index or views using the **KEY COLUMNS FOR INDEX ON VIEW** clause.
  - For incremental index updates on nicknames without capture tables, a log table is created. If any changes occur to the data in the nickname table or view, you must manually specify the log table. With base tables this is done automatically, so the user **must not** touch the log table.
  - The DB2 predicates CONTAINS, SCORE, and NUMBEROFMATCHES are only allowed for indexes on base tables or nicknames, but not on views.
  - Indexes on views are only allowed if you specify cache-search-result options in the command.

- The extended text-maintained staging infrastructure complementing the trigger-based log to support incremental updates can be applied to partitioned and non-partitioned base tables, however, it is not supported for views or nicknames.

text-column-name
The name of the column containing the text used for creating the full-text index. The column must have one of the following data types:
- CHAR (FOR BIT DATA)
- VARCHAR (FOR BIT DATA)
- LONG VARCHAR (FOR BIT DATA)
- CLOB
- DBCLOB
- BLOB
- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC
- XML

If the column data type is none of these, specify a transformation function using ***function-schema.function-name*** to convert the column type to a supported one.

Note that several indexes on the same columns are allowed, but **only** under either one of the following conditions:

The index is created on a view
Therefore, you can not use the index in the `CONTAINS`, `SCORE`, or `NUMBEROFMATCHES` search arguments.

The index is created on a table
If all the indexes are synchronized, they have identical properties on the same column in the following **`CREATE INDEX`** command details:
- Function name and schema
- ATTRIBUTES
- CCSID
- LANGUAGE
- FORMAT
- DOCUMENTMODEL
- INDEX CONFIGURATION

Therefore, it does not matter which index is chosen by the `CONTAINS`, `SCORE`, or `NUMBEROFMATCHES` arguments.

*function-schema***.***function-name*
The schema and the name of a user-defined function used to access text documents that are in a column of an unsupported type. The function performs a column type conversion, using as input parameter an arbitrary column type. It returns the value as one of the Net Search Extender supported types.

`ATTRIBUTES` **(***SQL-column-expression* **AS** *Attribute-name***, ...)**
Ensures that the content of a column expression is indexed in addition to the text column. This content can then be searched by the **`ATTRIBUTES`**

clause in a search statement. The SQL-column expressions have to be defined using unqualified column names of the table on which the index is created. The only data type allowed is DOUBLE. Cast operators can be used in the column expressions, but implicit casting of DB2 is **not** possible. The attribute-names must follow the rules for attribute-names in document models and must be distinct from all attribute names in the indexes model-definition file.

Determine the attribute names for expressions by using the following rules:

- If explicitly named by the SQL **AS** clause in the column expression, use the specified name. An example would be:

  ```
  ATTRIBUTES (C1+C2 AS myname)
  ```

- If a column of the specified table is used without **AS**, the name of the column is used. For example:

  ```
  ATTRIBUTES (C1)
  ```

- If an expression is used without **AS** and it does not refer to a named column, **CREATE INDEX** reports an error.

For example:

```
ATTRIBUTES (CAST(JULIAN_DAY(date) AS DOUBLE) as day, (price1+price2)/2 as avg_price)
```

Note that attributes that are not enclosed by single quotation marks are mapped to uppercase, and must be specified in uppercase during search.

CCSID *ccsid*

The Coded Character Set Identifier is used when indexing text documents. The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='CCSID'.

Only set a CCSID if the data type of the column is binary.

LANGUAGE *language*

The language parameter specifies the language of the stop word dictionary that is selected if the index configuration value IndexStopWords is set to 0 (ignore stop words during indexing). This parameter must always be set for Thai (TH_TH) to enable Thai word breaking, and Turkish to correctly distinguish the use of the dotted and dotless "i".

FORMAT *format*

The format of text documents in the column, for example, HTML. This information is necessary for indexing documents.

For structured document formats, you can specify information in a document model file. If no document model is specified, the text of the document is indexed using a default document model.

If the format keyword is not specified, the default value is taken from the DB2EXT.DBDEFAULTS view where the DEFAULTNAME='FORMAT'. The initial default set by Net Search Extender is TEXT. For data type XML, the FORMAT XML specifier is mandatory.

DOCUMENTMODEL *documentmodel-name* **IN** *modelfilepath*

The *modelfilepath* specifies the location of a model file. The *modelfilepath* must be a fully qualified path. The model file contains a model definition for the format in the **FORMAT** clause. It must be readable by the DB2 instance owner. A document model enables you to index and search specific sections of a document. You can define markup tags and section names in a document model. A document model is bound to a document format that supports HTML, XML, or GPP structures. You can only specify one document model in a model file.

Note that because the document model is only read during the **CREATE INDEX** command, any later changes are not recognized for this index.

Note that in a partitioned database environment, a shared file system must be used to ensure that the *modelfilepath* is accessible on every node on Linux or UNIX operating systems. However, on Windows operating systems, the *modelfilepath* for document model files must be set to local path available on each node.

USING CCSID *ccsid*
Specify a CCSID to interpret the contents of the model file. The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='MODELCCSID'.

UPDATE FREQUENCY
The index update frequency determines when the update occurs. If changes to the user table are less than that specified by the **UPDATE MINIMUM** option, the index is not updated. If you do not specify the **UPDATE FREQUENCY**, the default NONE is used, so that no index updates are made. This is useful when you expect no further changes to a text column or want to have manual control over the update process.

- **D.** The day(s) of the week when the index is updated: * (everyday) or 0..6 (0=Sunday)
- **H.** The hour(s) when the index is updated: * (every hour) or 0..23
- **M.** The minute(s) when the index is updated: 0..59
- **NONE.** No further index updates are made. The update must be started manually.

The default value is from the DB2EXT.DBDEFAULTS view where DEFAULTNAME='UPDATEFREQUENCY'.

If you decide not to use the **UPDATE FREQUENCY** parameter to schedule automatic index updates, you can use operating system functions, for example crontab, instead.

UPDATE MINIMUM *minchanges*
The minimum number of changes required to text documents before the index is updated based on the **UPDATE FREQUENCY** settings. Only positive integer values are allowed. The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='UPDATEMINIMUM'.

Note that this value is ignored if the **DB2TEXT UPDATE** command is executed manually. This option cannot be used with the **RECREATE INDEX ON UPDATE** option, as the number of changes is not available without a log table and triggers for incremental update.

For partitioned database environments, the **UPDATE MINIMUM** is checked on every partition.

REORGANIZE AUTOMATIC | MANUAL
Updates performed based on the update frequency settings will only reorganize the index if **REORGANIZE AUTOMATIC** is specified. This step is completed automatically according to the value of `select REORGSUGGESTED from DB2EXT.TEXTINDEXES` after the update.

REORGANIZE MANUAL can only be performed with a manual **UPDATE** command, using the **REORGANIZE** option.

If the **REORGANIZE** clause is omitted, the default is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='AUTOMATICREORG'.

REPLICATION CAPTURE TABLE *capture-table-schema***.***capture-table-name* **CONTROL TABLE SCHEMA** *capture-control-schema*

For incremental update processing, the specified replication capture table is taken instead of a log table or text-maintained staging table that is otherwise created for the index. Therefore, `schemaname`, `tablename`, and the replication capture table name relate to objects in the local DB2 (federated) database.

The capture-control-schema is the schema name of the replication control tables, for example IBMSNAP_PRUNE_SET on the local DB2. The replication control tables must be available as nicknames on the local DB2 system after setting up the replication.

At minimum, there must be nicknames available for the following capture control tables:

- IBMSNAP_SIGNAL
- IBMSNAP_PRUNE_SET
- IBMSNAP_PRUNCNTL
- IBMSNAP_REGISTER
- IBMSNAP_REG_SYNC (Non DB2 database remote sources only)

As DB2 Replication Center does not automatically guarantee to create local nicknames for a remote capture table and capture control tables, this can be a manual task. The task is similar to creating a nickname for the table that the text index is to be created on.

The column names of primary key columns in the user table nickname and the capture table nickname must match. In addition, the names of the columns IBMSNAP_OPERATION, IBMSNAP_COMMITSEQ and IBMSNAP_INTENTSEQ must not be changed in the capture table nickname.

After index creation, the column names DB2EXT.TEXTINDEXES(LOGVIEWNAME) and DB2EXT.TEXTINDEXES(LOGVIEWSCHEMA) both refer to the local name of the replication capture table.

As Net Search Extender does not require all the functionality of the DB2 Replication Center, the Change Data table (CD) or the Consistent-Change Data (CCD) table must obey following rules:

- Use change capture registration and not the full refresh copying option.
- No horizontal subsetting of capturing changes is allowed. For example, by triggers. See Chapter 6, "Subsetting data in your replication environment" in the *DB2 Replication Guide and Reference, Version 8*.
- Registering changes for a subset of columns is only allowed if the primary key columns, the text column, and all columns involved in the attribute and cache table expressions of the **DB2TEXT CREATE INDEX** command are included.
- The primary key columns must be included in the capture table. Note that the after-image is sufficient.
- The capture tables must not be condensed. For each primary key there must be one entry with the latest data. However, Net Search Extender requires a full history to be available.

- The table must use the D/I option. This enables updates to primary keys on the source table to be transformed into a pair of inserts/deletes.

For supported remote source versions in DB2 V9.7 see the Technical Support pages.

Note:

Ensure that the correct source table name is inserted into the registration table. Depending on the type of remote DBMS, the remote tablename or the local nickname must be used:
- DB2: remote table name (the tablename on the remote server)
- Non DB2: local nickname (the corresponding nickname in the federated DB2 database)

A user mapping must exist for the local user to access the remote data source via nicknames and the remote user must have control privilege on the tables.

If the DB2 instance owner user ID is different from the local user ID, an additional user mapping for the DB2 instance owner user ID is needed.

The specified base table name must not be a view on a nickname. This is because a view can be over several nicknames and several CD and CCD tables can also be involved. As only one CD or CCD table can be specified in the replication capture clause, a view on nicknames can not be supported. In addition, nicknames on a remote views can not be supported because the primary key is missing.

The CD or CCD table must be a nickname and can not be a view or an alias.

The **`COMMITCOUNT`** option cannot be used when the **`REPLICATION CAPTURE TABLE`** option is used with **`create index`** command.

`COMMITCOUNT FOR UPDATE` *count*

For **incremental** update processing a commitcount can be specified. If not specified, a default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='COMMITCOUNT'.

The **`COMMITCOUNT FOR UPDATE`** value for the index can be found in `DB2EXT.TEXTINDEXES.COMMITCOUNT`. This can be changed for each index using the **`ALTER INDEX`** command. It also applies to the scheduled update processing according to the **`UPDATE FREQUENCY`** specification. A value of 0 means that the update is completed in one transaction, with values >0 specifying the number of documents to process in one transaction. You are recommended not to use a non-zero `COMMITCOUNT` or, if you must, to set the value high enough so that the number of immediate commits during a single incremental update remains very small. Each commit involves moving index files and cleaning up the index log file which takes a considerable amount of time particularly if performed repeatedly. The `COMMITCOUNT` applies across all changes for documents as listed in the log table as well as the extended text-maintained staging infrastructure, if configured for the index.

If `COMMITCOUNT` is not set, then the `NUMBER_DOCS` parameter from the `db2ext.textindexes` is not updated. Therefore, to view the number of documents already processed during the updating process, use the `CONTROL LIST` command.

`RECREATE INDEX ON UPDATE`
This does not allow incremental index updates, but recreates the index when an update operation is performed (by command or scheduled update).

Note: If you specify **`RECREATE INDEX ON UPDATE`**:
- no triggers are created on the user table,
- no log table is created, and
- you cannot configure the extended text-maintained staging infrastructure for the index.

`INDEX DIRECTORY` *directory*
The directory path in which the text index is to be stored. As the directory will contain index data, ensure that the directory has read/write and execute permissions for the DB2 instance owner user ID.

The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME=INDEXDIRECTORY'. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server.

Note that in a partitioned database environment, this directory has to exist on every physical partition.

`WORK DIRECTORY` *directory*
A separate work directory may be specified optionally, that will be used to store temporary files during index search and administration operations. The directory must exist and have read/write and execute permissions for the DB2 instance owner user ID.

The default value is taken from the DB2EXT.DBDEFAULTS view, where DEFAULTNAME='WORKDIRECTORY'. A subdirectory, `NODE<nr>`, is created under the directory to distinguish indexes on logical nodes of a server.

Note that in a partitioned database environment, this directory has to exist on every physical partition.

If a **`WORK DIRECTORY`** is not specified, a directory called `work` is created under **`INDEX DIRECTORY`**.

If a **`WORK DIRECTORY`** is specified, it is strongly recommended that it is located on the same physical file system as the index directory. Not following this recommendation causes massive index update performance degradation, because files from the work directory have to be copied physically into the index directory instead of being able to rename them.

Note: Files created in the **`INDEX DIRECTORY`** and **`WORK DIRECTORY`** will follow umask restriction set for the instance owner. Make sure that these umask restrictions must allow for group read/write access for the fenced user.

`ADMINISTRATION TABLES IN` *table-space-name*
The name of the regular table space for administration tables created for the index. The table space must exist. If not specified, the table space of the user table is chosen, if the index is created on a base table.

In case of a nickname or a view, a default table space is chosen by DB2.

When creating text indexes on views, nicknames, or text indexes for stored procedure search in a partitioned database environment, the table space for administration tables must be specified on a single node and must be called explicitly on this node.

To ensure that you connect to the correct node, use the **DB2NODE** environment variable. Note that the **ADMINISTRATION TABLES IN** clause is mandatory for creating indexes on range partitioned table. Else, the **CREATE INDEX** command returns an error. See CTE0150E for more information about the error message.

CACHE TABLE (*SQL-column-expression-list***)**

A cached table is built in addition to the index, which consists of the specified column expressions. This cache is used to return the result set via a stored procedure search without joining full-text search results with a DB2 table. Note that a regular DB2 search using the full-text index with the CONTAINS function is always possible.

Define the SQL-column expressions using unqualified column names of the table the index is created on. The allowed SQL-column expression types are all built-in and user-defined distinct types. The column names in the result set are determined using the following rules:

- If explicitly named by the SQL AS clause in the column expression, the specified name is used. For example:
  ```
  CACHE TABLE (C1+C2 AS myname)
  ```
- If a column of the specified table is used without the AS clause, the name of the column is used. For example:
  ```
  CACHE TABLE(C1)
  ```
- If an expression is used without AS, and which does not refer to a named column, **CREATE INDEX** reports an error.
- No duplicate column names are allowed.

CLOB data types are not supported as cache data types. You need to cast these to VARCHARS.

Note: Note that if the column names of the result set are not disjunct, the CREATE INDEX command returns an error. Also note that the cached table is not implicitly activated after creation, for example search by stored procedure is not possible until DB2TEXT ACTIVATE CACHE is performed.

This option may be used in a partitioned database environment only if the user table is stored in a single-partition table space. It cannot be used if the default configuration for the text-maintained staging infrastructure is set to ON for the table and the configuration is not manually disabled for the text index.

PERSISTENT IN *directory*

Specifies that the cache is also created persistently. A persistent cache can be reactivated more rapidly after a deactivation or a system restart than a non-persistent cache. The persistent cache is stored in the specified directory.

Note that if the directory is not specified, the default is taken from the db2ext.dbdefaults view, where DEFAULTNAME='CACHEDIRECTORY'.

TEMPORARY

Specifies that the cache is not stored persistently. If neither PERSISTENT or TEMPORARY is specified, the default is taken from the

DB2EXT.DBDEFAULTS view, where DEFAULTNAME='USEPERSISTENTCACHE'.

MAXIMUM CACHE SIZE *memsize*

Specifies the maximum size of the cached table to be built during **DB2TEXT ACTIVATE CACHE**. The *memsize* parameter must be specified in megabytes as a positive integer. There is no default value for *memsize*. If the integer is too small, the ACTIVATE CACHE command will fail. The actual cache size is calculated during the **ACTIVATE CACHE** command.

The limit for the maximum cache size on the different 32-bit platforms is:

- Windows: 1024 MB (1 GB = 1073741824 bytes)
- Linux: 2048 MB (2 GB = 2147483647 bytes)

On 64-bit installations, the maximum cache size limit depends on the available memory.

PCTFREE *percentage*

Specifies the percentage of the cache to be held free for additional documents. The percentage must be an integer value lower than 100 and greater or equal to 0. If not specified, the default is taken from the `db2ext.dbdefaults` view, where `DEFAULTNAME='PCTFREE'` The default is 50%.

INITIAL SEARCH RESULT ORDER (*SQL-order-by-list***)**

Specifies the order used for retrieving the user table contents during initial indexing. When using this option and skipping the dynamic ranking of full-text search results, the documents are returned in their indexing order, as stored in the cached result table.

You can work with presorted indexes and predefined search result ordering only if you are using the Stored Procedure search interface. For example: `INITIAL RESULT ORDER(length(column1) asc, column2+column3 desc)`.

Predefined search result ordering is not implemented for the SQL scalar search functions and the table valued function.

Note: The index order can **not** be ensured for the new or changed documents after incremental update.

KEY COLUMNS FOR INDEX ON VIEW (*SQL-columnname-list***)**

If indexes on views are created, the **KEY COLUMNS FOR INDEX ON VIEW** clause must be specified, otherwise it MUST NOT be specified. The list of column names specifies the columns that UNIQUELY identify a row in the view.

As this uniqueness cannot be checked by DB2 as in case of primary keys, the user is responsible to ensure the equivalent uniqueness. The specified columns are used to fill the PK01 column of the log table for the index.

INDEX CONFIGURATION (*option-value***), ...**

These are the index configuration values. The default values are underlined.

Option	Values	Description
TreatNumbersAsWords	0 or 1	Interprets sequences of digits as separate words if set to 1, even if they are adjacent to characters. The 0 default means that, for example, tea42at5 is considered as one word.

Option	Values	Description
IndexStopWords	0 or 1	Ignores or considers stop words during indexing. The default 1 indexes all text including stop words. Currently, the stopword list is an UCS-2 file `<language>.tsw` in directory `<instance>/sqllib/db2ext/resources`. Changes to this file have **no effect** after index creation. Also note that `<language>` is the **`LANGUAGE`** value from the **`CREATE INDEX`** command.
UpdateDelay	seconds	Specifies the duration in seconds for incremental update without capture tables. Only entries older than this duration will be taken from the log table. This is to avoid lost updates. For example, document changes that are not reflected in the index in transaction scenarios where user transactions interfere with update commands. Therefore, the UpdateDelay parameter should be set to the maximum duration of a user write transaction on the table the index was created on.
IgnoreEmptyDocs	0 or 1	If IgnoreEmptyDocs is set to 1, empty documents (with content length 0, or a null value) are not represented in the index. If this option is used, and the document content is null (empty), the next incremental update will delete the documents from the index.
AuxLog	ON or OFF	Controls the creation of the additional log infrastructure to capture changes that are not recognized by a trigger. The default setting for range partitioned tables is ON, else it is OFF. The default value can be changed in the default table with a setting for AuxLogNorm for non-range-partitioned tables and AuxLogPart for range-partitioned tables.

CONNECT TO *database-name*

The name of the database that is target for this command. You can omit this parameter, if DB2DBDFT is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*

Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Changes to the database

- Change Net Search Extender catalog views.
- Create an index log table in the specified table space. This is only if the **`RECREATE INDEX`** option is not specified and the capture table is not specified.
- Create an index event table in the specified table space.
- Deferred to first update: Creation of triggers on the user text table (only if **`RECREATE INDEX`** is not specified and no capture table is used)
- If a replication capture table is used, the following change is made to the capture control tables:
  – an insert into the IBMSNAP_PRUNCTNL and IBMSNAP_PRUNE_SET tables

  The entries in these tables are uniquely identified by the columns:

    - APPLY_QUAL='NSE' || <DB2 instance running NSE>
    - SET_NAME= <internal index identifier>
    - TARGET_SERVER=<DB2 database name target to DB2TEXT operation>
  - If the text-maintained staging infrastructure is configured for the index, an anchor table and a staging table are created to capture change information for documents.

Changes to the shared memory
Deferred to ACTIVATE execution: If CACHE TABLE clause is used, a cache for the result table is built in *shared memory*.

Changes to the file system

- Subdirectories `NODE<nr>` are created under index, work and cache directories.
- The directory *`internal_index_name`* is created under *`indexdirectory`*`/NODE<nr>` where *indexdirectory* refers to the corresponding parameter of this command and `NODE<nr>` is related to the partition number in a partitioned database environment.

Usage notes

Creation of a full-text index requires a primary key on the user table. Since DB2 Net Search Extender Version 9.1, a multi-column DB2 primary key can be used without type limitations. However, to use the table-valued search, no compound primary key is allowed.

The number of primary key columns is limited to 62, the total length of all primary key columns is limited to 1007 bytes for table spaces with pagesize 4k, 2031 bytes for table spaces with pagesize 8k, 4079 bytes for table spaces with pagesize 16k and 4096 bytes for table spaces with pagesize 32k. Note that if the primary key consists of more than one column, the mentioned limits must be reduced by 2 bytes for each additional column.

- The total size of the SQL expressions for ATTRIBUTES, CACHE TABLE and INITIAL SEARCH RESULT ORDER must not exceed 24 K-bytes.
- Initial index updates are always done as one logical transaction, there is no commitcount in this case.

On Windows platforms, **`modelfilepath`** has to be on the local file system. Windows network mapped drives are not supported.

Note:

If a primary key consists of more than one column , make sure that the order of columns mentioned in the primary key definition are exactly in the same order as they are in **`CREATE TABLE`** statement.

After creating the index, the length of primary key columns or the view key columns must not be changed by **`ALTER TABLE`** commands.

The synchronization between user table, full-text index and the cached result table is completed during the **`update index`** command.

DEACTIVATE CACHE command

Releases a cached table. A persistent cache is kept to be reused on the next **ACTIVATE** command. Until the next activation, search operations via the stored procedure are no longer possible on the deactivated cache.

Authorization

The authorization ID of the statement must include at least one of the following privileges:

- CONTROL privilege on the table or nickname on which the text index is defined
- DBADM authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **DEACTIVATE CACHE** command.

Command syntax

```
►►──DEACTIVATE CACHE FOR INDEX──┬──────────────────┬──index-name──FOR TEXT──────►
                                └─index-schema-"."─┘

►──┬──────────────────────┬──────────────────────────────────────────────────►◄
   └─|connection-options|─┘
```

connection-options:

```
├──┬──────────────────────────────────────────────────────────────┬──────────┤
   └─CONNECT TO──database-name──┬──────────────────────────────┬─┘
                                └─USER──userid──USING──password─┘
```

Command parameters

index-schema
: The schema of the text index as specified in the **CREATE INDEX** command. If no schema is specified, the userid of the DB2 connection is used as schema name.

index-name
: The name of the text index as specified in the **CREATE INDEX** command.

CONNECT TO *database-name*
: The name of the database that is target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

Note that this command could not be issued when one of the following commands is running on the index:

- **ACTIVATE CACHE**
- **DEACTIVATE CACHE**
- **UPDATE INDEX**
- **ALTER INDEX**
- **DROP INDEX**
- **CLEAR EVENTS**

Note: After deactivation of a persistent cache, the cache is made inaccessible for search by the stored procedure. However, this can be used for fast ACTIVATE, unless an update was performed in the meantime.

In this case, the persistent cache is automatically recreated from scratch using the **ACTIVATE CACHE** command.

DROP INDEX command

Drops a full-text index for a text column. If the cache for the index is activated, it is deleted using this command.

Authorization

The authorization ID of the statement must include at least one of the following privileges:

- CONTROL privilege on the table or nickname on which the text index is defined
- DBADM authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **DROP INDEX** command.

Command syntax

```
►►──DROP INDEX──┬──────────────────┬──index-name──FOR TEXT──────────────►
                └─index-schema-"."─┘

►──┬──────────────────────┬──────────────────────────────────────────►◄
   └─┤connection-options├─┘
```

connection-options:

```
├──┬──────────────────────────────────────────────────────────────┬──┤
   └─CONNECT-TO──database-name──┬──────────────────────────────┬─┘
                                └─USER──userid──USING──password─┘
```

Command parameters

index-schema
: The schema of the text index as specified in the **CREATE INDEX** command. If no schema is specified, the userid of the DB2 connection is used as the schema name.

index-name
: The name of the index as specified in the **CREATE INDEX** command. With the index schema, it uniquely identifies the full-text index in a database.

CONNECT TO *database-name*
: The name of the database that is target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*
: Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

The index is deleted, irrespective of the activation status of its cached table.

Note that the command must not be issued when one of the following commands is running on the index:

- **UPDATE INDEX**
- **CLEAR EVENTS**
- **ALTER INDEX**
- **ACTIVATE CACHE**
- **DEACTIVATE CACHE**
- **DROP INDEX**

Note: Indexes must be manually dropped before or after the user table in DB2 is dropped. If not, the index directories are not correctly cleaned up.

Changes to the database

- Change Net Search Extender catalog views
- Drop the DB2 index
- Drop the index log, staging and event tables
- Delete triggers on the user text table

When using the replication capture tables, entries in the IBMSNAP_PRUNE_SET and IBMSNAP_PRUNCTRNL tables are removed.

Changes to the shared memory

The cached table is deleted.

Changes to the file system

- The directory `internal_index_name` is deleted in the index and the work directories of the dropped index
- Delete a persistent cache for the index

DB2EXTTH command

This independent utility compiles a thesaurus definition file. After running the thesaurus compiler, the THESAURUS-related features of the search argument syntax can be used.

Authorization

None. This command is not necessarily restricted for the table owner, but makes only sense in the context of querying.

Command syntax

```
►►──db2extth──┬──┬───────┬── -ccsid──code page── -f──definition-file-name──┬──────►◄
              │  └─quiet─┘                                                  │
              ├─ -h─────────────────────────────────────────────────────────┤
              ├─ -H─────────────────────────────────────────────────────────┤
              ├─ -?─────────────────────────────────────────────────────────┤
              └─ -copyright─────────────────────────────────────────────────┘
```

Command parameters

-f *definition-file-name*
: The name of the file containing the thesaurus definition. The file name must contain either the absolute path or the relative path to the file. The file name is restricted to 8+3 characters, the extension being optional.

  The thesaurus dictionary is generated in the same directory as the definition file and has the same name. The only difference is that the dictionary has the following extensions: wdf, wdv, grf, grv, MEY, ROS, NEY, SOS, and Ik*n*, where *n* is a digit. Note that if existing thesaurus files have the same name, they are overwritten.

-ccsid *code page*
: The code page in which the thesaurus definition file is written.

-quiet Output information is not displayed.

-copyright
: Returns the internal build number of the product. Use this number when reporting problems.

-h | -H | -?
: Displays help information.

Usage notes

Use this command to compile a thesaurus definition file into a binary thesaurus definition format.

RESET PENDING command

When you use the extended text-maintained staging infrastructure, certain commands cause the staging table to go into pending mode, which blocks other database or text search operations. You can use the **RESET PENDING** command to execute a set integrity for all text-maintained staging tables associated with a given table. You do not have to find all text indexes and associated staging tables to execute a **SET INTEGRITY** command for each table.

After detaching a data partition, you must run the **RESET PENDING** command to update the staging table content.

Authorization

You must have the CONTROL privilege on the table.

If the SECADM modifies the access privileges for the text-maintained staging table or tables associated with a base table, you still require the privilege to execute set integrity statement for the text-maintained staging table or tables.

Required connection

This command must be issued from the DB2 database server.

Command syntax

```
►►──RESET PENDING FOR TABLE table-schema.table-name FOR TEXT──────────────────────►

►─┬──────────────────────┬──────────────────────────────────────────────────────►◄
  └─|connection-options|─┘
```

Connection-options:

```
├──┬──────────────────────────────────────────────────────────────┬──────────────┤
   └─CONNECT TO──database-name──┬──────────────────────────────┬──┘
                                └─USER──userid──USING──password─┘
```

Command parameters

Table-name
: The name of the table for which the text-maintained staging infrastructure has been added and which requires integrity processing.

Table-schema
: The schema of the table for which a command was executed that results in a pending mode for dependent tables.

Usage notes

Use the **RESET PENDING** command after running a command that causes dependent tables to be put into pending mode, such as a **LOAD** command with the **INSERT** parameter, or a command that requires a set integrity statement to refresh dependent tables, such as **ALTER TABLE** ... **DETACH**.

UPDATE INDEX command

Checks disk space required for the update operation before the actual index update is started. If the check is successful it will continue with the index update process.

Purpose

The disk space required is calculated for a single update process and update index would be terminated if the estimated space is not available. Refer Part 7, "Planning considerations," on page 55

The indexing process is started by bringing the index up to date to reflect the current contents of the text columns with which the index is associated.

While the update is being performed, search using the CONTAINS predicate is possible. For an index with an activated cached result table, search by stored procedure is also possible during update. However, columns in the cached table may show new values, even though the changed text is not yet committed to the full-text index.

Using the **RECREATE INDEX ON UPDATE** option in the **CREATE INDEX** command will clear the index before recreation. Until completion of the update, empty results will be returned.

Authorization

The authorization ID of the statement must include at least one of the following privileges:
- CONTROL privilege on the table or nickname on which the text index is defined
- DATAACCESS authority

Prerequisite

The instance owner must hold DBADM with DATAACCESS authority. The SYSADM no longer holds SECADM nor DBADM privilege in Version 9.7. The SECADM must explicitly grant DBADM with DATAACCESS to the instance owner before running the **UPDATE INDEX** command.

Command syntax

```
►►──UPDATE INDEX──┬─────────────────────┬──index-name──FOR TEXT──┬────────────┬──────►
                  └─index-schema-"."────┘                        └─REORGANIZE─┘

►──┬─────────────────────┬──┬──────────────────────┬──┬───────────────────┬──────────►
   └─COMMITCOUNT──count──┘  └─USING UPDATE MINIMUM─┘  └─SKIPDISKSPACECHECK─┘

►──┬──────────────────────┬───────────────────────────────────────────────────────►◄
   └─|connection-options|─┘
```

connection-options:

```
├──┬─────────────────────────────────────────────────────────────────────┬──────────┤
   └─CONNECT TO──database-name──┬─────────────────────────────────────┬─┘
                                └─USER──userid──USING──password───────┘
```

Command parameters

index-schema
: The schema of the text index. This is specified in the **CREATE INDEX** command. If no schema is specified, the user ID of the DB2 connection is used.

index-name
: The name of the text index. This is specified in the **CREATE INDEX** command.

REORGANIZE

If a text column is frequently updated, then subsequent updates to the index can become inefficient. To make the update process efficient again, reorganize the index. Use the DB2EXT.TEXTINDEXES view to determine if an index needs to be reorganized.

Use the **REORGANIZE AUTOMATIC** option of the **CREATE INDEX** command to avoid manually checking and reorganizing the index.

Note: The reorganization process takes place after a regular update.

USING UPDATE MINIMUM

Uses the **UPDATE MINIMUM** settings from the **CREATE INDEX** command and starts an incremental update only if the specified number of changes was reached. If the text-maintained staging infrastructure is configured, the number of changes is combined for this determination. The default is to unconditionally start the update.

For distributed databases, the **UPDATE MINIMUM** is checked on every partition.

COMMITCOUNT *count*

An INTEGER value >=0 determines the number of documents processed in one transaction by the search engine and by DB2 for incremental index updates. You can change this value using the **ALTER INDEX** command.

However, for initial updates, such as the first update after the **CREATE INDEX** command, or any update with **RECREATE INDEX ON UPDATE** option, there is only one logical transaction which ignores COMMITCOUNT.

Using a non-zero COMMITCOUNT is not recommended due to it's massive negative impact on indexing throughput.

SKIPDISKSPACECHECK

Specifying this option skips the disk space check for an incremental index update. By default the disk space will be checked for each update index operation.

CONNECT TO *database-name*

The name of the database that is the target for this command. You can omit this parameter, if **DB2DBDFT** is set and the user is running the command on the server. Note that the user ID must have the required DB2 authorizations.

USER *userid* **USING** *password*

Use a *userid* and *password* to connect to the database. If not specified, a connection is attempted from the current user ID without a password.

Usage notes

This command runs synchronously. It starts the update processing on all required DB2 logical/physical partitions in a partitioned database environment. The duration depends on the number of documents to indexed and the number of documents already indexed. The status of the update can be seen through a view that is created for each index. The name of this view can be retrieved from DB2EXT.TEXTINDEXES in column EVENTVIEWNAME.

There are two options to view the number of committed documents that have been processed. To determine how many documents have been committed to the index, use the DB2EXT.TEXTINDEXES (NUMBER_DOCS) view. Use the event view associated with the index for information about starting, committing changes, and finishing update processing.

To view the number of documents that have been processed, while an index update is still active, use the **CONTROL LIST ALL LOCKS FOR INDEX** command.

Note: The views only display information from the connected partition.

For incremental updates on a base table with multiple physical partitions, the time on each partition must be synchronized. If the times are not synchronized, updates may be lost or may not occur at all.

You cannot issue the command if one of the following commands is running on the index:

- **CLEAR EVENTS**
- **ALTER INDEX**
- **DROP INDEX**
- **ACTIVATE CACHE**
- **DEACTIVATE CACHE**
- **UPDATE INDEX**

After updating an index with a deactivated persistent cached result table, the persistent cache is deleted, such that the next **ACTIVATE CACHE** command recreates it based on the database content.

If the user interrupts this command, all processes involved in the update function will stop. If COMMITCOUNT was used in an incremental update, some updates might have been committed and are visible in the index, while others may require a new update command. Forcefully interrupting the update index process has the potential to corrupt the index.

To stop the automatic updating of an index, look for the DB2 instance owner process running the update index command on the partition used for update services. Stop this process and the update processing on all the partitions.

Note: As the command works in two separate phases for index creation on all partitions and initial index updates, issue a **db2text drop index** command to ensure that the index is not partly available. If this command is not issued, the next update, which can be triggered by a manual update command or the update frequency option, would perform a complete re-indexing to ensure a consistent state.

Changes to the database

- Inserts to the event table
- Delete from the index log table, and text-maintained staging table for the index, if configured

When using the replication capture tables, the following changes are made to the database.

- A signal is added to the IBMSNAP_SIGNAL table before starting the initial update
- The synchpoint of IBMSNAP_PRUNE_SET is changed after the incremental update

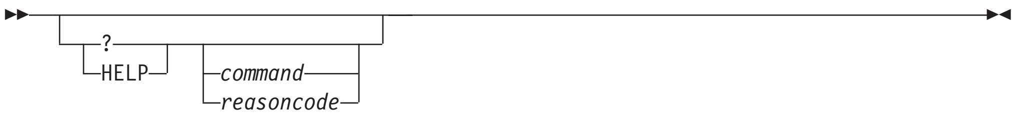
HELP command

Displays the list of available **DB2TEXT** commands, or the syntax of an individual **DB2TEXT** command.

Authorization

None

Command syntax



Command parameters

HELP | ?
: Provides help for the specified command or reason code.

command
: The first keywords that identify a **DB2TEXT** command:
- ENABLE
- DISABLE
- CREATE
- DROP
- ALTER
- UPDATE
- CLEAR
- START
- STOP
- CONTROL
- ACTIVATE
- DEACTIVATE
- RESET PENDING

reasoncode
: Reason code from a Net Search Extender command such as CTE0192.

Usage notes

If more than the first keyword is specified, the rest are ignored and the syntax of the identified command is displayed.

If no *command* parameter is specified after **?** or **HELP** (or no parameter at all), **DB2TEXT** lists all available **DB2TEXT** command parameters.

COPYRIGHT command

Provides Net Search Extender product and copyright information.

Authorization

None

Command syntax

```
►►──┬─COPYRIGHT─┬──────────────────────────────────────────────────────────►◄
    └─LEVEL─────┘
```

Command parameters

COPYRIGHT | LEVEL
: Provides the version copyright statement, version number, and build information for the product.

Chapter 58. Net Search Extender install and uninstall command reference for UNIX

When trying to install and uninstall Net Search Extender on UNIX, it is important to have knowledge of the syntax used for these processes.

This section describes the syntax of the Net Search Extender install and uninstall commands for UNIX. It also includes the **db2nsels** command that displays the installed copies of the Net Search Extender product starting with DB2 version 9.1.

db2nse_install command

Installs a new version of Net Search Extender on UNIX operating systems.

Authorization

Root user authority

Command syntax

```
►►──db2nse_install──┬──────────────┬──┬──────────────────────┬──┬────┬──────────►◄
                    └─package-file─┘  └─path──┬────┬──┬────┬─┘  └─-v─┘
                                              └─-s─┘  └─-f─┘
```

Command parameters

package-file
: The name of the file containing the Net Search Extender product.

path
: The DB2 database path where you now want to install Net Search Extender.

-s
: Silent installation. Checks are carried out and a log file is written to the /tmp directory.

-f
: Force installation. No checks are done.

-v
: Displays the program version and exits.

Usage notes

This command installs Net Search Extender contained in the package file passed as a parameter. If you do not specify any additional parameters, the installation program checks the system for eligible copies of DB2 database systems where Net Search Extender can be installed. All DB2 database installation paths are listed but only a viable subset is selectable. You are asked by the system to select an installation path.

If you select to install silently, there is no user interaction. The outcome of the installation process is passed as a return code to the calling program which might be a command shell or a shell script. It is up to you to handle the return code correctly. A log file that documents the installation is written to the /tmp directory.

If you select to force install, there is no user interaction and there are no additional checks. The outcome of the installation process is passed as a return code to the calling program which might be a command shell or a shell script. It is up to you to handle the return code correctly. A log file that documents the installation is written to the `/tmp` directory.

Both the parameters **`-s`** and **`-f`** require that a *path* is passed. No default path exists for these parameters.

If the **`-v`** parameter is passed, only the version of the program is displayed and the program exits without any further action.

db2nse_deinstall command

Removes Net Search Extender on UNIX operating systems.

The **`db2nse_deinstall`** command is located in each install subdirectory of a DB2 copy where Net Search Extender is installed.

Authorization

Root user authority

Command syntax

```
►►──db2nse_deinstall──┬────┬──────────────────────────────────────────►◄
                      └─-v─┘
```

Command parameters

`-v` Displays the program version and exits.

Usage notes

This command removes Net Search Extender beginning with DB2 Version 9.1. The command removes the copy of Net Search Extender in the directory where it is issued. It does not remove other copies of Net Search Extender installed elsewhere.

db2nsels command

Lists all of the installed copies of Net Search Extender, starting with DB2 Version 9.7.

The **`db2nsels`** command is located in the `/usr/local/bin` directory.

Authorization

Root user authority

Command syntax

```
►►──db2nsels──┬────┬──────────────────────────────────────────────────►◄
              ├─-c─┤
              └─-v─┘
```

Command parameters

-c Displays the installed Net Search Extender versions as a simple compact list separated by colons, and exits.

-v Displays the program version, for example 9.7, and exits.

Usage notes

The command result is displayed either in a table or as a compact list where the items are separated by colons. The results include the path, the version and the fix pack number of the Net Search Extender installation.

For example, if you call **db2nsels** with no parameters, the output might be as follows:

```
db2nsels

Install path                  Level       FP
-----------------------------------------------------------------
/opt/ibm/db2/V9.7             9.7.0.0      0
/test/V9.7                    9.7.0.0      0
```

If you use the **-c** parameter, the returned output is a compact list where each information item is separated by a colon. This type of output is easy to handle using a program or a shell script. For example:

```
db2nsels -c

#PATH:VRMF:FIXPACK
/opt/ibm/db2/V9.7:9.7.0.0:0
/test/V9.7:9.7.0.0:0
```

Chapter 59. Syntax of search arguments

A search argument is the condition that you specify when searching for terms in text documents. It consists of search parameters and one or more search terms.

Examples of search arguments are given in Chapter 42, "Specifying SQL search arguments," on page 133, and in a file called `search` in the Net Search Extender `samples` directory.

The SQL scalar search functions that use search arguments are:

CONTAINS
This function uses a search argument to search for text in a particular text document. It returns the INTEGER value 1 if the document contains the text, or any relation specified in the search argument. Otherwise, it returns 0.

NUMBEROFMATCHES
This function uses a search argument to search in text documents and returns an INTEGER value indicating how many matches resulted per document.

SCORE
This function uses a search argument to search in text documents. It returns a value for each document found, indicating how well the found document is described by the search argument compared to other documents in the same index.

Note: The same syntax is used in the search arguments of the stored procedure search and the SQL table-valued function.

Search argument syntax

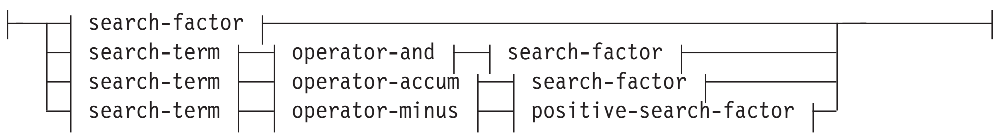
There are different syntax arguments that can be used when searching.

```
►►──┬───────────────────────┬──┬──────────────────────────┬───────────────►
    └─RESULT LIMIT──number──┘  └─EXPANSION LIMIT──number──┘

►──┬─────────────────────────────────────────┬────────────────────────────►
   └─STOP SEARCH AFTER──number──┬─DOCUMENT──┬┘
                                └─DOCUMENTS─┘

►──┬─┤ boolean-search-expression ├─┬──────────────────────────────────────►◄
   └─┤ freetext-argument ├─────────┘
```

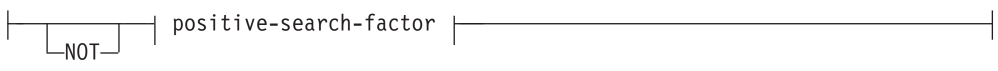
Boolean-search-expression:

```
├──┬─┤ search-term ├───────────────────────────────────────────────────┬──────────┤
   └─┤ boolean-search-expression ├──┤ operator-or ├──┤ search-term ├──┘
```

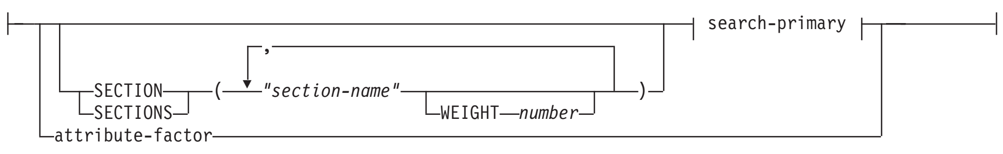
search-term:



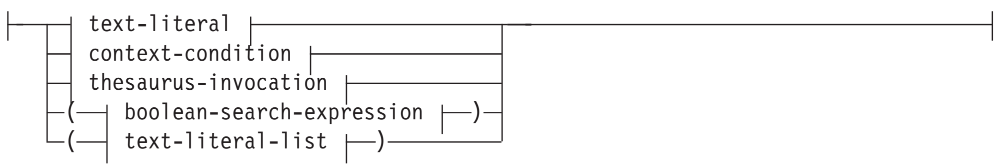
Search-factor:



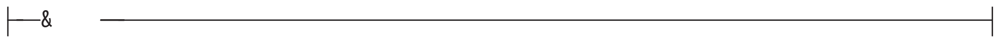
Positive-search-factor:



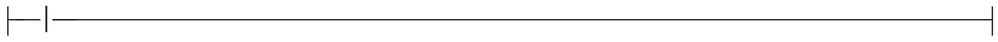
Search-primary:



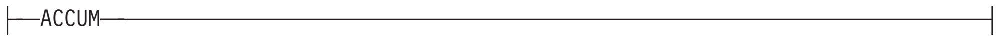
Operator-and:



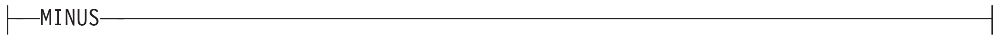
Operator-or:



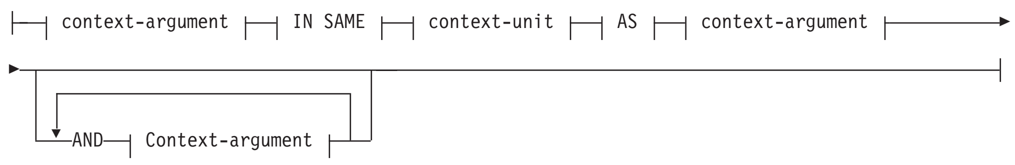
Operator-accum:



Operator-minus:



Context-condition:



Context-argument:

```
├──┬─┤ text-literal ├──────────┬──────────────────────────────────────────────────┤
   ├─(──┤ text-literal-list ├──)─┤
   └─┤ thesaurus-invocation ├───┘
```

Text-literal-list:

```
   ┌─,─────────────┐
   ▼               │
├────text-literal──┴──────────────────────────────────────────────────────────────┤
```

Context-unit:

```
├──┬─PARAGRAPH─┬──────────────────────────────────────────────────────────────────┤
   └─SENTENCE──┘
```

Text-literal:

```
├──┬──────────────────────────────┬──┬─────────────────┬─"word-or-phrase"──────────►
   ├─PRECISE FORM OF──────────────┤  └─WEIGHT──number──┘
   ├─STEMMED FORM OF──────────────┤
   └─FUZZY FORM OF──┬──────────────┬┘
                    └─match-level──┘

►──┬──────────────────────────────┬───────────────────────────────────────────────┤
   └─ESCAPE──"escape-character"───┘
```

thesaurus-invocation:

```
├──THESAURUS──"thesaurus-name"──EXPAND────────────────────────────────────────────►

►──┬─┬─SYNONYM─────────────┬─TERM OF──┤ text-literal ├─────────────────┬──────────┤
   │ ├─RELATED─────────────┤                                           │
   │ └─RELATION──(number)──┘                                           │
   └─┬─BROADER───┬─TERM OF──┤ text-literal ├──┬─────────────────────────┬┘
     └─NARROWER──┘                            └─FOR──count──┬─LEVEL───┬┘
                                                            └─LEVELS──┘
```

Attribute-factor:

```
├──ATTRIBUTE──"attribute-name"──┬─BETWEEN──valueFrom AND valueTo─┬────────────────┤
                                ├─>──valueFROM───────────────────┤
                                └─<──valueTO─────────────────────┘
```

freetext-argument:

```
├──IS ABOUT──┬────────────┬─"word-or-phrase"──┬──────────────────────────────┬────┤
             └─language───┘                   └─ESCAPE──"escape-character"───┘
```

Examples

Examples are given in Chapter 42, "Specifying SQL search arguments," on page 133.

Search parameters

This topic describes the different types of parameters when searching including a description of the parameters.

Parameters

RESULT LIMIT *number*
: A keyword specifying the maximum number of results to be returned by the full-text search.

  The RESULT LIMIT should be used together with the SCORE function to ensure that the returned results are scored and only the best matching results are processed.

EXPANSION LIMIT *number*
: A keyword specifying the maximum number of terms that a wildcard term can be expanded to for searching. For example, to determine how many times you can expand the search term 'a*'. If your index is very large and you are using many wildcard terms, you must adjust the value of this keyword if you want to obtain a larger result set. The expansion order depends on the internal organization of the text index and cannot be predetermined. If your wildcard expression is too general, and can be expanded into more search terms than specified by 'EXPANSION LIMIT', the search returns with an error, indicating that the search result has been truncated due to this limit exhaustion.

STOP SEARCH AFTER *number* **DOCUMENT | DOCUMENTS**
: A keyword specifying the search threshold. The search is stopped when the given number of documents is reached during the search, and an intermediate result is returned. A lower value will increase the search performance, but may lead to fewer results and omit documents with a potentially high rank.

  Note that there is no default value and the *number* value must be a positive integer.

boolean-search-expression
: The search-terms and search-factors can be combined using the boolean operators NOT, AND, OR, ACCUM, and MINUS according to the syntax diagrams. The operators have the following precedence order (with the strongest first): NOT> MINUS = ACCUM = AND > OR. This can be seen in the following example:

```
"Pilot" MINUS "passenger" & "vehicle" | "transport" & "public"
```

  is evaluated as:

```
(("Pilot" MINUS "passenger") & ("vehicle")) | ("transport" & "public")
```

  The operator ACCUM evaluates to true, if one of the boolean arguments evaluates to true (which is comparable to the OR operator). The rank value is computed by accumulating rank values from both operands. The ACCUM operator has the same binding (precedence) as AND. The operator MINUS evaluates to true, if the left operand evaluates to true. The rank value is computed by taking the rank value for the left operand and subtracting a penalty, if the right operand evaluates to true.

search-primary
A search-primary, consisting of a text-literal-list evaluates to true, if any of the text-literals is found in the (specified section of the) document. A search-primary consisting of a thesaurus-invocation evaluates to true, if any of the expanded text-literals is found in the (specified section of the) document.

SECTION | SECTIONS *section-name*

A keyword specifying one or more sections in a structured document that the search is to be restricted to. The section name must be specified in a model file specified at index creation time or be expressed in XPath notation.

Section names are case sensitive. Ensure that the case of the section name in the model file and query is identical.

This model describes the structure of documents that contain identifiable sections, so that the content of these sections can be individually searched. Section names cannot be masked using masking characters. The *positive-search-factor* using the SECTION clause evaluates to true, if the search primary is found in one of the specified sections.

Section names are not valid XPath expressions that are evaluated during query execution. If no model file is used, the default section names are phrased in XPath notation. The absolute path expression to the element (such as `/father/child/grandchild`) is used as the name for identifying the section. Full XPath expressions are not supported as section names.

context-argument IN SAME context-unit AS context-argument AND context-argument ...
This condition lets you search for a combination of text-literals occurring in the same paragraph or same sentence. Context arguments are always equivalent to text-literal-lists, and thesaurus expansion may be used to expand a text-literal to such a list.

The condition evaluates to true, if there is a context-unit (paragraph or sentence) in the document, which contains at least one of the text-literals of each expanded context-argument. This can be seen in the following example:

```
("a","b") IN SAME PARAGRAPH AS ("c","d")
          AND THESAURUS "t1" EXPAND SYNONYM TERM OF "e".
```

Assuming e1, e2 are synonyms of e, the following paragraphs would match:

```
".. a c e .." ,  ".. a c e1..",  "a c e2..",
".. a d e .." ,  ".. a d e1..",  "a d e2..",
".. b c e .." ,  ".. b c e1..",  "b c e2..",
".. b d e .." ,  ".. b d e1..",  "b d e2..".
```

PRECISE FORM OF
A keyword that causes the word (or each word in the phrase) following **`PRECISE FORM OF`** to be searched for exactly as typed. This form of search is case-sensitive; that is, the use of upper- and lowercase letters is significant. For example, if you search for `mice`, you do not find "Mouse".

This parameter requires that the index configuration parameter Respect case is set to yes. This configuration setting cannot be changed after the index has been built.

STEMMED FORM OF
A keyword that causes the word (or each word in the phrase) following `STEMMED FORM OF` to be reduced to its word stem before the search is carried out. This form of search is not case-sensitive. For example, if you search for `mouse`, you find "Mouse".

The way in which words are reduced to their stem form is language-dependent. Currently, only English stemming is supported and the word must follow regular inflection endings.

FUZZY FORM OF
A keyword for making a "fuzzy" search, which is a search for terms that have a similar spelling to the search term. This is particularly useful when searching in documents that were created by an Optical Character Recognition (OCR) program. Such documents often include misspelled words. For example, the word `economy` could be recognized by an OCR program as `econony`. Note that successful matches are only returned for words in a document where the first three characters match. In the previous example, `ecanomy` is not a match. Fuzzy search cannot be used if a word in the search atom contains a masking character.

match-level
An integer between 1 and 100 specifying the degree of similarity, where 100 is more similar than 1. 100 specifies an "exact match", and 60 is already considered a very "fuzzy value". The fuzzier the match level is, the longer the elapsed search time, since more documents qualify for the search. The default match level is 70.

WEIGHT *number*
Associates a text-literal with a weight value to change the default score. The allowed weight values are integers between 0 (the lowest score weighting) and 1000 (the highest); the default value is 100.

word-or-phrase
A word or phrase to be searched for. The characters that can be used within a word are language-dependent. It is also language-dependent whether words need to be separated by separator characters. For English and most other languages, each word in a phrase must be separated by a blank character.

To search for a character string that contains double quotation marks, type the double quotation marks twice. For example, to search for the text "wildcard" character, use:

```
"""wildcard"" character"
```

Note that in the example, it is only possible to search for one set of quotation marks. You cannot search for two quotation marks in a sequence. There is also a maximum length of 128 bytes for each word or phrase.

Masking characters
A word can contain the following masking characters:

_ (underscore)
Represents any single character.

% (percent)
Represents any number of arbitrary characters. If a word consists

of a single %, then it represents an optional word of any length. A word cannot be composed exclusively of masking characters, except when a single % is used to represent an optional word. If you use a masking character, you cannot use the THESAURUS keyword. Masking characters cannot be used inside thesaurus query parts. If they are used in combination, search results are unpredictable. Masking characters cannot follow a non-alphanumeric character. Masking characters cannot be used inside a fuzzy search as masking always expands into a single word.

ESCAPE *escape-character*
A character that identifies the next character as one to be searched for and not as one to be used as a masking character. For example, if an escape-character is $, then $%, $_, and $$ represent %, _, and $. Any % and _ characters not preceded by $ represent masking characters.

During search, you are only allowed to use single-byte escape characters. No double-byte characters are allowed.

THESAURUS *thesaurus-name*
A keyword used to specify the name of the thesaurus to be used to expand a text-literal. The thesaurus name is the file name (without its extension) of a thesaurus that has been compiled using the thesaurus compiler. It must be located in `<os-dependent>/sqllib/db2ext/thes`. Alternatively, the full path can be specified preceding the file name.

EXPAND *relation*
Specifies which relation is used to expand the text-literal using the thesaurus. The thesaurus has predefined relations described in the **`DB2EXTTH`** command. These are referred to using the following keywords:

- SYNONYM, a symmetrical relationship expressing equivalence.
- RELATED, a symmetrical relationship expressing association.
- BROADER, a directed hierarchical relationship that can be followed by specified depth levels.
- NARROWER, a directed hierarchical relationship that can be followed by specified depth levels.

For user-defined relations, use `RELATION(number)`, that corresponds to the relation definition in `DB2TEXTTH`.

TERM OF *text-literal*

The text-literal, to which other search terms are to be added from the thesaurus.

count **LEVELS**

A keyword used to specify the number of levels (the depth) of terms in the thesaurus that are to be used to expand the search term for the given relation. If you do not specify this keyword, a count of 1 is assumed. The value of depth must be a positive integer value.

ATTRIBUTE *attribute-name*
Searches for documents that have attributes matching the specified condition. The attribute-name refers to the name of an attribute expression in the `CREATE INDEX` command, or to an attribute definition in the document model file.

The attribute-factor is allowed for attributes of type double only. The precision of the value is guaranteed for 15 digits. Numbers that consist of 16 digits and higher are rounded. Usage of masking characters is not allowed in attribute-name, valueFrom and, valueTo. For an explanation, see the following:

BETWEEN *valueFrom* **AND** *valueTo*
A **BETWEEN** attribute factor evaluates to true if the value of the attribute is greater than (not equal to) *valueFrom* and smaller than (not equal to) *valueTo*.

>*valueFrom*
A ">" attribute factor evaluates to true if the value of the attribute is greater than (not equal to) valueFrom.

<*valueTo*
A "<" attribute factor evaluates to true if the value of the attribute is lower than (not equal to) valueTo.

If the attribute name in the **CREATE INDEX** command is specified with quotation marks, or is defined in a model file, the specified attribute name must match exactly. Whereas, if no quotation marks are specified in the `CREATE INDEX` command, the attribute name must be in uppercase.

IS ABOUT *language word-or-phrase*
An option that lets you specify a free-text search argument. Using **IS ABOUT**, you can search for any (but not necessarily all) of the words that you specify in word-or-phrase in any order in a document. The closer together the terms used in word-or-phrase are and the more terms that are included in a document, the higher the returned score for the document.

The parameter language is optional and must be set only for Thai (TH_TH) where it is required for tokenization purposes, and for Turkish (TR_TR), where it is required for proper case mapping.

Note that **IS ABOUT** is useful only if document score values are requested and the search results are ordered by score values.

Chapter 60. SQL scalar search function and the SQL table-valued function

Net Search Extender provides SQL scalar search functions and an SQL table-valued function for searching text documents stored in DB2.

This section describes the following SQL search functions.

Search function	Purpose
"CONTAINS scalar function"	Searches for text in a particular document.
"NUMBEROFMATCHES scalar function"	Searches and returns the number of matches found.
"SCORE scalar function" on page 254	Searches and returns the score value of a found text document.
"DB2EXT.TEXTSEARCH command" on page 254	The SQL table-valued function returns a table of primary keys, a number of matches, and score values.
"DB2EXT.HIGHLIGHT" on page 257	To get about why a document qualified as a search result.

CONTAINS scalar function

Searches for text in a text document indexed by Net Search Extender. It returns the INTEGER value 1 if the document contains the text, or any relation specified in the search argument. Otherwise, it returns 0.

Function syntax

```
►►—CONTAINS—(—column-name—,—search-argument—)——————————————————————►◄
```

Function parameters

CONTAINS function parameters

column name
: The name of a table column. The column must have an associated text index. You can create text indexes by using the administration command **DB2TEXT CREATE INDEX**.

search-argument
: A string of type VARCHAR containing the terms to be searched.

Note: You cannot use the CONTAINS query on a text index created on a view.

NUMBEROFMATCHES scalar function

Searches in text documents and returns an INTEGER value indicating how many matches resulted per document.

Function syntax

```
►►──NUMBEROFMATCHES──(──column-name──,──search-argument──)─────────────────────►◄
```

Function parameters

column name
: The name of a table column. The column must have an associated text index. You can create text indexes by using the administration command **DB2TEXT CREATE INDEX**.

search-argument
: A string of type VARCHAR containing the terms to be searched.

Note: You cannot use the NUMBEROFMATCHES query on a text index created on a view.

SCORE scalar function

Searches in text documents and returns a score value for each document found, indicating how well the found document is described by the search argument.

SCORE returns a DOUBLE value. As the search term appears more frequently in the document, the score of the document increases.

Function syntax

```
►►──SCORE──(──column-name──,──search-argument──)───────────────────────────────►◄
```

Function parameters

column name
: The name of a column. The column must have an associated text index. You can create text indexes by using the administration command **DB2TEXT CREATE INDEX**.

search-argument
: A string of type VARCHAR containing the terms to be searched.

Note: You cannot use the SCORE query on a text index created on a view. The values returned by score are only meaningful if they are compared to other values retrieved from the same index.

DB2EXT.TEXTSEARCH command

In addition to the stored procedure search and the SQL scalar search functions, Net Search Extender provides two SQL table-valued functions which look very similar to the stored procedure.

Purpose

Both table-valued functions are called `db2ext.textsearch`. The only difference between them is that one supports the HIGHLIGHT function and has two additional parameters, `numberOfHits` and `hitInformation`.

Both table-valued functions return the results from the user tables sorted according to the sort criteria defined in the parameter called **INITIAL SEARCH RESULT ORDER** in the **CREATE INDEX** command. If the SQL query statement containing the table-valued function has a join at the end of the statement (for example, something like `where T.primkey = S.key`), the order of the result rows depends on the join method and not on the order defined in the **CREATE INDEX** command.

- You cannot use the table-valued function on tables with a compound primary key.
- The table-valued function may be used in a partitioned database environment only if the user table is stored in a single-partition table space. You must also ensure that you connect to the correct node using the **DB2NODE** environment variable.
- db2ext.textsearch without highlight support:

```
db2ext.textSearch
                (
                query             VARCHAR(4096),
                indexSchema       VARCHAR(128),
                indexName         VARCHAR(128),
                resultFirstRow    INTEGER,
                resultNumberRows  INTEGER,
                primKeyBinding    <supported types>,// same type as primary key
                )

                return table
                (
                primKey           <supported types>,// same type as primary key
                numberOfMatches   INTEGER,
                score             DOUBLE,
                totalNbResults    INTEGER
                )
```

- db2ext.textsearch with highlight support:

```
             db2ext.textSearch
                (
                query             VARCHAR(4096),
                indexSchema       VARCHAR(128),
                indexName         VARCHAR(128),
                resultFirstRow    INTEGER,
                resultNumberRows  INTEGER,
                primKeyBinding    <supported types>,// same type as primary key
                numberOfHits      INTEGER
                )

                return table
                (
                primKey           <supported types>,// same type as primary key
                numberOfMatches   INTEGER,
                score             DOUBLE,
                totalNbResults    INTEGER
                hitInformation    BLOB(20K)
                )
```

Function parameters

The following are input parameters.

query See Chapter 59, “Syntax of search arguments,” on page 245 for additional information.

indexSchema, indexName
Identifies the index to search.

resultFirstRow
: The result list of the query is returned in parts. This parameter describes which row of the query result list is the first one to be entered into the result table of the table-valued function. The value must be >= 0.

  Note that the number 0 identifies the first row in the query result list.

resultNumberRows
: This parameter describes how many rows of the query result list are entered into the result table of the table-valued function, and where 0 means that all the results need to be returned.

  Note that this is different from the result limit query parameter that determines the maximum size of the query result list.

primaryKeyBinding
: The type of this parameter determines the type of the **primaryKey** Output parameter. If the text index has been created for a base table with a primary key of type `<type1>`, then `primaryKeyBinding` must also be of type `<type1>`.

  Additionally, the parameter determines the scope of the text search. If **primaryKeyBinding** is set to `NULL ("CAST(NULL as <type1>)"`, the scope of the search will be all the documents stored in the index. Alternatively, you can restrict the search to documents **primaryKeyBinding** is bound to.

  For example, if **primaryKeyBinding** is set to `CAST(5 as BIGINT)`, you restrict the search to the single document with the BIGINT primary key value of "5".

  Note that only single column primary keys of the following types are supported: `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `DOUBLE`, `VARCHAR FOR BIT DATA`, `DATE`, `TIME`, and `TIMESTAMP`.

numberOfhits
: This option specifies the number of terms that are highlighted using the highlighting function called `db2ext.highlight`. If 0 is specified, all of the hits are highlighted up to a maximum of 1100 hits. This process may be time-consuming.

Function parameters

The following return values are stored in a temporary table which needs to be joined to your user table if further results are requested. Note that the `NUMBEROFMATCHES`, `SCORE`, `TOTALNUMBEROFRESULTS`, and `HITINFORMATION` are only calculated if they are requested in your `select` statement.

primKey
: The primary key of the found document.

numberofmatches
: NUMBEROFMATCHES is an INTEGER value indicating how many matches resulted for each document.

score
: Score returns a DOUBLE value. As the search term increases in frequency in the document, the document score increases.

totalNumberOfResults
: The query result list denotes how many results were found. Note that each row has the same value.

Also note that when you use the STOP SEARCH AFTER, or the RESULT LIMIT together with the SCORE syntax in a query, this number is no longer reliable.

hitInformation
The hit information returned by db2ext.textsearch is necessary for highlight processing. Currently, hit information for approximately 1100 hits can be contained in this output parameter. If the number of hits exceeds this threshold, hit information for these further hits is ignored.

Note that this value is only returned if you specify numberOfHits.

Usage

With the SQL table-valued function you are able to search on views in the same way you do with the stored procedure search. Only with the SQL table-valued function no shared memory is needed, so the index does not need to have a cache that must be activated.

This function is primarily for those users who have used an SQL query within the stored procedure search. However, the restriction is that only a single column primary key on base tables is supported.

The following example shows how you can work on a multi-column primary key table:

```
select s.id from
db2ext.sample s, table (db2ext.textSearch(
        '"characteristics"',
         'DB2EXT',
         'COMMANDS',
         1,
         20,
         cast(NULL as INTEGER))) t
where s.id = t.primkey
```

In this example, you must first create a view on this table with a single unique key and then create the index on this view.

For an example of using the SQL table-valued function with the db2ext.highlight function, see “DB2EXT.HIGHLIGHT.”

DB2EXT.HIGHLIGHT

Use the db2ext.highlight function to get information that can be used to display why a particular document qualifies as a search result.

More specifically, it can be used to:
- get hits
- get hits and surrounding text
- get the document with user-defined highlight tags surrounding the hits.

Note that the db2ext.highlight function can only be used with the db2ext.textsearch table-valued function. The table-valued function searches the index providing the results for the HIGHLIGHT function to use.

Function syntax

```
►►—db2ext.highlight——————————————————————————————————————————————►

►–(—document-content—,—hit-information—,—hit-processing-information—)————————►◄
```

Function parameters

The following are input parameters:

document content CLOB(100K)
: Only UTF8 documents of TEXT or serialized XML format are supported. For highlighting natively stored XML documents, the XML data must be serialized to CLOB by using the XMLSERIALZE SQL/XML function.

  To increase the CLOB value, use the "DB2EXTTH command" on page 233.

hit information BLOB(20K)
: A string containing hit information. This is returned by the db2ext.textsearch function, if the **`numberOfHits`** parameter is specified.

hit processing information VARCHAR(1024)
: This parameter is a list of option value pairs separated by a comma ',' character with each string character surrounded by " " characters. It specifies how highlighting should be processed for the specified document. If none of the options is specified, the original document content is returned unmodified.

  TAGS = ("STRING", "STRING")
  : This option enables the user to specify the tags to be inserted before and after a hit in the document. If this option is omitted, no tags are added before and after a hit in the document.

  WINDOW_NUMBER = INTEGER
  : This option specifies how many parts (or windows) of the document should be returned by the highlight function. Each window contains one or more hits and the first hit in each window determines the part of the document returned to the user. These hits may or may not have text surrounding the hit.

    If this option is omitted, 0 is taken as the default and the entire document containing start and end tags (if specified) is returned. In this case, the `WINDOW_SIZE` option is ignored.

  WINDOW_SIZE = INTEGER
  : This option specifies the recommended size of the window in bytes. This actual size may vary, depending on the number of hits, length of hits and the start and end tag sizes. If the option is omitted, 0 is the default and only hits without surrounding text will be returned.

  WINDOW_SEPARATOR = "STRING"
  : This option specifies the tag used to separate one window from the next window. If the option is omitted, "..." is the default value.

  FORMAT = "STRING"
  : This option specifies the format of the document. Valid values are `XML` or `TEXT`. If this option is omitted, then `TEXT` is taken as the default value. Ensure that the format value is the same as that specified during indexing.

MODEL_NAME = "STRING"
This option specifies the model name related to the specified XML document. Note that if the FORMAT is TEXT, this option results in an error condition.

SECTIONS = ("section-name1", ..., "section-nameN")
For XML documents, highlighting can be restricted to relevant sections. For example, they can be defined in the model file. To specify these sections, separate the one or more section names with a comma. If this option is omitted, highlighting is performed on the whole XML document. Note that if the FORMAT is TEXT, this option is ignored.

Section specification ("section-name1",...,"section-nameN") used in DB2EXT.HIGHLIGHT must be the same as is for the DB2EXT.TEXTSEARCH function.

Function parameters

The following are return parameters.

CLOB(200K)
The HIGHLIGHT function returns a CLOB value containing the document parts modified by the HIGHLIGHT function.

Usage

The following example shows how you can use the HIGHLIGHT function:

```
select p.id,
       p.title,
       db2ext.highlight(p.content,
       t.hitinformation,
       'TAGS = ("<bf>", "</bf>"),
       WINDOW_NUMBER = 5,
       WINDOW_SIZE = 200,
       WINDOW_SEPARATOR = "...",
       FORMAT = "XML",
       SECTIONS = ("section1-name", "section2-name")')

FROM patent p, table (db2ext.textsearch(
       '"relational database systems"',
       'DB2EXT',
       'TI_FOR_CONTENT',
       0,
       20,
       CAST(NULL as BIGINT),
       15)) t

WHERE p.id = t.primkey
```

Using documents larger than 100 KB will cause the SQL query to terminate and produce an SQL error (SQL1476N and sql error -433). To avoid this, use the db2exthl command to increase the permitted document content size.

Note: Special characters, such as "newline" will be returned as is.

Highlighting natively stored XML documents requires a serialization of these XML documents to CLOB before they can be passed to the HIGHLIGHT table-valued function. The following example shows you how you can use the HIGHLIGHT function on natively stored XML documents using the XMLSERIALZE SQL/XML function. The patent content of the following sample is stored as native XML. Notice that FORMAT="XML" is also specified:

```
select p.id,
       p.title,
       db2ext.highlight(XMLSERIALIZE(p.content AS CLOB(100K)),
       t.hitinformation,
       'TAGS = ("<bf>","</bf>"),
       FORMAT = "XML",
       SECTIONS = ("section1-name", "section2-name")')

FROM patent p, table (db2ext.textsearch(
        '"xml database systems"',
        'DB2EXT', 'TI_FOR_XML',
        0,
        20,
        CAST(NULL as BIGINT),
        15)) t

WHERE p.id = t.primkey
```

Restrictions

- Only XML and flat text documents are supported.
- Only UTF8 databases are supported. For binary documents, you need to ensure that the documents are encode in UTF8.
- Thai documents are not supported.
- If there is a mismatch between the document format used during indexing and query time the HIGHLIGHT function will return unpredictable results. This is particularly true for cases where a transformation function is used for obtaining the text during indexing. Any change to the transformation function between the time of the indexing, and the time of submitting the search request that affects the position of text tokens in the output will render the results of the highlight function invalid.
- Only hits found in the text parts of a document will be highlighted.
- The highlight table-valued function can only be used with the db2ext.textsearch function.
- String values cannot contain the " character.

Chapter 61. Stored procedure search function

Net Search Extender provides a stored procedure search for returning predefined result tables. The result table is specified in the cache table section during create index.

Use the stored procedure search when you need to return a small number of results in a specific order.

An example would be a Web application where the first 20 rows with the best score are returned, but the rest of the results can also be returned in increments of 20 rows.

Note: The stored procedure function may be used in a partitioned database environment only if the user table is stored in a single-partition table space.

You must also ensure that you connect to the correct partition using the **`DB2NODE`** environment variable.

DB2EXT.TEXTSEARCH for stored procedure search

The columns in the result set returned by the stored procedure are given by the **`CACHE TABLE`** option of the **`DB2TEXT CREATE INDEX`** command. If `scoringFlag=1`, then a column of type double is added.

Function syntax

```
db2ext.TextSearch(

        IN      query                   VARCHAR(4096),
        IN      indexSchema             VARCHAR(128),
        IN      indexName               VARCHAR(128),
        IN      resultFirstRow          INTEGER,
        IN      resultNumberRows        INTEGER,
        IN      scoringFlag             INTEGER,
        IN      searchTermCountsFlag    INTEGER,
        OUT     searchTermCounts        VARCHAR(4096),
        OUT     totalNumberOfResults    INTEGER )
```

Function parameters

The following are input parameters.

Query See Chapter 59, “Syntax of search arguments,” on page 245 for further information.

`indexSchema`, `indexName`
To identify the index to search.

`resultFirstrow`
The query result list is returned in parts. The parameter describes which row of the query result list is the first one to be put into the result set of the stored procedure. The first row in the query result list is identified by the number 0.

`resultNumberRows`
This parameter describes how many rows of the query result list are put into the result set of the stored procedure.

This is not to be confused with the "result limit" expression in the query, which determines the maximum size of the query result list.

The value should be >= 0. Where 0 means that all the results need to be returned.

Note: If a larger result set is requested, ensure that a temporary user table space is available. If there is none available, then create a table space. The following example creates a table space on a UNIX operating system:

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

`scoringFlag`
: 0 means that there is no scoring and 1 means that there is scoring. If scoring is requested, an additional column with the score values is returned with the highest value first.

`searchTermCountsFlag`
: This controls the searchTermCounts processing. If **`searchTermCountsFlag`** is `0`, searchTermCounts is not calculated.

Function parameters

The following are output parameters.

`searchTermCounts`
: The number of occurrences of each search term query in the index. These counts are returned as a blank separated list in the order of search terms in the query.

: See the **`searchTermCountsFlag`** for information.

`totalNumberOfResults`
: The total number of results found in the query result list.

: Also note that when you use the `STOP SEARCH AFTER`, or the `RESULT LIMIT` together with the `scoringFlag` syntax in a query, this number is no longer reliable.

Usage

The columns in the result set returned by the stored procedure are given by the **`CACHE TABLE`** option of the **`DB2TEXT CREATE INDEX`** command. If `scoringFlag=1`, then a column of type double is added. This column contains the `SCORE` value.

Use the following options to increase the performance of a second query with the same string as the first query. Note that this must be in a different cursor window with no `totalNumberOfResults` required:

- If you do not require scoring, add the following syntax: `STOP SEARCH AFTER` *x* `DOCUMENTS`, where *x* is the `resultFirstRow` + `resultNumberRows`.
- If you require scoring, add the following syntax: `STOP SEARCH AFTER` *y* `DOCUMENTS`, where *y* is equal to the `totalNumberOfResults` in the first query.

To ensure that you connect to the correct node for searching, it may be necessary to set the **`DB2NODE`** environment variable.

For UNIX, use the following command:

```
export DB2NODE=<no>
```

Note that it is important that all physical nodes have a synchronized time.

For Windows, use:

```
set DB2NODE= <no>
```

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

Chapter 62. Windows system errors

The following is a list of Windows system errors:

System errors

1	Incorrect function.
2	The system cannot find the file specified.
3	The system cannot find the path specified.
4	The system cannot open the file.
5	Access is denied.
6	The handle is invalid.
8	Not enough storage is available to process this command.
14	Not enough storage is available to complete this operation.
15	The system cannot find the drive specified.
29	The system cannot write to the specified device.
30	The system cannot read from the specified device.
32	The process cannot access the file because it is being used by another process.
36	Too many files opened for sharing.
38	Reached the end of the file.
39	The disk is full.
80	The file exists.
82	The directory or file cannot be created.
100	Cannot create another system semaphore.
101	The exclusive semaphore is owned by another process.
102	The semaphore is set and cannot be closed.
103	The semaphore cannot be set again.
104	Cannot request exclusive semaphores at interrupt time.
105	The previous ownership of this semaphore has ended.
110	The system cannot open the device or file specified.
111	The file name is too long.
112	There is not enough space on the disk.
121	The semaphore timeout period has expired.
126	The specified module could not be found.
127	The specified procedure could not be found.
147	Not enough resources are available to process this command.
155	Cannot create another thread.

161	The specified path is invalid.
164	No more threads can be created in the system.
170	The requested resource is in use.
183	Cannot create a file when that file already exists.
187	The specified system semaphore name was not found.
206	The filename or extension is too long.
267	The directory name is invalid.
288	Attempt to release mutex not owned by caller.
298	Too many posts were made to a semaphore.
998	Invalid access to memory location.
1051	A stop control has been sent to a service that other running services are dependent on.
1052	The requested control is not valid for this service.
1053	The service did not respond to the start or control request in a timely fashion.
1054	A thread could not be created for the service.
1055	The service database is locked.
1056	An instance of the service is already running.
1057	The account name is invalid or does not exist, or the password is invalid for the account name specified.
1058	The service cannot be started, either because it is disabled or because it has no enabled devices associated with it.
1059	Circular service dependency was specified.
1060	The specified service does not exist as an installed service.
1061	The service cannot accept control messages at this time.
1062	The service has not been started.
1063	The service process could not connect to the service controller.
1064	An exception occurred in the service when handling the control request.
1066	The service has returned a service-specific error code.
1067	The process terminated unexpectedly.
1068	The dependency service or group failed to start.
1069	The service did not start due to a logon failure.
1070	After starting, the service hung in a start-pending state.
1071	The specified service database lock is invalid.
1072	The specified service has been marked for deletion.
1073	The specified service already exists.
1078	The name is already in use as either a service name or a service display name.

1079	The account specified for this service is different from the account specified for other services running in the same process.
1082	No recovery program has been configured for this service.
1154	One of the library files needed to run this application is damaged.
1219	The credentials supplied conflict with an existing set of credentials.
1242	The service is already registered.
1243	The specified service does not exist.
1244	The operation being requested was not performed because the user has not been authenticated.
1245	The operation being requested was not performed because the user has not logged on to the network. The specified service does not exist.
1392	The file or directory is corrupted and unreadable.
1455	The paging file is too small for this operation to complete.
1793	The user's account has expired.

Chapter 63. Net Search Extender information catalogs

Net Search Extender stores important information about defaults, configurations, text indexes, and formats in catalog tables. To view this information, you can query some views on tables.

The following views and tables reflect the current configuration of your system:

- Database level information views:
  - `db2ext.dbdefaults`
- Index level information views:
  - `db2ext.textindexes`
  - `db2ext.textindexformats`
  - `db2ext.indexconfiguration`
- Table views for a text index:
  - Event view
  - Log table view
  - Staging table view

Views for database-level information

The view db2ext.dbdefaults displays all the default values for the Net Search Extender database.

The defaults on the database level can be changed and are available as attribute-value pairs in this view:

```
db2ext.dbdefaults
```

```
db2 select DEFAULTNAME, DEFAULTVALUE from DB2EXT.DBDEFAULTS
```

Table 3. db2ext.dbdefaults view

Attribute	**Default Value**	**Notes**
CCSID	CCSID of database	Default CCSID for documents. This is applied if no CCSID is specified in the CREATE INDEX command.
FORMAT	TEXT	Document default format. This is applied if no format is specified in the CREATE INDEX command.

Table 3. db2ext.dbdefaults view (continued)

Attribute	**Default Value**	**Notes**
INDEXDIRECTORY	See the path name under Notes	Directory for full-text index files. This is applied if no index directory is specified in the CREATE INDEX command. For Linux and UNIX operating systems: `INSTHOME/sqllib/db2ext/indexes` where *INSTHOME* is the instance owner home directory. For Windows operating systems: `INSTPROFDIR\instance-name\db2ext\indexes` where *INSTPROFDIR* is the instance profile directory
LANGUAGE	EN_US	The document language.
MODELCCSID	CCSID of database	CCSID of document model files.
UPDATECOMMITCOUNT	0	The number of changes processed in one transaction during an update.
CLEARCOMMITCOUNT	0	The number of changes processed in one transaction during a **CLEAR INDEX** command.
UPDATEFREQUENCY	NONE	When to check for updates in new indexes.
UPDATEMINIMUM	1	The minimum number of changes before update is executed.
WORKDIRECTORY	See the path name under Notes	Directory for index temporary files. For Linux and UNIX operating systems: `INSTHOME/sqllib/db2ext/indexes` where *INSTHOME* is the instance owner home directory. For Windows operating systems: `INSTPROFDIR\instance-name\db2ext\indexes` where *INSTPROFDIR* is the instance profile directory
CACHEDIRECTORY	See the path name under Notes	Default directory for **PERSISTENT CACHE** option of the **CREATE INDEX** command. For Linux and UNIX operating systems: `INSTHOME/sqllib/db2ext/indexes` where *INSTHOME* is the instance owner home directory. For Windows operating systems: `INSTPROFDIR\instance-name\db2ext\indexes` where *INSTPROFDIR* is the instance profile directory
PCTFREE	50	The percentage of the cache left free for future inserts.

Table 3. db2ext.dbdefaults view (continued)

Attribute	Default Value	Notes
USERPERSISTENTCACHE	1	Use the persistent cache.
AUTOMATICREORG	1	The **REORGANIZE** option in the **CREATE INDEX** command. This means automatic reorganization.
TREATNUMBERSASWORDS	0	Do not interpret sequences of characters and numbers as separate words, even if they are adjacent characters. For example, the 0 default means that tea42at5 is considered as one word.
INDEXSTOPWORDS	1	Index all text including stop words.
VERSION		NSE V9.7 Current version number of Net Search Extender.
UPDATEDELAY	0	Specifies the duration in seconds for incremental update without capture tables. Only entries older than this duration will be taken from the log table. This is to avoid lost updates. For example, document changes that are not reflected in the index in transaction scenarios where user transactions interfere with update commands. Therefore, the UpdateDelay parameter should be set to the maximum duration of a user write transaction on the table the index was created on.
AUXLOGNORM	OFF	Do not enable the extended text-maintained staging infrastructure for non-partitioned tables by default. The staging infrastructure can be enabled for a text index with explicit index configuration AUXLOG ON.
AUXLOGPART	ON	By default, enable the extended text-maintained staging infrastructure for range-partitioned tables. The staging infrastructure can be disabled for a text index with explicit index configuration AUXLOG OFF.
LOCKSCHEDULERFILE	0	Set to '1' to block concurrent write-access to the scheduler file `ctedem.dat`. In some scenarios with a high degree of parallelism to create, drop, or alter update frequencies, update processes may otherwise be started unnecessarily.

Note: On Windows operating systems the default index directory has changed. In the DB2 Net Search Extender Version 9.5, the value was *<DB2-installation-path>*`\db2ext\indexes`. See Directory structure for your installed DB2 database product (Windows) for the DB2 installation path

Views for index-level information

Index-level information can be queried using table views.

You can query information at an index-level using the following Net Search Extender views:

- `db2ext.textindexes`
- `db2ext.textindexformats`
- `db2ext.indexconfiguration`
- `<index eventview name schema>.<index eventview name>`

For backward compatibility reasons, DB2 Text Information Extender views `db2ext.textcolumns`, `db2ext.formats`, and `db2ext.models` are still supported, but deprecated.

Note that in the `db2ext.textcolumns` view the `OPERATION`, `OPERATIONBEGIN`, and `OPERATIONEND` columns are no longer supported.

db2ext.textindexes view

Each database enabled for Net Search Extender contains a `db2ext.textindexes` view. This contains information about settings, statistics, and defaults for the created text indexes in this database.

When you create a text index, new entries are created in `db2ext.textindexes`. When you drop the text indexes, these entries are deleted.

You can query the view to obtain information about the indexes. This is an example using the index schema:

```
db2 "select COLNAME from DB2EXT.TEXTINDEXES where INDSCHEMA='myschema'
     and INDNAME='myindex'"
```

Note, however, that you cannot modify the view using normal SQL data manipulation commands, or explicitly create or drop the catalog view. Additional contents of the view are found in the following table.

Also note that the replication parameters are not included in this view.

Table 4. db2ext.textindexes view

Attribute	Type	Comments
INDSCHEMA	VARCHAR(128)	Schema name of the text index.
INDNAME	VARCHAR(128)	Name of the text index.
TABSCHEMA	VARCHAR(128)	The table name of the schema for base tables, nicknames, and views.
TABNAME	VARCHAR(128)	Alias name the index was created on.
COLNAME	VARCHAR(128)	Column the index was created on.
CCSID	INTEGER	Document CCSID for this index.
LANGUAGE	VARCHAR(5)	Document language for this index.
FUNCTIONSCHEMA	VARCHAR(128)	Schema of the column mapping function.
FUNCTIONNAME	VARCHAR(18)	Name of the column mapping function.
INDEXDIRECTORY	VARCHAR(256)	Directory for full-text index files.
WORKDIRECTORY	VARCHAR(256)	Directory for index temporary files.
CACHEDIRECTORY	VARCHAR(256)	Directory for persistent cache (if persistentcache=1).

Table 4. db2ext.textindexes view (continued)

Attribute	**Type**	**Comments**
UPDATEFREQUENCY	VARCHAR(300)	Trigger criterion for applying automatic updates to this index.
UPDATEMINIMUM	INTEGER	Minimum number of documents that must be changed before an update is performed.
EVENTVIEWSCHEMA	VARCHAR(128)	Schema of the event view created for this index.
EVENTVIEWNAME	VARCHAR(128)	Name of the event view created for this index.
LOGVIEWSCHEMA	VARCHAR(128)	Schema of the log view created for an index.
LOGVIEWNAME	VARCHAR(128)	Name of the log view created for an index (important for incremental update on views).
COMMITCOUNT	INTEGER	Default for commitcount update.
NUMBER_DOCS	INTEGER	Total number of documents currently in the index. Note that during an index update, this value is only updated if the `commitcount` is set.
REORG_SUGGESTED	INTEGER	Indicates if performance can be improved by running UPDATE INDEX REORGANIZE. This parameter is only true (1) if at least one of the nodes has an index reorganization suggested.
REORGAUTOMATIC	INTEGER	1, if the index gets automatically reorganized during the update operation.
RECREATEONUPDATE	INTEGER	1, if the index gets automatically reorganized during the update operation.
CREATIONTIME	TIMESTAMP	Time of index creation.
UPDATETIME	TIMESTAMP	Time of last update. If UPDATE TIME is equal to CREATION TIME, then no update has been processed.
PERSISTENTCACHE	INTEGER	1, if persistent cache is used.
MAXIMUMCACHESIZE	INTEGER	Maximum size of cache.
PCTFREE	INTEGER	Percentage of cache left free for future inserts.
CACHETABLE	VARCHAR(32000)	Column expression list for the CACHE TABLE.
RESULTORDER	VARCHAR(32000)	SQL-order-by for INITIAL RESULT ORDER.
ATTRIBUTES	VARCHAR(32000)	Column expression list for ATTRIBUTES.
VIEWKEYCOLUMNS	VARCHAR(32000)	Key columns for index on view.
AUXSTAGINGSCHEMA	VARCHAR(16)	Schema for text-maintained staging tables; set to SYSIBMTS

Table 4. db2ext.textindexes view (continued)

Attribute	Type	Comments
AUXSTAGINGNAME	VARCHAR(48)	System-generated name of the text-maintained staging table (only when configured).

db2ext.indexconfiguration view

Index configuration parameters are available in the db2ext.indexconfiguration view.

The view is available through normal SQL query facilities. This is an example using the index name:

```
db2 "select VALUE from DB2EXT.INDEXCONFIGURATION where INDSCHEMA='myschema'
     and INDNAME='myindex' and PARAMETER ='INDEXSTOPWORDS'"
```

Additional contents of the view are found in the following tables.

Table 5. db2ext.indexconfiguration view

Attribute	Type	Comments
INDSCHEMA	VARCHAR(128)	Schema name of the index.
INDNAME	VARCHAR(128)	Name of the index.
PARAMETER	VARCHAR(30)	Type of parameter.
VALUE	VARCHAR(512	Value of the parameter.

For the `PARAMETER` and `VALUE` attributes, there are several values available.

Table 6. db2ext.indexconfiguration view

Attribute and values	Attribute and values
PARAMETER	VALUE
- TREATNUMBERASWORDS	- 0 or 1
- INDEXSTOPWORDS	- 0 or 1
- UPDATEDELAY	- seconds >= 0
AUXLOGPART	- ON or OFF
AUXLOGNORM	- ON or OFF

For further information, see the **`CONFIGURATION`** option of the **`CREATE INDEX`** command.

db2ext.textindexformats view

Format and model information for indexes is available in the db2ext.textindexformats view.

This is an example using the index name:

```
db2 "select FORMAT from DB2EXT.TEXTINDEXFORMATS where INDSCHEMA='myschema'
     and INDNAME='myindex'"
```

Additional contents of the view are found in the following table.

Table 7. db2ext.textindexformats view

Attribute	Type	Notes
INDSCHEMA	VARCHAR(128)	Schema name for the index (used as prefix for indexname and schemaname in the log table).
INDNAME	VARCHAR(128)	Index name specified in CREATE INDEX command.
FORMAT	VARCHAR(30)	The model is bound to this format.
MODELNAME	VARCHAR(30)	The name of a document model.
MODELFILE	VARCHAR(256)	File containing the model definition.
MODELCCSID	INTEGER	CCSID of MODELFILE.
DEFAULT	INTEGER	Currently 1, as multiple formats in an index are not currently supported.

Table views for a text index

You can query information at an index level using the event view and log table view.

Event view

This view allows you to get information about indexing status and error events, and when problems occur during indexing, for example, a document not being found. These index update events are then written to the index's event table.

This view allows you to get information about indexing status and error events, and when problems occur during indexing, for example, a document not being found. These index update events are then written to the index's event table.

The schema and name are stored in the db2ext.textindexes view. To get the name of the event view, use the following example:

```
db2 "select EVENTVIEWSCHEMA, EVENTVIEWNAME from DB2EXT.TEXTINDEXES
       where INDSCHEMA = 'myschema' and INDNAME = 'myindex'
```

The event view for an index consists of the following columns.

Table 8. The event view

Attribute	Type	Comments
OPERATION	INTEGER	The operation on the user table is be reflected in full-text index (insert = 0/ update = 1/ delete = 2). When using a replication capture table, update operations are split into a delete and an insert operation. In this case, an insert operation in the event table can be either from an insert, or an update operation on the source table the index was created on.
TIME	TIMESTAMP	Timestamp of event entry creation.
REASON	INTEGER	Reason code. For a list of reason codes, see Chapter 64, “Text Search Engine reason codes,” on page 279.

Table 8. The event view (continued)

Attribute	Type	Comments
SEVERITY	INTEGER	The severity of the table entry. For example, 1 is for information purposes, 4 indicates a warning, and 8 means a table entry error.
MESSAGE	VARCHAR(1024)	Additional text information.
KEY1, ... KEY14	Depends on the user table	First primary key column of user table to the last primary key column (a maximum of 14).
PARTITION	INTEGER	The database partition number on which this error occurs. In a non-partitioned database environment, this is 0.

The events can be cleared by the **`DB2TEXT CLEAR EVENTS`** command.

Note: Informational events, such as starting, committing, and finishing update processing are also available in this view.

In this case, `Key1, ... Key14` and `OPERATION` all have NULL values.

In the case of indexes on views, the PK01, ..., PK14 columns relate to the columns specified in the KEY COLUMNS clause of the **`CREATE INDEX`** command.

Log tables, views, and nicknames

The purpose of the log table is to store the change operations on the user table or view that require synchronization with the external full-text index.

For indexes created on regular tables or nickname tables, there are triggers created on the user table to feed change information into the log table. However, if replication capture tables are used, no log table is created and the replication capture table is used instead.

For log tables, the update command reads the entries and deletes them after successful synchronization.

However, in the case of indexes on views, triggers cannot fill the log table. As you can update the view, the user is responsible for this task.

Table 9. The log table view

Attribute	Type	Comments
OPERATION	INTEGER	The type of change on the user table that requires index synchronization: (0 = insert, 1 = update, 2 = delete).
TIME	TIMESTAMP	The timestamp for the creation of a row in this table.
PK01 ... PKnm	Same as user table	In case of errors, the column where the problem occurred. They are a copy of the primary key columns of the user table or the equivalent key columns in case of an index on a view.

The user who creates the table is able to select, update, insert, and delete this view.

Table 10. The text-maintained staging table

Attribute	Type	Notes
PK 1..n	Same as user table	A copy of the primary key definition(s) of the user table.
Globaltransid	CHAR(8)	Internal transaction id
Globaltranstime	CHAR(13)	Timestamp
Operationtype	Integer	Insert 1
		Delete -1

The content of the table is maintained by the text search, however an administrator may delete entries in the table.

If you specify a replication capture table in the create index command, no log table is created and the replication capture table is used instead. The replication capture table must contain the following columns:

Table 11. The replication capture table

Attribute	Type	Notes
IBMSNAP_OPERATION	INTEGER	The type of change on the CD or CCD table that requires index synchronization: (I = insert, U= update, D= delete). When using a replication capture table, update operations are split into a delete and an insert operation. In this case, an insert operation in the event table can be either from an insert, or an update operation on the source table the index was created on.
IBMSNAP_COMMITSEQ	CHAR	Maps to the corresponding column of the CD or CCD table.
IBMSNAP_INTENTSEQ	CHAR	Maps to the corresponding column of the CD or CCD table.
PK01 ... PKnm	Same as user table	In case of errors, the column where the problem occurred. They are the primary key columns of the user table.

The user who defines the table is able to select, update, insert, and delete with the grant option.

Chapter 64. Text Search Engine reason codes

Text Search Engine can produce error codes and reasons for the error.

0	Operation performed successfully - no error occurred.
1	An invalid handle was passed to a function.
2	Function could not allocate enough memory.
3	Function could not perform due to access limitations or security restrictions.
4	The operation is not supported for this version of the Text Search Engine run time.
5	The operation is currently not enabled.
6	The application violated the Text Search Engine protocol by calling Text Search Engine functions in illegal order.
7	An unexpected error occurred. Please report this to your service representative.
8	An invalid language was specified.
9	The specified language is valid but not supported by the Text Search Engine run time.
10	An invalid CCSID was specified.
11	The specified CCSID is valid but not supported by the Text Search Engine run time.
12	An invalid document ID was specified.
13	The specified document format is valid but not supported by the Text Search Engine run time.
14	An invalid document format was specified.
15	The operation could not succeed due to access limitation during file input/output.
16	The operation could not succeed due to read errors during file input/output.
17	The operation could not succeed due to read errors during file input.
18	The operation could not succeed due to write errors during file output.
19	The operation could not succeed due to seek errors during file input/output.
20	The operation could not succeed due to tell errors during file input/output.
21	The operation could not succeed due to close errors during file input/output.
22	The operation could not succeed due to errors during rename operations.
23	The operation could not succeed due to errors during remove operations.
24	The operation could not succeed due to errors during mkdir operations.

25	One or more function arguments did have an invalid value (for example, an unexpected null pointer or an invalid enumeration type value).
26	The specified directory does not exist.
27	An unexpected Text Search Engine error occurred. Please examine the Text Search Engine error code in the error info object for further details.
28	An unexpected COS error occurred. Please report this error.
29	An attempt was made to update an empty document.
30	The specified argument is not supported for this operation.
31	The date attribute parser found an invalid value when trying to parse a date attribute.
32	The number attribute parser found an invalid value when trying to parse a number attribute.
33	Attribute name invalid, probably too long.
35	Reserve number for future use.
36	The input document contains an attribute (DATE, NUMBER, or STRING) that exceeds the length limit for attributes. The attribute text has been truncated to that limit.
38	The warning threshold as set by the user has been exceeded. As a consequence, this error has been generated.
39	The input document could not be indexed. It contains too many nested fields.
40	The limit of different attributes for one of the attribute types has been exceeded for this index.
46	The iterator is not (or no longer) valid, because its list is empty or deleted.
47	The function is not supported for the passed kind of handle. This error occurs, for example, when trying to use itlQueryResultEntryObtainData on a list iterator that does not represent a query result iterator.
48	This warning is issued if a stop word file cannot be found for the specified language and resource path.
49	This warning is issued if a stop word file does not contain any stop words.
50	This warning is issued if a stop word file does contain invalid data.
100	The index could not be opened because it does not exist with the specified name and directory.
101	The specified index name is not a valid index name.
102	The specified index directory is not a valid directory name.
103	The operation cannot be performed because the Text Search Engine detected a corruption of the index structure or index file sets.
104	The specified index cannot be created because it already exists with the given name and directory.
109	Before any other operation can be performed on this index, a rollback operation must be performed.
110	The index configuration file does not contain the mandatory section as specified in the error context.

111	The index configuration file does not contain the mandatory option as specified in the error context.
112	The index configuration file contains invalid data in the option as specified in the error context.
113	The index configuration file does not match the Text Search Engine version.
200	The specified document model name is not a valid model name.
201	The specified document model field name is not a valid field name.
202	The specified document model is not known.
203	The specified document model already exists and cannot be redefined.
204	Too many or too large document models have been added to the index.
205	The document model contains too many elements.
206	The document model element contains a parameter (XML attribute) that is not allowed for this type of element.
207	The document model element contains a parameter value that is not allowed for this type of parameter (XML attribute).
208	The document model element does not contain a required parameter (XML attribute), like "name".
209	The document model does not seem to be XML, or starts with an unexpected XML element.
210	The given XPath (locator value) contains an unexpected token.
211	The given XPath (locator value) contains an unexpected axis specifier (name followed by two colons).
212	The given XPath (locator value) contains an unexpected node test.
213	The document model directory file (extension .mdx) is corrupted.
214	The document model index file (extension .mox) is corrupted.
215	The document contains an XML element which is mapped to a document attribute and which contains another document attribute. The inner attribute is ignored.
216	The given parameter value is too long as a GPP or HTML tag.
217	The document model contains a duplicate field definition.
218	The document model contains a duplicate attribute definition.
300	The operation cannot be performed because the Text Search Engine detected a corruption in the index files used for document name mapping.
301	The operation cannot be performed because the Text Search Engine detected an invalid document number.
302	The operation cannot be performed because the Text Search Engine detected an invalid document identifer.
303	The operation cannot be performed because the Text Search Engine found no index entry for the document identifier.
304	The operation cannot be performed because the Text Search Engine found no index entry for the document number.

305	The operation cannot be performed because the Text Search Engine detected an overflow in used document numbers.
306	The document identifier that the application tried to index has appeared already in the list of documents. The Text Search Engine does not support duplicate document identifiers appearing in one indexing sequence, that is, before the update has been committed.
340	The term strength is not valid.
341	The relation number is not valid, must be in.
342	The relation type is invalid, use one of the defines described in API.
343	The phrase (term) is too long.
344	Unexpected end of file encountered while reading.
345	Version conflict detected when reading index/thesaurus files.
346	Overflow in thesaurus buffers.
347	Invalid name, probably too long a name, for file or directory.
348	Lookup did not find term (phrase) in dictionary or entry in definition file does not contain mandatory term.
349	Definition file is empty.
350	Thesaurus dictionary or definition file as specified via input parameter does not exist.
351	Syntax errors in definition file.
352	The Relationship was specified incorrectly.
352	The Relationship number was out of range.
360	An invalid single character masking was used.
361	An invalid multiple character masking was used.
362	Operator arity is smaller than number of operands given in query.
363	Operator value out of range defined by ItlEnOperator enumeration.
364	Value for rank formula out of enumeration range.
365	Number identifying proximity segment is out of range.
366	Query is under construction and cannot be redefined or reset.
367	Scope given as previous search result denotes empty result.
368	Invalid call requesting to add field names before setting the first one.
369	Invalid search flag requesting an invalid comparison with index content is ignored. If, for example, a case-sensitive comparison was requested for an index that was build in a case insensitive manner, this reason code is shown in the error information.
370	Masking of strings is not supported for Thai or DBCS languages.
371	No valid query input. For example, the search terms is available.
372	Invalid comparison operations requested.
373	Invalid comparison operations requested.
374	Search index handle was requested for an empty index.

375	The combination of operator and requested operator mode is not supported.
380	Search result is incomplete, search was discontinued due to threshold.
381	Index lookup revealed that query contained stopwords.
401	The operation cannot be performed because the Text Search Engine detected a corruption in the index files used for field/attribute name mapping.
402	The operation cannot be performed because the Text Search Engine detected an invalid field or attribute name.
403	The operation cannot be performed because the given field or attribute name is unknown.
404	The limit of different attributes for one of the attribute types or of different fields has been exceeded for this index.
500	The document/data contains an invalid character sequence (in a UTF8, UTF16, or DBCS source).
501	The code page converter was in error.
502	The document/data contains an incomplete character sequence (in a UTF8, UTF16, or DBCS source).
503	The code page converter has an invalid descriptor.
600	The XML document contains an asynchronous entity. For example, an unquoted XML attribute value.
602	Invalid character reference, (for example, or).
603	Invalid binary entity reference.
604	XML Parser Expat could not be created.
605	An attribute name in tag must be unique.
607	XML Parser found an invalid external entity reference.
608	Documents includes an incorrect token, like missing a < or >.
609	XML documents must have an enclosing tag, and after this enclosing end tag no text is allowed.
610	A processing instruction is not allowed at its position. For example, the first processing instruction is not the prolog <?xml .. ?>.
611	An element is a sequence of start tag, content, and end tag. This error occurred, for example, from the sequence "<s> text /s>", because the end tag is not correct.
612	Memory allocation failed in XML parser.
614	Invalid parameter entity reference.
615	A non-complete character, maybe only the first byte of a 2-byte UTF8 character.
616	Recursive entity reference.
617	XML Syntax error; for example text outside the enclosing start and end tag.
618	Every start tag needs a matching end tag.
619	Unclosed cdata section.

620	Unclosed token; for example text after the last token in a document.
621	There is an entity in the document that could not be resolved.
622	Unexpected error.
631	Could not parse field or attribute information in meta-tag. Tag must have the format <meta name="abc" content="xyz">; maybe attributes name or content of the meta-tag not correct.
632	The entity could not be transformed to a character.
650	Different field definitions begin with the same start tag.
651	One start tag contains another, so the tags are ambiguous.
652	If a field and an attribute use the same start tag, then they must use the same end tag or both no end tag.
653	A field not yet closed if the document ends.
654	No document model is specified for the structured format. The document will be parsed as plain text document without field or attribute informations.
670	The operation could not be performed, because it requires the "Outside In" (TM) libraries, which could not be found.
671	The operation could not be performed, because a required procedure from the "Outside In" (TM) libraries could not be loaded. Probably the libraries are outdated or corrupted.
672	An error occurred while the document was processed with "Outside In".

Part 14. Troubleshooting

Chapter 65. Tracing faults

If you need to report an error to an IBM representative, you may be asked to switch on tracing so that information can be written to a file that can be used for locating the error.

About this task

As system performance is affected when tracing is switched on, only use the trace facility when directed by an IBM Support Center representative, or by your technical support representative.

Procedure

- To turn tracing on, use the DB2 facility:
  ```
  db2trc on
  ```
- To receive information specific to Net Search Extender, a mask with component 96 can be used:
  ```
  db2trc on -m *.*.96.*.*
  ```

What to do next

In the case of severe errors, it may also help to look in **db2diag** log file.

Chapter 66. Dropping DB2 objects without using the correct Net Search Extender commands

Dropping a table

Before you drop a table, you must drop all text indexes associated with that table. Text indexes that are not removed before dropping a table still exist in the database memory.

About this task

Before you drop a table with one or more text indexes, you must issue the following command for each text index:

```
db2text drop index index_name for text
```

If you accidently drop a table before you drop the indexes, parts of the indexes will still exist, for example, the administration tables and text index files.

To remove these files, drop the indexes using the **db2text drop index** command, even though the table no longer exists.

Dropping a database

You must remove all text indexes associated with a database before dropping that database. Indexes that are not removed before dropping a database still exist in the database memory.

Procedure

Before you drop a database with one or more text indexes:

1. Issue the following command for each text index:

   ```
   db2text drop index index_name for text
   ```

   If you do not use this command, you must delete all of the index files manually in `index_directory` and `index_work_directory`.
2. If the indexes belonging to the dropped database were created during an automatic update, you will need to edit the scheduler file `ctedem.dat`.
   a. Enter the following:
      - For UNIX:

        ```
        db2text stop force
        cd ~/sqllib/db2ext
        ```

      - For Windows:

        ```
        db2text stop force
        cd db2_install_path\sqllib\db2_instance_name\db2ext
        ```

   b. Open the file `ctedem.dat` in the directory and remove all entries referring to the dropped database.

Chapter 67. Installation return codes on Windows

The following topic contains information regarding installation return codes on Windows.

`setup.exe` return codes on Windows

The `setup.exe` program returns the following codes in the `setup.log` file:

- 0 Success
- -1 General error
- -2 Invalid mode
- -3 Required data not found in the `setup.iss` file
- -4 Not enough memory available
- -5 File does not exist
- -6 cannot write to the response file
- -7 Unable to write to the log file
- -8 Invalid path to the install shield silent response (.iss) file
- -9 Not a valid list type
- -10 Data type invalid
- -11 Unknown error during setup
- -12 Dialog boxes are out of order
- -51 Cannot create the specified folder
- -52 Cannot access the specified file or folder
- -53 Invalid option selected

Chapter 68. Hints and tips

This topic contains useful hints and tips for topics such as authorization, language, modifying db2cli.ini files, and client server interoperability.

Authorization
: When you issue the **`DB2TEXT START`** command on Windows, ensure that you are a member of the Administrators group. Otherwise, the **`DB2TEXT START`** command fails and returns the following message: `CTE0218 Function "OpenSCManager()" failed with error code "5"`.

Authorization
: On Windows. ensure that the Net Search Extender instance service `DB2EXT-`*`DB2_instance_name`* runs under a user account rather than the system account. If you run under the system account, you can not enable your database.

Language
: As Net Search Extender event log messages are always displayed in the DB2 server language, event log messages for commands issued from the DB2 Command Line Processor might be displayed in a different language.

Modifying the `db2cli.ini` file
: If you modified the `db2cli.ini` file and have problems using Net Search Extender, restore the original version of `db2cli.ini`.

Client server interoperability
: You must install the same Net Search Extender fix pack level on both the server and client.

  Interoperability between client to server, and vice versa, is only possible for the supported platforms.

Log size
: If the indexing process is not completed because the error and warning messages require more space than the available DB2 log size, DB2 will rollback the whole transaction and not commit the log table entries. This means that you will not be able to see the entries.

  Check the DB2 documentation for information about how to increase your transaction log size to avoid this situation.

DBCS object names
: If you use DBCS object names in **`db2text`** administration commands, it is necessary to enclose those names in double quotation marks to avoid uppercase transformation.

Incremental index update on nicknames
: If initial updates on two or more indexes are started at the same time, the update command might return a SQL0803N error. In this case, try the update command again.

Single masking and character normalization
: Words like 'über' are normalized and are stored in the index in their normalized form ('ueber'). Therefore, if you issue a query containing single character masking, for example '_ber', you will not find 'über'.

Using duplicate cache column names
: If you use duplicate cache column names, you will not get an error during

text index creation or index update, but you will not be able to search. When you try to search, you will get an SQL error message stating that duplicate columns were used.

Incorrect shared memory size
If the specified maximum cache size is too small in a **db2text activate cache** command, the required cache size that is displayed in the resulting error message is incorrect.

Check for the correct cache size by using the DB2EXT.MAXIMUM_CACHE_SIZE and DB2EXT.PCTFREE functions. Correct the maximum cache size by using the **db2text alter index** command and activate the cache again.

Unicode tables in a non Unicode database
You are not allowed to create a text index on a Unicode table if your database does not support Unicode.

Incorrect code page for LANG variable in a query on Linux
If you use a 7-bit ASCII code page for the **LANG** variable setting in a query, the following error is displayed: `SearchString parse: check LANG & locale charmap values.`.

To avoid this happening, change your **LANG** variable to a 8-bit **LANG** value, restart DB2, and try searching again.

File access problems
If you cannot access files, for example, a document model, a text index, or a thesaurus, ensure that you used the correct password and have the correct authorization for running the Net Search Extender instance services. This applies especially to shared resources on mapped network drives.

Cache cannot be used
During search or activate cache, the following error message might be displayed: `CTE0271 Cache not usable, DEACTIVATE and ACTIVATE RECREATE required.` To solve this problem, check your system settings and try to increase the amount of paging space and free memory.

Instance services not dropped after an uninstall
If the instance services are not dropped after you uninstall Net Search Extender, use the following tool to drop the services manually: `ctereg` *instancename* `unregister`. For example, `ctereg db2-0 unregister`.

cteprcrx terminates abnormally on UNIX
Check that the used instance owner does not have a separate fenced user ID. To check this, open the file `.fenced` in *instance_home_dir*`/sqllib/adm` and check if the instance owner is also the fenced user.

Index update fails with SQL0668N
If the index update fails with SQL0668N (rc=1), the access to the text-maintained staging table was blocked due to a database operation on the base table that requires integrity processing for dependent tables.

Use the **RESET PENDING** command to unblock the staging table, and re-run the index update command

When SQL scalar search functions cannot be used on tables spanning multiple partitions
In situations where the text search function does not directly reference the partitioned table or if it is a member of a subselect that contains an OUTER

JOIN clause, the search will return an error (SQL0270N rc=109). If this occurs, attempt to manually rewrite the query to a different form, and reissue it.

CTE0249 error when verifying partitioned database environment setup
On Windows, if you verify the partitioned database environment setup and encounter the error `CTE0249 Executable program "cteprisc" terminated abnormally`, perform the following actions:

1. For each of the computers in the partitioned database environment configuration, check that the environment has been prepared for a partitioned DB2 server, as instructed in . Specifically, ensure that the **Trust computer for delegation** check box on the **General** tab of each computer's account **Properties** dialog box in the Active Directory Users and Computers console is checked.
2. Ensure that the Windows service "DB2 Remote Command Server" has been started on all the participating computers.

CTE0150E error when creating a text index on a range partitioned table without the ADMINISTRATION TABLES IN clause
If you try to create a text index on a range partitioned table make sure to specify the ADMINISTRATION TABLES IN clause without fail. See CTE0150E for more information.

Part 15. Appendixes

Appendix A. Overview of the DB2 technical information

DB2 technical information is available in multiple formats that can be accessed in multiple ways.

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command-line help
  - Command help
  - Message help

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hardcopy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at ibm.com. Access the DB2 Information Management software library site at http://www.ibm.com/software/data/sw-library/.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this email address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss. English and translated DB2 Version 10.1 manuals in PDF format can be downloaded from www.ibm.com/support/docview.wss?rs=71&uid=swg27009474.

Although the tables identify books available in print, the books might not be available in your country or region.

The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed below.

Note: The *DB2 Information Center* is updated more frequently than either the PDF or the hard-copy books.

Table 12. DB2 technical information

Name	Form Number	Available in print	Availability date
Administrative API Reference	SC27-5506-00	Yes	July 28, 2013
Administrative Routines and Views	SC27-5507-00	No	July 28, 2013
Call Level Interface Guide and Reference Volume 1	SC27-5511-00	Yes	July 28, 2013
Call Level Interface Guide and Reference Volume 2	SC27-5512-00	Yes	July 28, 2013
Command Reference	SC27-5508-00	Yes	July 28, 2013
Database Administration Concepts and Configuration Reference	SC27-4546-00	Yes	July 28, 2013
Data Movement Utilities Guide and Reference	SC27-5528-00	Yes	July 28, 2013
Database Monitoring Guide and Reference	SC27-4547-00	Yes	July 28, 2013
Data Recovery and High Availability Guide and Reference	SC27-5529-00	Yes	July 28, 2013
Database Security Guide	SC27-5530-00	Yes	July 28, 2013
DB2 Workload Management Guide and Reference	SC27-5520-00	Yes	July 28, 2013
Developing ADO.NET and OLE DB Applications	SC27-4549-00	Yes	July 28, 2013
Developing Embedded SQL Applications	SC27-4550-00	Yes	July 28, 2013
Developing Java Applications	SC27-5503-00	Yes	July 28, 2013
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-5504-00	No	July 28, 2013
Developing RDF Applications for IBM Data Servers	SC27-5505-00	Yes	July 28, 2013
Developing User-defined Routines (SQL and External)	SC27-5501-00	Yes	July 28, 2013
Getting Started with Database Application Development	GI13-2084-00	Yes	July 28, 2013

Table 12. DB2 technical information (continued)

Name	Form Number	Available in print	Availability date
Getting Started with DB2 Installation and Administration on Linux and Windows	GI13-2085-00	Yes	July 28, 2013
Globalization Guide	SC27-5531-00	Yes	July 28, 2013
Installing DB2 Servers	GC27-5514-00	Yes	July 28, 2013
Installing IBM Data Server Clients	GC27-5515-00	No	July 28, 2013
Message Reference Volume 1	SC27-5523-00	No	July 28, 2013
Message Reference Volume 2	SC27-5524-00	No	July 28, 2013
Net Search Extender Administration and User's Guide	SC27-5526-00	No	July 28, 2013
Partitioning and Clustering Guide	SC27-5532-00	Yes	July 28, 2013
pureXML Guide	SC27-5521-00	Yes	July 28, 2013
Spatial Extender User's Guide and Reference	SC27-5525-00	No	July 28, 2013
SQL Procedural Languages: Application Enablement and Support	SC27-5502-00	Yes	July 28, 2013
SQL Reference Volume 1	SC27-5509-00	Yes	July 28, 2013
SQL Reference Volume 2	SC27-5510-00	Yes	July 28, 2013
Text Search Guide	SC27-5527-00	Yes	July 28, 2013
Troubleshooting and Tuning Database Performance	SC27-4548-00	Yes	July 28, 2013
Upgrading to DB2 Version 10.5	SC27-5513-00	Yes	July 28, 2013
What's New for DB2 Version 10.5	SC27-5519-00	Yes	July 28, 2013
XQuery Reference	SC27-5522-00	No	July 28, 2013

Table 13. DB2 Connect-specific technical information

Name	Form Number	Available in print	Availability date
DB2 Connect Installing and Configuring DB2 Connect Personal Edition	SC27-5516-00	Yes	July 28, 2013
DB2 Connect Installing and Configuring DB2 Connect Servers	SC27-5517-00	Yes	July 28, 2013
DB2 Connect User's Guide	SC27-5518-00	Yes	July 28, 2013

Displaying SQL state help from the command line processor

DB2 products return an SQLSTATE value for conditions that can be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure

To start SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.
For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Accessing different versions of the DB2 Information Center

Documentation for other versions of DB2 products is found in separate information centers on ibm.com®.

About this task

For DB2 Version 10.1 topics, the *DB2 Information Center* URL is http://pic.dhe.ibm.com/infocenter/db2luw/v10r1.

For DB2 Version 9.8 topics, the *DB2 Information Center* URL is http://pic.dhe.ibm.com/infocenter/db2luw/v9r8/.

For DB2 Version 9.7 topics, the *DB2 Information Center* URL is http://pic.dhe.ibm.com/infocenter/db2luw/v9r7/.

For DB2 Version 9.5 topics, the *DB2 Information Center* URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Terms and conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about 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 grant 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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country/region 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, changes, or both in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by IBM are provided for convenience only and do not in any manner serve as an endorsement of those

Trademarks

Index

Printed in USA

SC27-5526-00

Spine information:

IBM DB2 10.5 for Linux, UNIX, and Windows

Net Search Extender Administration and User's Guide