DB2 TextExtender

(Assignment 2)

Relational Database Design
RELDES/2i1071/2i1417/2i4217 autumn term 2006

v. 2.3

Table of Contents

Introduction

DB2 TextExtender is an extension to DB2 Universal Database. It can be described as a “full-text retrieval program” that makes it possible to search within text documents that are either stored in the database or stored as external files outside of the database management system’s control. TextExtender performs the search of text documents by searching in a predefined index. Searching is in other words not being done in the actual document.

TextExtender consists mainly of three parts. These are:

Command line interpreter. This is a command prompt for performing commands that are specific to the TextExtender (e.g. for indexing documents). Many of the commands used for searching in the documents and creating tables etc. are mainly done from DB2’s ordinary environment (Command Editor or Control Center).

User-defined functions (UDFs). Functions included in ordinary SQL queries to allow searching in text documents. Since the UDFs are additions to SQL, the searching is performed as usual from the Command Editor and it is also possible to integrate queries on ordinary columns (e.g. name, date etc.) together with the search in text documents (see appendix on searching).

Application programming interface (API). These are functions that can be called from C-programs in order to search in text documents and show the result from the search.

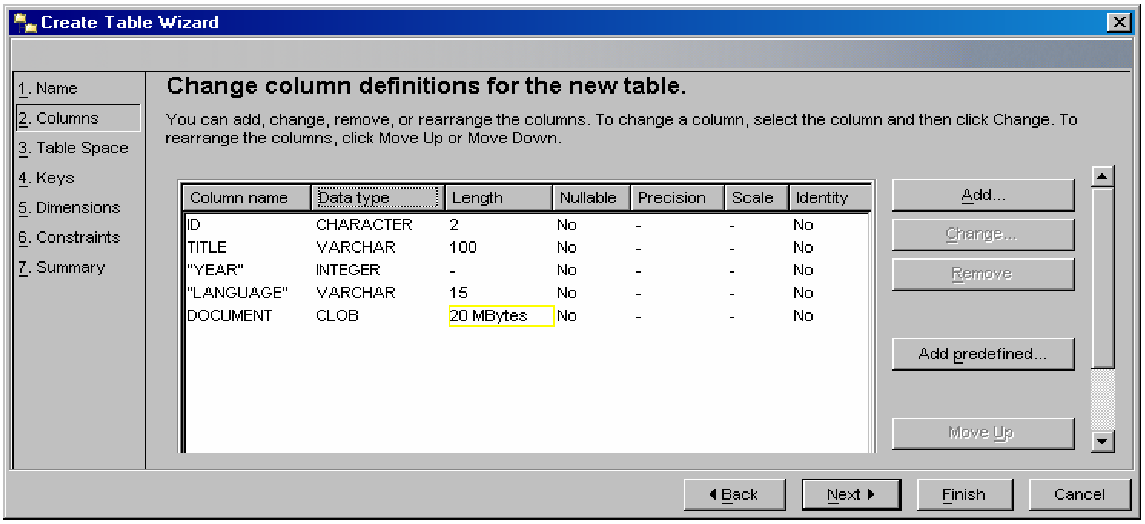
Working with DB2 TextExtender

This chapter describes how to create a full-text database consisting of master theses and information about their author(s), title, year and language. This full-text database will later be used to perform searching with regard to the content of the theses.

Creating the database

1. Create a database and name it TXTDB.

2. Create a table named PAPER according to the definition below. Define the same columns as in the picture below.



Change column definitions for the new table.

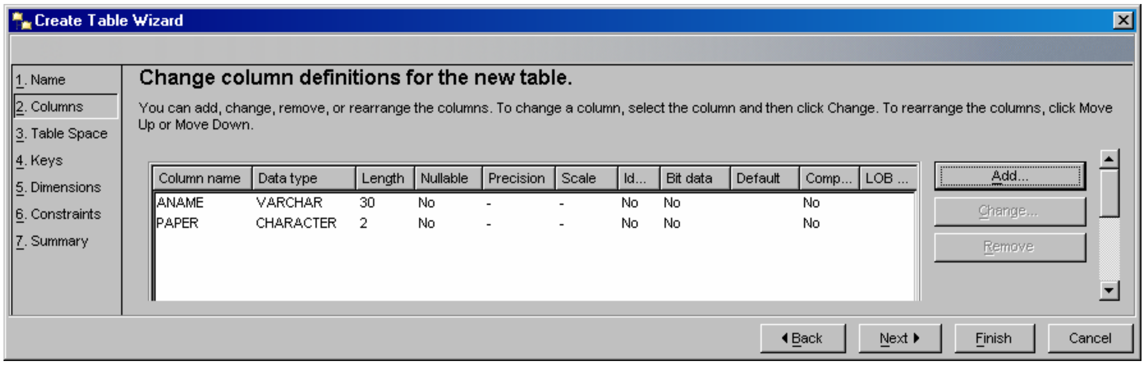
You can add, change, remove, or rearrange the columns. To change a column, select the column and then click Change. To rearrange the columns, click Move Up or Move Down.

Column name	Data type	Length	Nullable	Precision	Scale	Identity
ID	CHARACTER	2	No	-	-	No
TITLE	VARCHAR	100	No	-	-	No
"YEAR"	INTEGER	-	No	-	-	No
"LANGUAGE"	VARCHAR	15	No	-	-	No
DOCUMENT	CLOB	20 MBytes	No	-	-	No

The DOCUMENT column is going to be of the data type CLOB (Character Large Object). This column is going to contain the files with the theses, which are going to be imported later. Change the size of the CLOB (to approx. 20 Mb) so we are ensured that the documents will fit. Make sure the column is neither logged nor compact.

- Define the column ID as the table's primary key.

3. Also create the table AUTHOR according to the following.

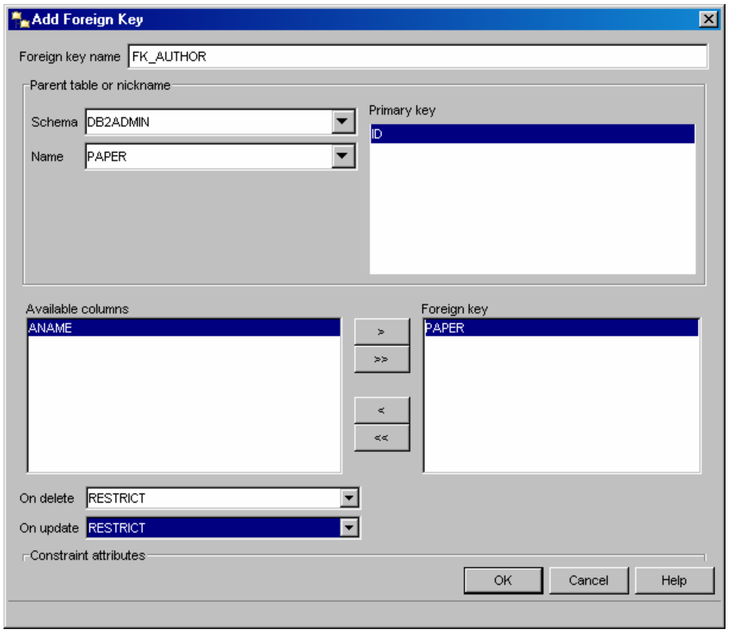


Change column definitions for the new table.

You can add, change, remove, or rearrange the columns. To change a column, select the column and then click Change. To rearrange the columns, click Move Up or Move Down.

Column name	Data type	Length	Nullable	Precision	Scale	Id...	Bit data	Default	Comp...	LOB ...
ANAME	VARCHAR	30	No	-	-	No	No		No	
PAPER	CHARACTER	2	No	-	-	No	No		No	

- Define both columns as the table's primary key.
- Also add a foreign key according to this:



Foreign key name FK_AUTHOR

Schema DB2ADMIN

Name PAPER

Primary key ID

Available columns ANAME

Foreign key PAPER

On delete RESTRICT

On update RESTRICT

4. Create a folder on your removable disk (D:) which could be named TXTemp: **D:\TXTemp.** Create a subfolder to TXTemp named Docs: **D:\TXTemp\Docs.**

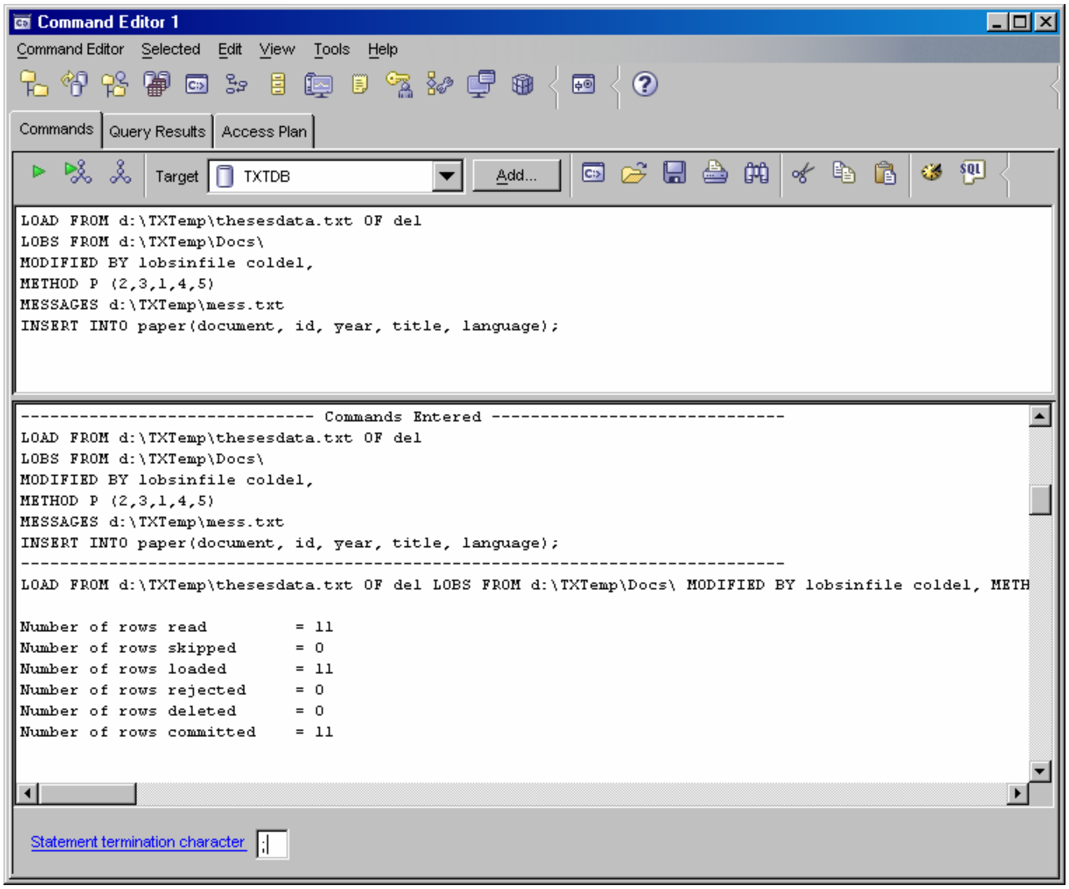
5. Copy the master theses files (*.rtf, *.html and *.doc) from the folder: \\Db-srv-1\StudentCourseMaterial\RELDESautumn2006\Lab2\Docs on the DB-SRV-1 server to the Docs folder you have just created. See the *Introduction to DB2* compendium for information about how to log in and get access to files on the DB-SRV-1 server.

6. Go to a **Command Editor** and connect to the database with the following command:

   ```
   CONNECT TO txtdb
   ```

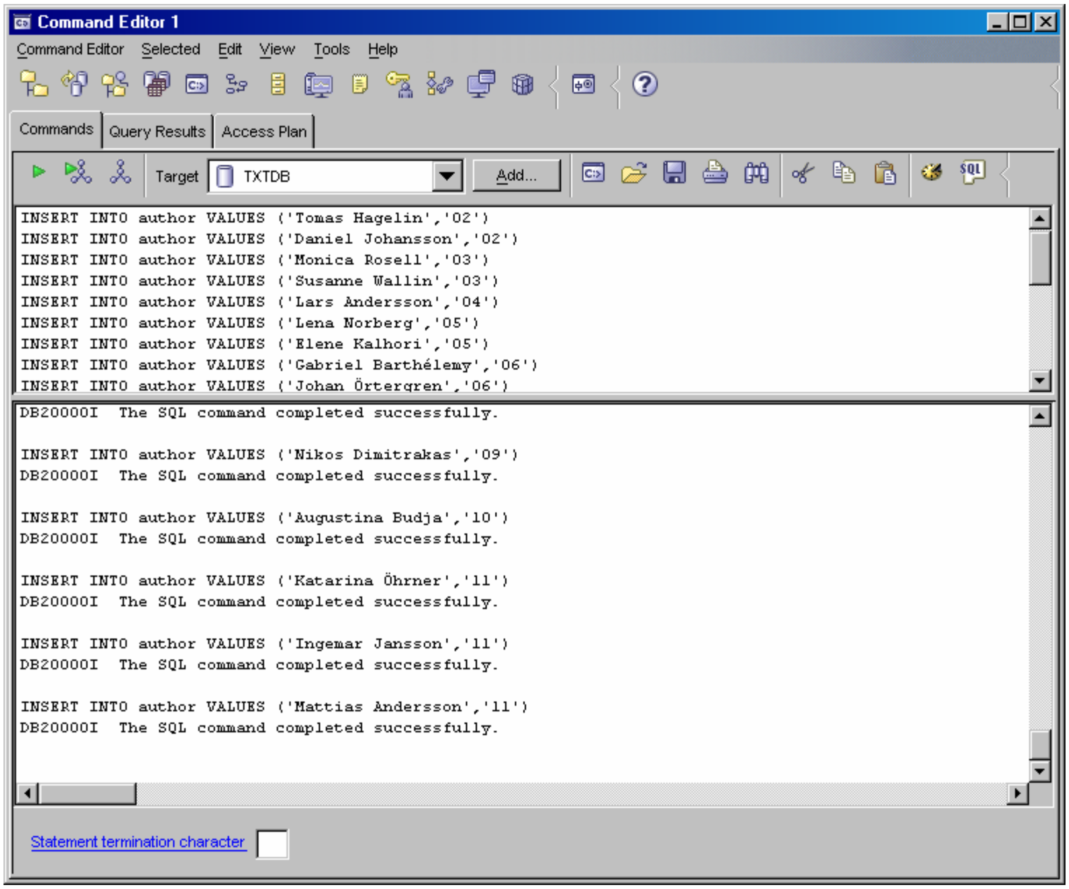
7. Fill the tables created previously with the documents and other data by running the special LOAD command and then the INSERT statements from *populate.txtdb.script* in **Command Editor**. The *populate.txdb.script* file can be found in the folder: \\Db-srv-1\StudentCourseMaterial\RELDESautumn2006\Lab2. You will need to copy this file together with *thesesdata.txt* and *mess.txt* to your TXTemp folder created in the step 4 before you can execute the LOAD and INSERT statements as shown in the figures below.

   Note: When you execute INSERT statements, you have to check if you have the correct settings for the *Statement Termination Character!* See the *Introduction to DB2* compendium for information about how to change these settings.



```
LOAD FROM d:\TXTemp\thesesdata.txt OF del
LOBS FROM d:\TXTemp\Docs\
MODIFIED BY lobsinfile coldel,
METHOD P (2,3,1,4,5)
MESSAGES d:\TXTemp\mess.txt
INSERT INTO paper(document, id, year, title, language);
```

```
------------------------------ Commands Entered ------------------------------
LOAD FROM d:\TXTemp\thesesdata.txt OF del
LOBS FROM d:\TXTemp\Docs\
MODIFIED BY lobsinfile coldel,
METHOD P (2,3,1,4,5)
MESSAGES d:\TXTemp\mess.txt
INSERT INTO paper(document, id, year, title, language);
------------------------------------------------------------------------------
LOAD FROM d:\TXTemp\thesesdata.txt OF del LOBS FROM d:\TXTemp\Docs\ MODIFIED BY lobsinfile coldel, METH

Number of rows read         = 11
Number of rows skipped      = 0
Number of rows loaded       = 11
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 11
```



```
INSERT INTO author VALUES ('Tomas Hagelin','02')
INSERT INTO author VALUES ('Daniel Johansson','02')
INSERT INTO author VALUES ('Monica Rosell','03')
INSERT INTO author VALUES ('Susanne Wallin','03')
INSERT INTO author VALUES ('Lars Andersson','04')
INSERT INTO author VALUES ('Lena Norberg','05')
INSERT INTO author VALUES ('Elene Kalhori','05')
INSERT INTO author VALUES ('Gabriel Barthélemy','06')
INSERT INTO author VALUES ('Johan Örtergren','06')
```

```
DB20000I  The SQL command completed successfully.

INSERT INTO author VALUES ('Nikos Dimitrakas','09')
DB20000I  The SQL command completed successfully.

INSERT INTO author VALUES ('Augustina Budja','10')
DB20000I  The SQL command completed successfully.

INSERT INTO author VALUES ('Katarina Öhrner','11')
DB20000I  The SQL command completed successfully.

INSERT INTO author VALUES ('Ingemar Jansson','11')
DB20000I  The SQL command completed successfully.

INSERT INTO author VALUES ('Mattias Andersson','11')
DB20000I  The SQL command completed successfully.
```

Explanation of the LOAD-statement:

The LOAD command is used to load CLOBs into the database. The main syntax follows:

```
LOAD FROM path
OF format-type
LOBS FROM CLOB-path
MODIFIED BY list of modifiers
METHOD column mapping method
MESSAGES filename
INSERT INTO table (list of columns)
```

- **The first path** defines where the file with the data (to be loaded into the table) is located.

- **Format-type** describes the format for the file specified above. *DEL* is a valid value and means that the file is of the format Delimited ASCII.

- **CLOB-path** defines where the CLOB-documents are located. One is expected only to write the filename in the data file that was specified in the first path. The CLOB-path is then placed before these filenames.

- **List of modifiers** defines which different type of changes that should be applied to the loaded data file (that was specified in the first path). There are several different values that are valid here. The values should be separated by a space. Valid values include *LOBSINFILE* and *COLDEL**x*** where x is the character that separates values in the data file. If *LOBSINFILE* is specified this means that the CLOB's complete name has been modified by the CLOB-path specified earlier. If *COLDEL* is specified, it means that values in the data file are separated by a comma character (,).

- **Column mapping method** specifies in which way the values from the data file should be loaded. There are three different methods. We are in this case using the method P which means that the order of columns can be specified by mentioning the position of the columns in the data file, e.g. P (1,3,2) means that column 1 should be loaded first, column 3 second and column 2 last.

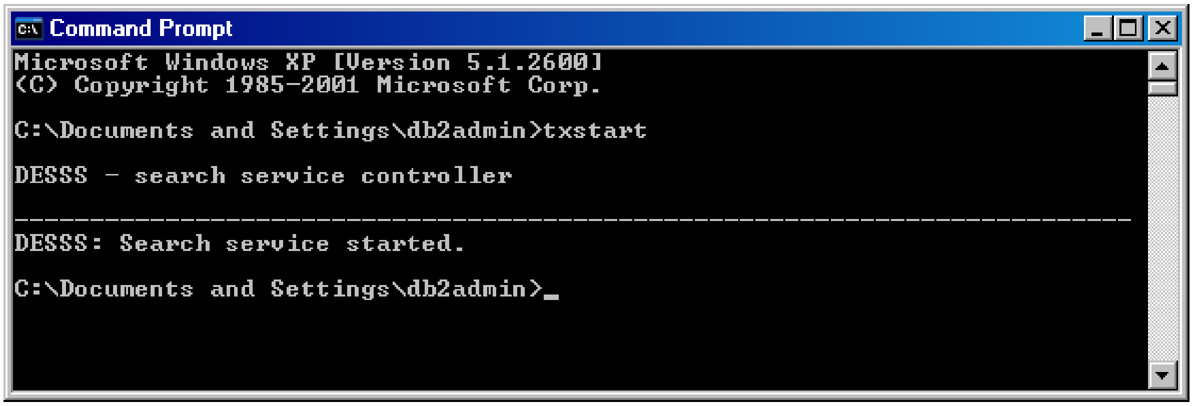
- **Filename** defines where DB2 should write its messages. The file must be created in advance (e.g. as an empty text file).

- **Table (list of columns)** specifies which table and which columns that the loaded data should be placed in. The order of the columns is mapped to the order of columns specified in the METHOD-clause.

Activate the database for TextExtender

So far we have actually only worked with standard DB2 functionality. Now it's time to get started with TextExtender.

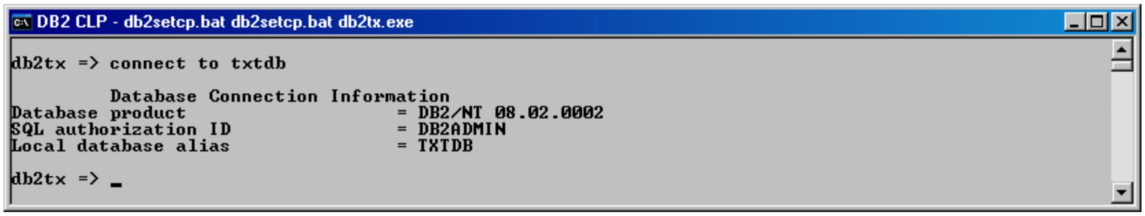
8. Open a *Command Prompt* window and give the command *txstart*. This is a process that has to be started before one can create indexes or search in indexed documents.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\db2admin>txstart

DESSS - search service controller

-----------------------------------------------------------------------------
DESSS: Search service started.

C:\Documents and Settings\db2admin>_
```

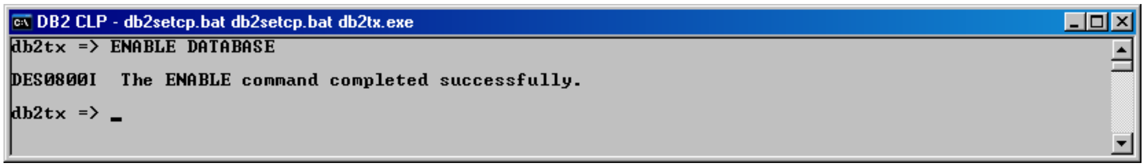
9. Open **DB2TX Command Line Processor** via the Start-menu under *Start> Programs> Databases> IBM DB2> DB2 Text Extender> DB2 Text Extender Command Line Processor*.

10. Connect to the database with the command *CONNECT TO txtdb.*



```
db2tx => connect to txtdb

         Database Connection Information
Database product                      = DB2/NT 08.02.0002
SQL authorization ID                  = DB2ADMIN
Local database alias                  = TXTDB

db2tx => _
```

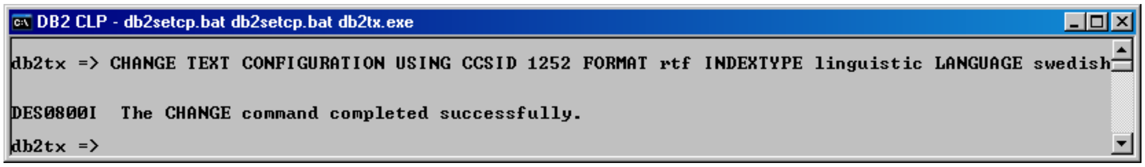
11. Give the command *ENABLE DATABASE* to activate the current database for TextExtender. This command creates a TextExtender-table named DB2TX.TextColumns. This table contains information about tables and columns that are activated for TextExtender.



```
db2tx => ENABLE DATABASE

DES0800I  The ENABLE command completed successfully.

db2tx => _
```

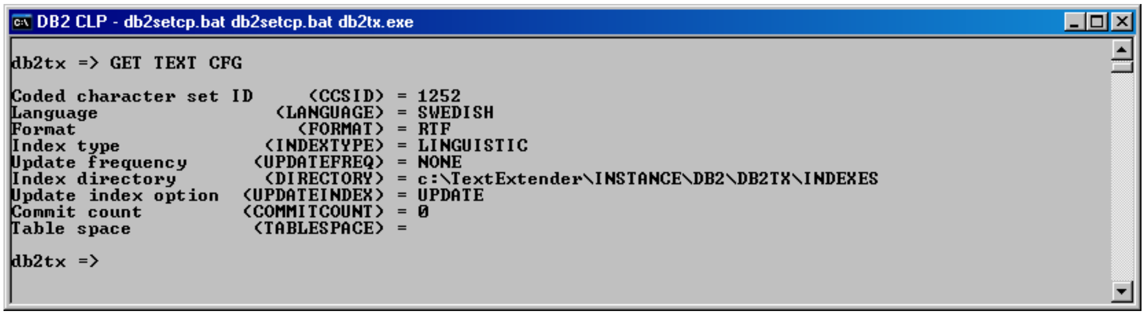
12. Before we activate any column for TextExtender we can perform some standard configuration in DB2 so that we minimize the need for changes at a later stage. We can with the following command set default values for format, character set coding, language and index type:

CHANGE TEXT CONFIGURATION USING CCSID 1252 FORMAT rtf INDEXTYPE linguistic LANGUAGE swedish



```
db2tx => CHANGE TEXT CONFIGURATION USING CCSID 1252 FORMAT rtf INDEXTYPE linguistic LANGUAGE swedish

DES0800I  The CHANGE command completed successfully.

db2tx =>
```

We are thereby assuming that these values will correspond to the majority of our documents. We can also verify that our new configurations have been registered with the command *GET TEXT CFG*:



```
db2tx => GET TEXT CFG

Coded character set ID      (CCSID) = 1252
Language                 (LANGUAGE) = SWEDISH
Format                     (FORMAT) = RTF
Index type              (INDEXTYPE) = LINGUISTIC
Update frequency       (UPDATEFREQ) = NONE
Index directory         (DIRECTORY) = c:\TextExtender\INSTANCE\DB2\DB2TX\INDEXES
Update index option   (UPDATEINDEX) = UPDATE
Commit count          (COMMITCOUNT) = 0
Table space            (TABLESPACE) =

db2tx =>
```

We chose Swedish as default language, RTF as default file format and Swedish character set. We also set linguistic to be the default index type. Differences between different index types are explained later. Setting the default file format actually has no effect. DB2 manages to recognize the file format of every file regardless of what is set as default.

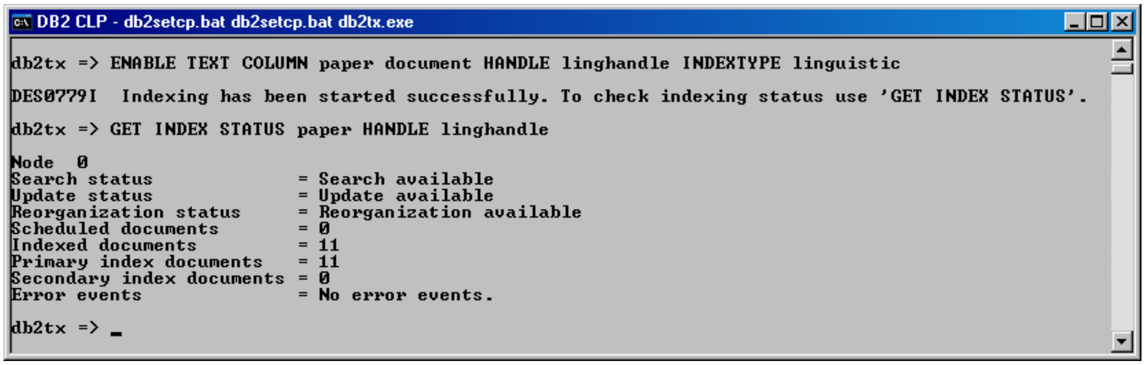
13. Give the command *ENABLE TEXT COLUMN* according to below in order to create a linguistic index.

ENABLE TEXT COLUMN paper document HANDLE linghandle INDEXTYPE linguistic

- *paper* is the name of the table.
- *document* is the name of the CLOB column.
- *linghandle* is the name of the handle for the index that will be created. The handle becomes a column in the table. The column in this case gets the name *linghandle*. We will later see how to use the handle in order to get to the index.
- *linguistic* is the index type to be used for the created index. (Observe that you can skip the INDEXTYPE-clause, if you want to use the default index type.)

14. After a few seconds you can give the following command in order to see how far the indexing has gotten:

GET INDEX STATUS paper HANDLE linghandle

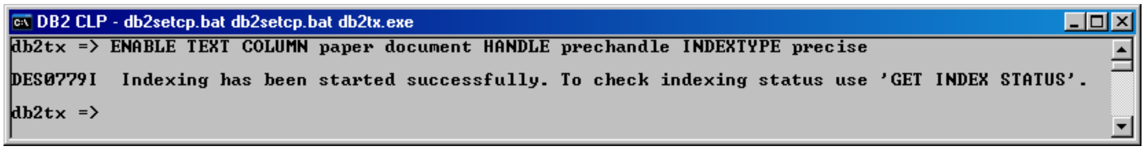


```
db2tx => ENABLE TEXT COLUMN paper document HANDLE linghandle INDEXTYPE linguistic

DES0779I  Indexing has been started successfully. To check indexing status use 'GET INDEX STATUS'.

db2tx => GET INDEX STATUS paper HANDLE linghandle

Node  0
Search status              = Search available
Update status              = Update available
Reorganization status      = Reorganization available
Scheduled documents        = 0
Indexed documents          = 11
Primary index documents    = 11
Secondary index documents  = 0
Error events               = No error events.

db2tx => _
```

15. If the indexing is not complete (i.e. there are still scheduled documents), you can wait for a while and give the command again.

16. Now do the same again to create a new index of the index type *precise*. Use the following command:

ENABLE TEXT COLUMN paper document HANDLE prechandle INDEXTYPE precise

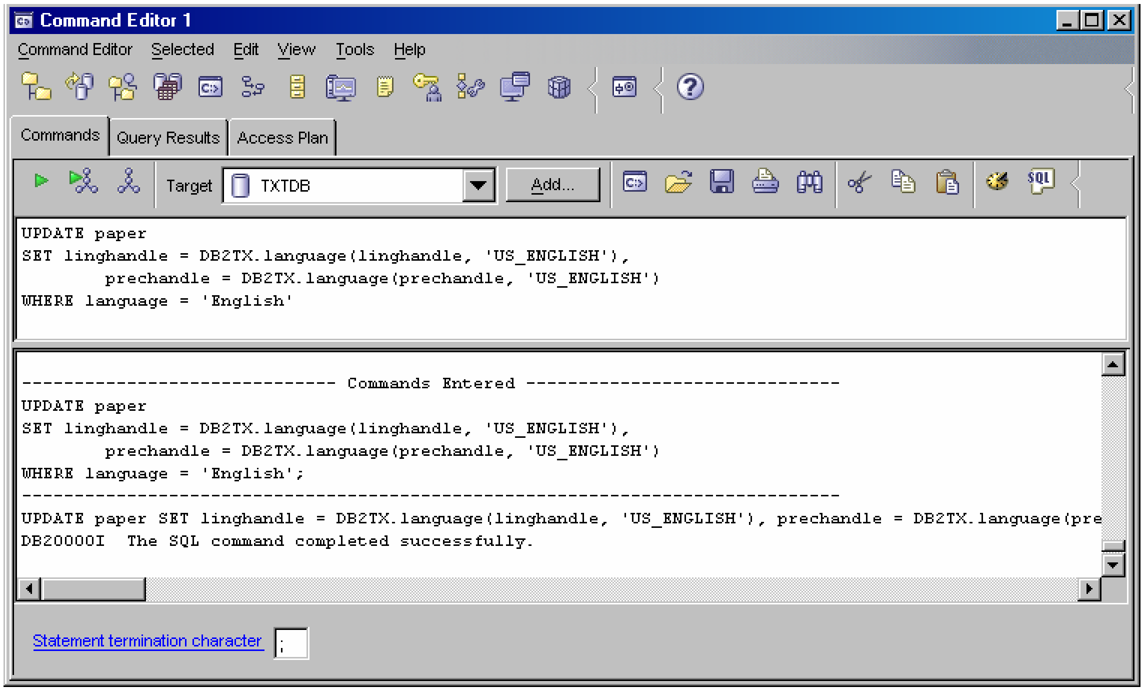


```
db2tx => ENABLE TEXT COLUMN paper document HANDLE prechandle INDEXTYPE precise

DES0779I  Indexing has been started successfully. To check indexing status use 'GET INDEX STATUS'.

db2tx =>
```

17. Since all documents are not written in the same language we must now update our handles, since the handles now believe that every document is written in Swedish (i.e. the default language). We can set the language in a handle for every specific row. We can use the following SQL statement that uses the function DB2TX.language(*handle*, *language*):

UPDATE paper
SET linghandle = DB2TX.language(linghandle, 'US_ENGLISH'),
    prechandle = DB2TX.language(prechandle, 'US_ENGLISH')
WHERE language = 'English'

Note: Check the settings for the *Statement Termination Character* before you execute this query! The *Statement Termination Character* must be specified, otherwise you will not be able to run statements which are split over several lines.



Target TXTDB

```
UPDATE paper
SET linghandle = DB2TX.language(linghandle, 'US_ENGLISH'),
        prechandle = DB2TX.language(prechandle, 'US_ENGLISH')
WHERE language = 'English'
```

```
------------------------------ Commands Entered ------------------------------
UPDATE paper
SET linghandle = DB2TX.language(linghandle, 'US_ENGLISH'),
        prechandle = DB2TX.language(prechandle, 'US_ENGLISH')
WHERE language = 'English';
------------------------------------------------------------------------------
UPDATE paper SET linghandle = DB2TX.language(linghandle, 'US_ENGLISH'), prechandle = DB2TX.language(pre
DB20000I  The SQL command completed successfully.
```

Statement termination character ;

So, we have changed both handles on every row that contains an English document to new handles. The function DB2TX.language takes a handle, changes the language and returns the new handle.

Querying the database

In this section we will perform queries against our database. We will perform:

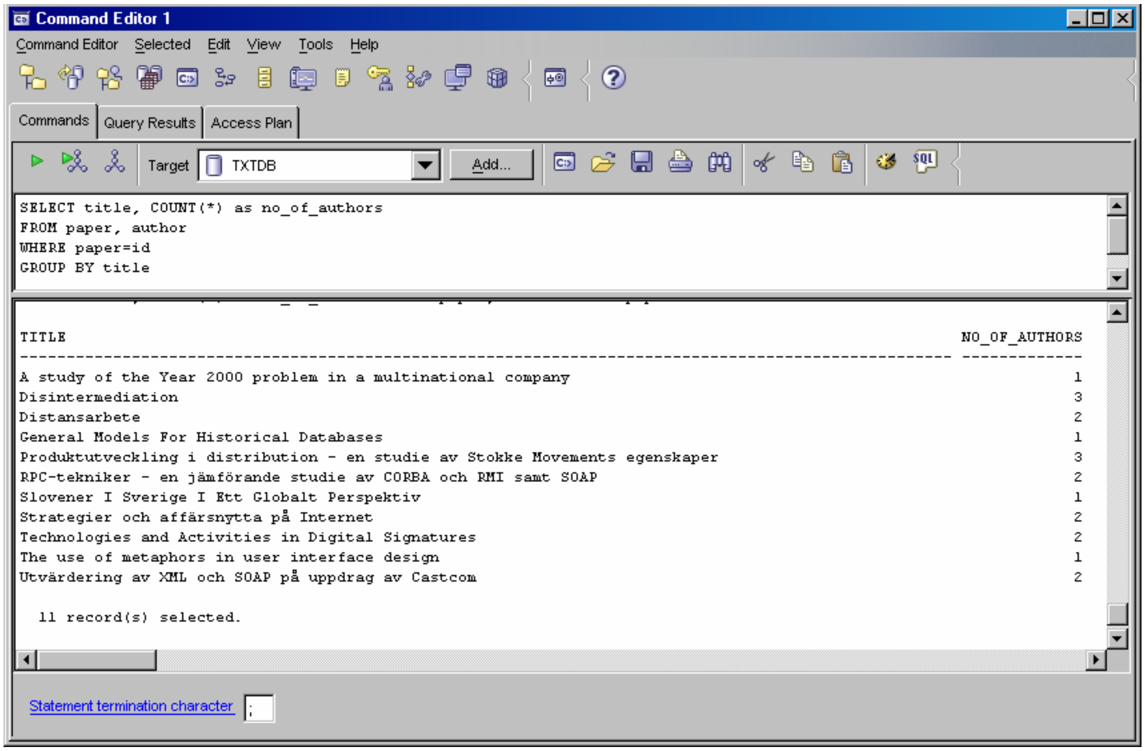
- ordinary SQL queries,
- queries about the meta information of the documents,
- queries about textual contents (both linguistic and precise) of the documents,
- queries that combine all the previous types of queries.

Ordinary SQL queries

1. We can start with the following query (that illustrates that our database is an ordinary database):

   Show the amount of authors per paper!

   SELECT title, COUNT(*) as no_of_authors
   FROM paper, author
   WHERE paper=id
   GROUP BY title



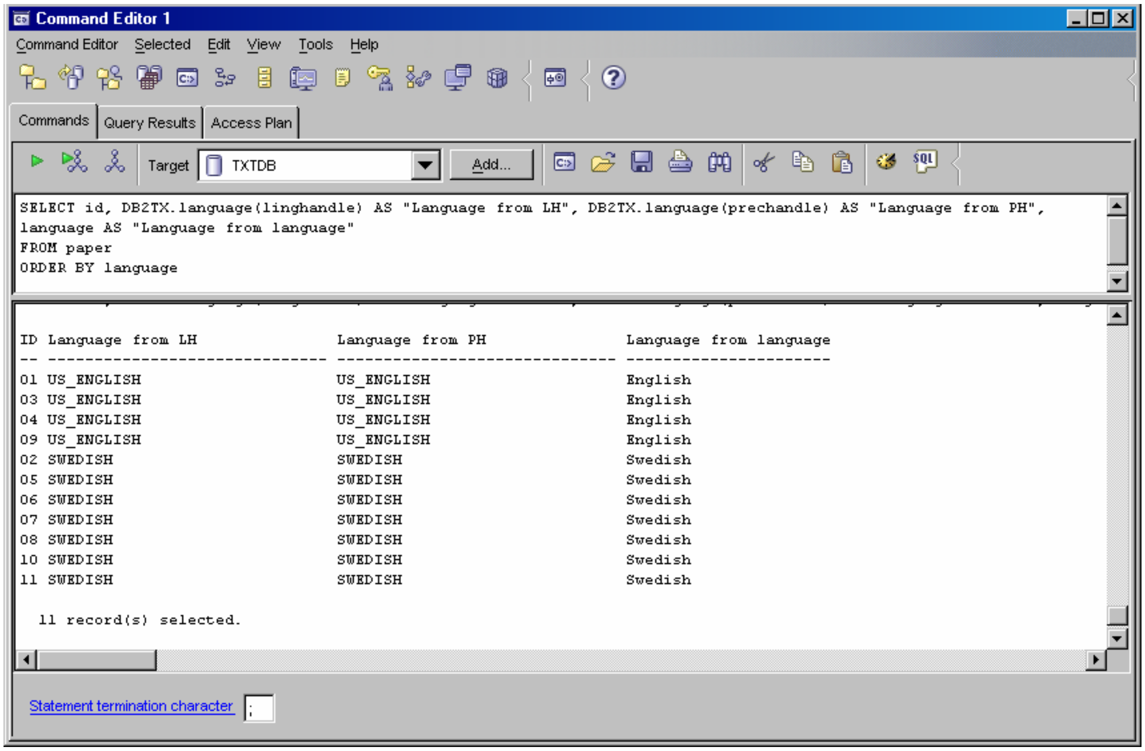
```
SELECT title, COUNT(*) as no_of_authors
FROM paper, author
WHERE paper=id
GROUP BY title
```

```
TITLE                                                                                                  NO_OF_AUTHORS
------------------------------------------------------------------------------------------------------ -------------
A study of the Year 2000 problem in a multinational company                                                        1
Disintermediation                                                                                                  3
Distansarbete                                                                                                      2
General Models For Historical Databases                                                                            1
Produktutveckling i distribution - en studie av Stokke Movements egenskaper                                        3
RPC-tekniker - en jämförande studie av CORBA och RMI samt SOAP                                                     2
Slovener I Sverige I Ett Globalt Perspektiv                                                                        1
Strategier och affärsnytta på Internet                                                                             2
Technologies and Activities in Digital Signatures                                                                  2
The use of metaphors in user interface design                                                                      1
Utvärdering av XML och SOAP på uppdrag av Castcom                                                                  2

  11 record(s) selected.
```

Queries about the documents' meta information

1. Show the language of each document (retrieved from our handles)! Here we can use the function DB2TX.language that we used earlier. We can also retrieve the language from the column language to be able to see if they correspond to each other.

   SELECT id, DB2TX.language(linghandle) AS "Language from LH",
   DB2TX.language(prechandle) AS "Language from PH",
   language AS "Language from language"
   FROM paper
   ORDER BY language



```
SELECT id, DB2TX.language(linghandle) AS "Language from LH", DB2TX.language(prechandle) AS "Language from PH",
language AS "Language from language"
FROM paper
ORDER BY language
```

```
ID Language from LH                 Language from PH                 Language from language
-- ------------------------------ ------------------------------ ----------------------
01 US_ENGLISH                       US_ENGLISH                       English
03 US_ENGLISH                       US_ENGLISH                       English
04 US_ENGLISH                       US_ENGLISH                       English
09 US_ENGLISH                       US_ENGLISH                       English
02 SWEDISH                          SWEDISH                          Swedish
05 SWEDISH                          SWEDISH                          Swedish
06 SWEDISH                          SWEDISH                          Swedish
07 SWEDISH                          SWEDISH                          Swedish
08 SWEDISH                          SWEDISH                          Swedish
10 SWEDISH                          SWEDISH                          Swedish
11 SWEDISH                          SWEDISH                          Swedish

  11 record(s) selected.
```

Queries about the documents' textual contents (both linguistic and precise)

There are several functions that can be used to run queries against the CLOB-documents' textual contents. These functions are shown in the table below.

Functions (UDF)	Purpose
CONTAINS	Evaluates a condition for every document. Returns 0 or 1, where 0 means false and 1 means true.
NO_OF_MATCHES	Returns the amount of hits for a certain condition for every document.
RANK	Returns a ranking value per document given a condition. The returned value lies between 0 and 1, where 1 is the highest value.

The conditions can contain some keywords. These keywords operate differently depending on the index type being used. The following table summarizes the keywords we will use:

	Index type	
Keyword	***Linguistic***	***Precise***
PRECISE FORM OF	Not available	Default
STEMMED FORM OF	Default	Not available
SYNONYM FORM OF	Available	Available
SOUNDS LIKE	Available	Available
IN SAME SENTENCE AS	Available	Available
IN SAME PARAGRAPH AS	Available	Available

The first four keywords can be followed by a language. If the language is left out the default language is used. It is recommended that one always specify the language, to avoid confusion. Valid values for language include *SWEDISH* and *US_ENGLISH*. (The keywords for the languages are case-sensitive!)

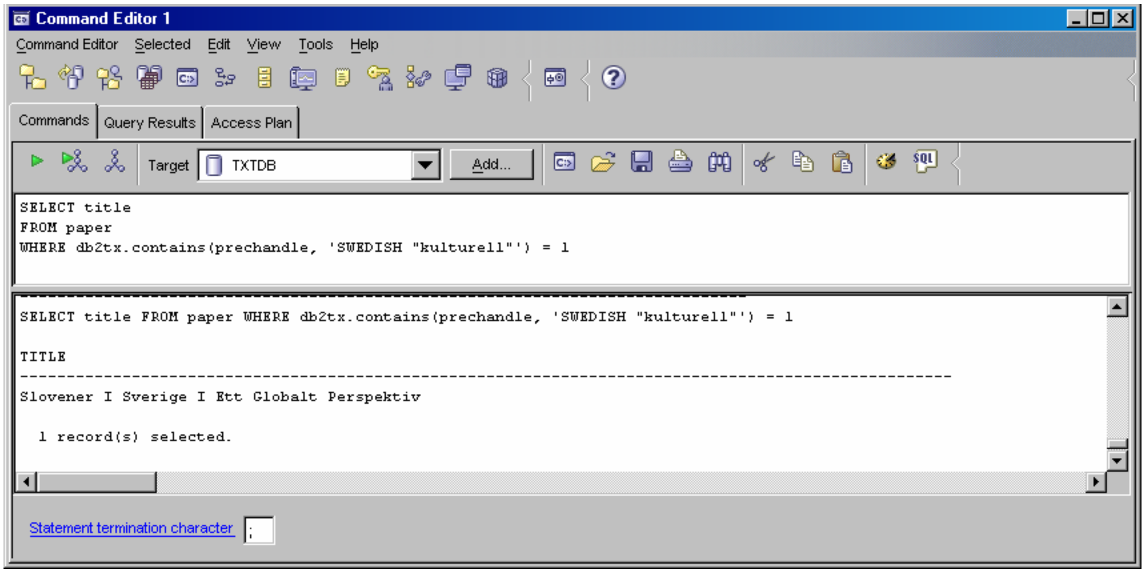
The conditions can of course also contain logical operators like *NOT*, *AND (&)* and *OR (|)*. There is also the possibility to use wild-cards like *%* and *_*.

In order to exemplify the different functions' usage we will now look at a few examples.

1. Which documents contain the Swedish word *kulturell*? In this case we want to retrieve the title of the documents that contain exactly the word *kulturell*.

   This can be done with the following SQL query:

```
SELECT title
FROM paper
WHERE db2tx.contains(prechandle, 'SWEDISH "kulturell"') = 1
```


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB
Add...
Statement termination character ;

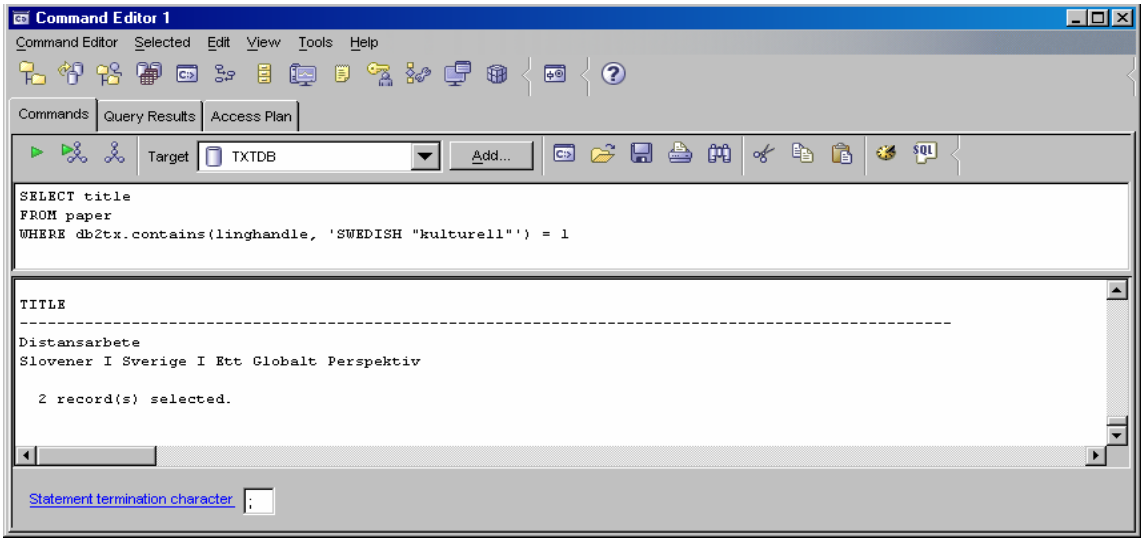
```
SELECT title
FROM paper
WHERE db2tx.contains(prechandle, 'SWEDISH "kulturell"') = 1
```

```
------------------------------------------------------------------------------
SELECT title FROM paper WHERE db2tx.contains(prechandle, 'SWEDISH "kulturell"') = 1

TITLE
----------------------------------------------------------------------------------------------------
Slovener I Sverige I Ett Globalt Perspektiv

  1 record(s) selected.
```

By using prechandle in our search we implicitly specify that we want to use the keyword PRECISE FORM OF.

2. If we now instead would like to retrieve all documents that contain some form of inflection of the word *kulturell* we could use our linghandle:

   SELECT title
   FROM paper
   WHERE db2tx.contains(linghandle, 'SWEDISH "kulturell"') = 1

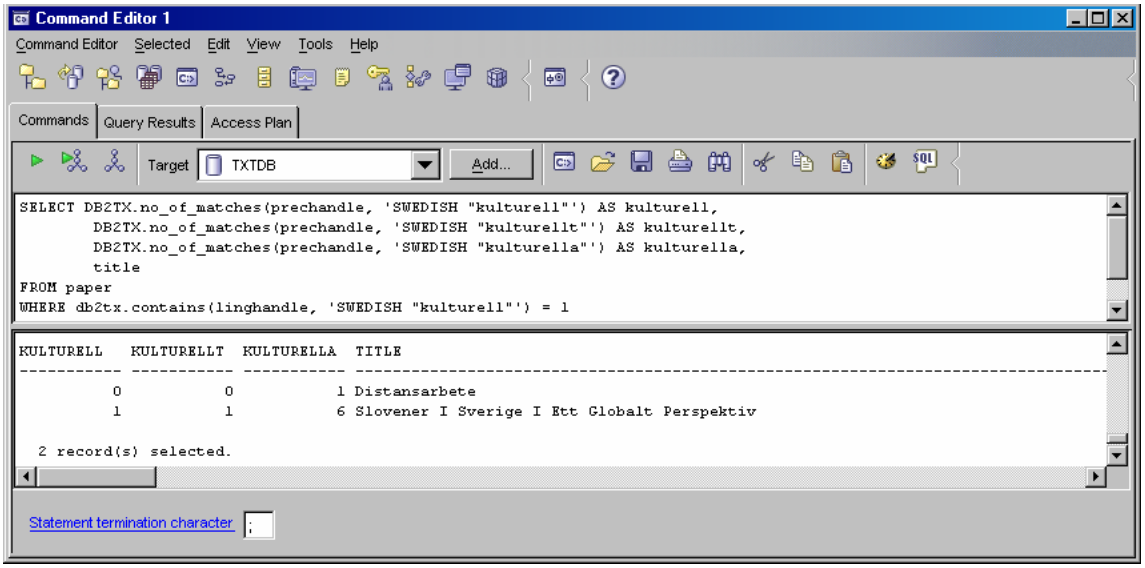


```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'SWEDISH "kulturell"') = 1
```

```
TITLE
----------------------------------------------------------------------------------------------------
Distansarbete
Slovener I Sverige I Ett Globalt Perspektiv

  2 record(s) selected.
```

3. We can modify the query so that we can see which types of inflections of the word exist in each document. We can in the SELECT clause include a column for each inflection and by using the function DB2TX.no_of_matches see the number of occurrences in each document.

   SELECT DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturell"') AS kulturell,
   DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturellt"') AS kulturellt,
   DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturella"') AS kulturella,
   title
   FROM paper
   WHERE db2tx.contains(linghandle, 'SWEDISH "kulturell"') = 1



```
SELECT DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturell"') AS kulturell,
        DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturellt"') AS kulturellt,
        DB2TX.no_of_matches(prechandle, 'SWEDISH "kulturella"') AS kulturella,
        title
FROM paper
WHERE db2tx.contains(linghandle, 'SWEDISH "kulturell"') = 1
```

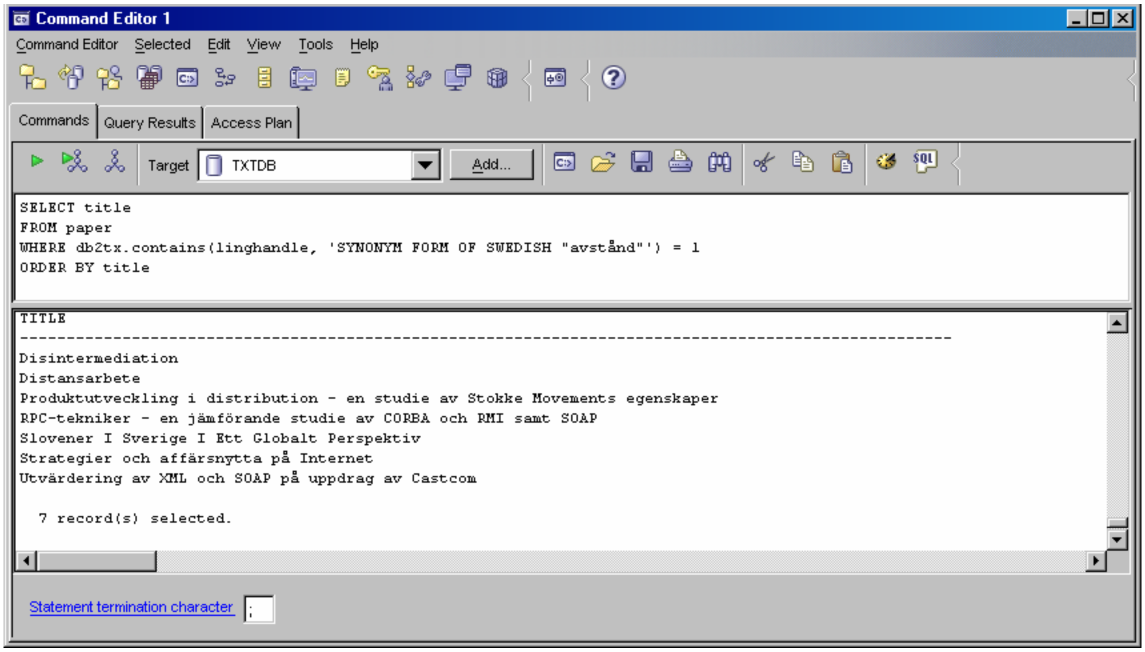
```
KULTURELL   KULTURELLT  KULTURELLA  TITLE
----------- ----------- ----------- ------------------------------------------------------------------------------
          0           0           1 Distansarbete
          1           1           6 Slovener I Sverige I Ett Globalt Perspektiv

  2 record(s) selected.
```

So, by doing this we can see that the paper "Distansarbete" contains the inflected form *kulturella* 1 time, which makes it pass this condition, but it will not pass the condition of the precise form *kulturell*.

We can also choose to look for a general concept and not a specific word. To search for concepts, use the synonym forms of a word.

4. Show all documents that mention the Swedish word *avstånd* (distance) or its synonyms in some form. (We order the titles in the alphabetical order to make it easier to compare the query results in this section.)

   SELECT title
   FROM paper
   WHERE db2tx.contains(linghandle, 'SYNONYM FORM OF SWEDISH "avstånd"') = 1
   ORDER BY title

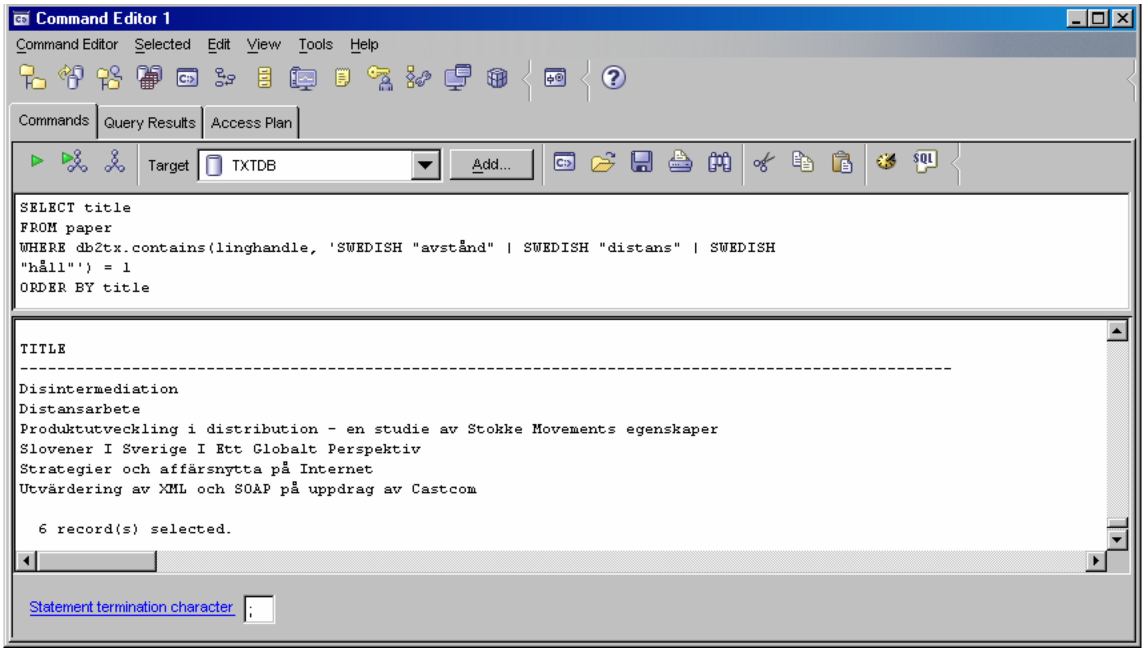


```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'SYNONYM FORM OF SWEDISH "avstånd"') = 1
ORDER BY title
```

```
TITLE
----------------------------------------------------------------------------------------------------
Disintermediation
Distansarbete
Produktutveckling i distribution - en studie av Stokke Movements egenskaper
RPC-tekniker - en jämförande studie av CORBA och RMI samt SOAP
Slovener I Sverige I Ett Globalt Perspektiv
Strategier och affärsnytta på Internet
Utvärdering av XML och SOAP på uppdrag av Castcom

  7 record(s) selected.
```

5. If we now want to check only some specific synonyms like *distans*, *avstånd* and *håll* but not the rest of the synonyms, it is favourable to use the logical operator OR (|):

   SELECT title
   FROM paper
   WHERE db2tx.contains(linghandle, 'SWEDISH "avstånd" | SWEDISH "distans" | SWEDISH "håll"') = 1
   ORDER BY title


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB
Add...
Statement termination character ;

```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'SWEDISH "avstånd" | SWEDISH "distans" | SWEDISH
"håll"') = 1
ORDER BY title
```

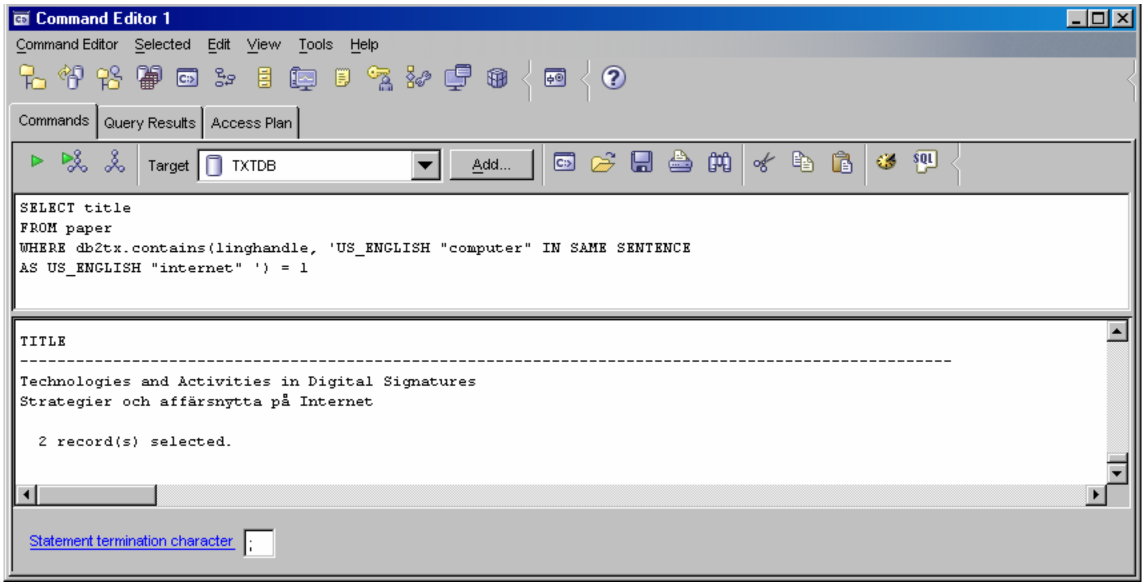
```
TITLE
----------------------------------------------------------------------------------------------------
Disintermediation
Distansarbete
Produktutveckling i distribution - en studie av Stokke Movements egenskaper
Slovener I Sverige I Ett Globalt Perspektiv
Strategier och affärsnytta på Internet
Utvärdering av XML och SOAP på uppdrag av Castcom

  6 record(s) selected.
```

Please note that the paper "RPC-tekniker…" is not included in this result because it does not contain any of our three synonyms. The reason why this document passed the condition in the query 4 is that it contains the word *intervall* which is another synonym of the word *avstånd*.

Another possibility that exists is to check whether two or more subconditions are fulfilled in the same sentence or paragraph:

6. Show the papers that contain the English words *computer* and *internet* in the same sentence! This can be done with the following query:

   SELECT title
   FROM paper
   WHERE db2tx.contains(linghandle, 'US_ENGLISH "computer" IN SAME SENTENCE AS US_ENGLISH "internet" ') = 1

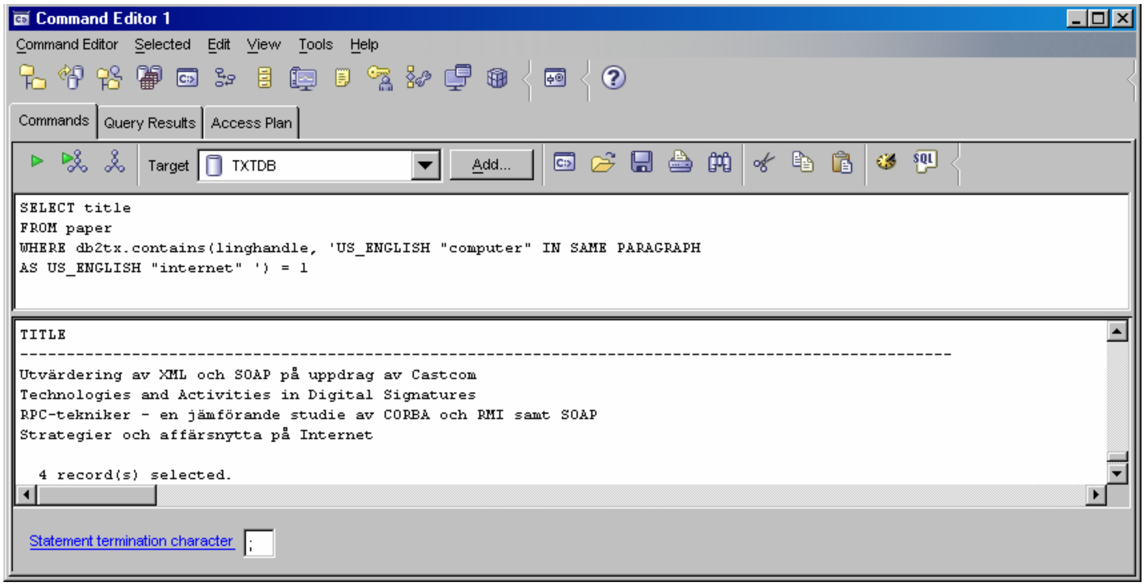

Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB
Add...
Statement termination character ;

```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'US_ENGLISH "computer" IN SAME SENTENCE
AS US_ENGLISH "internet" ') = 1
```

```
TITLE
----------------------------------------------------------------------------------------------------
Technologies and Activities in Digital Signatures
Strategier och affärsnytta på Internet

  2 record(s) selected.
```

7. Now show the papers that contain the English words *computer* and *internet* in the same paragraph instead! (So that we really can see there is a difference!)

```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'US_ENGLISH "computer" IN SAME PARAGRAPH
AS US_ENGLISH "internet" ') = 1
```

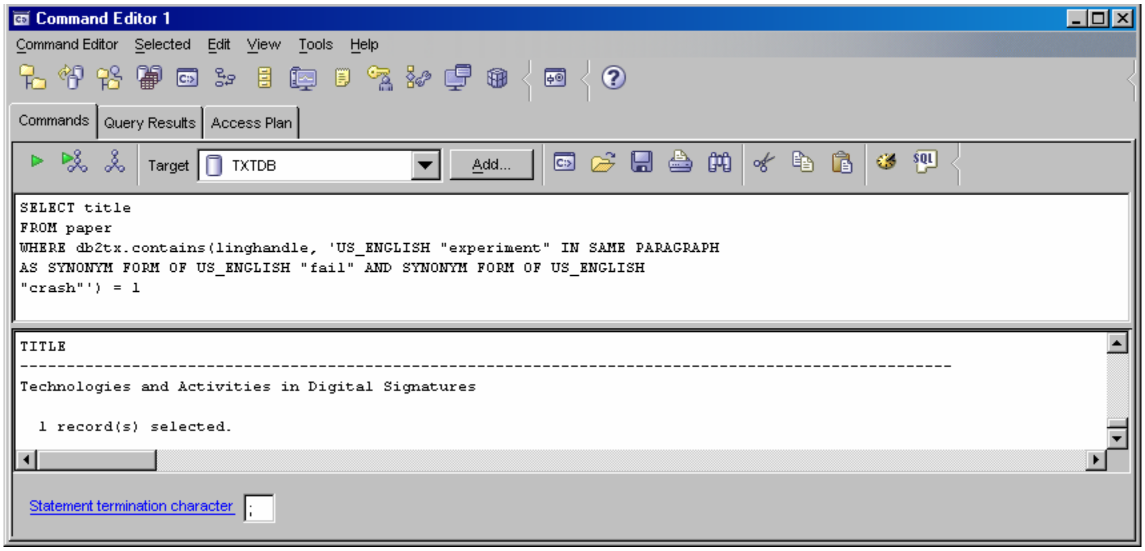


```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'US_ENGLISH "computer" IN SAME PARAGRAPH
AS US_ENGLISH "internet" ') = 1
```

```
TITLE
----------------------------------------------------------------------------------------------------
Utvärdering av XML och SOAP på uppdrag av Castcom
Technologies and Activities in Digital Signatures
RPC-tekniker - en jämförande studie av CORBA och RMI samt SOAP
Strategier och affärsnytta på Internet

  4 record(s) selected.
```

8. We can also check the following: Which papers have the word *experiment* (or an inflection of it) and synonyms of the words *fail* and *crash* in the same paragraph.

```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'US_ENGLISH "experiment" IN SAME PARAGRAPH
AS SYNONYM FORM OF US_ENGLISH "fail" AND SYNONYM FORM OF US_ENGLISH
"crash"') = 1
```



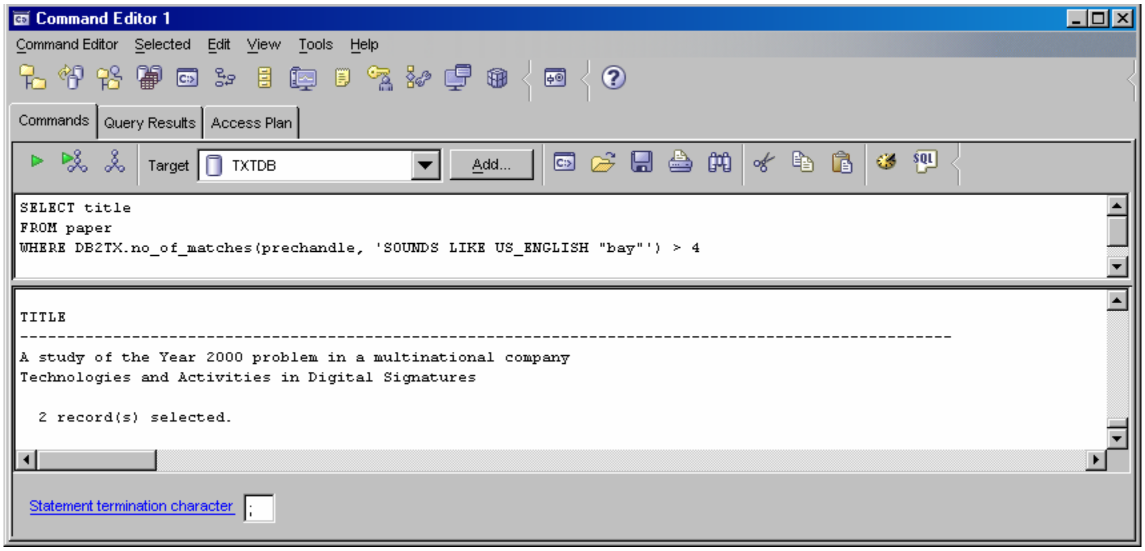
```
SELECT title
FROM paper
WHERE db2tx.contains(linghandle, 'US_ENGLISH "experiment" IN SAME PARAGRAPH
AS SYNONYM FORM OF US_ENGLISH "fail" AND SYNONYM FORM OF US_ENGLISH
"crash"') = 1
```

```
TITLE
----------------------------------------------------------------------------------------------------
Technologies and Activities in Digital Signatures

  1 record(s) selected.
```

If you'd like to search for a word that you are not quite sure of how it is spelled, you could use the keyword *SOUNDS LIKE*:

9. Find all documents that contain at least 5 occurrences of words that sound like the English word *bay*!

   SELECT title
   FROM paper
   WHERE DB2TX.no_of_matches(prechandle, 'SOUNDS LIKE US_ENGLISH "bay"') > 4


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB Add...

```
SELECT title
FROM paper
WHERE DB2TX.no_of_matches(prechandle, 'SOUNDS LIKE US_ENGLISH "bay"') > 4
```

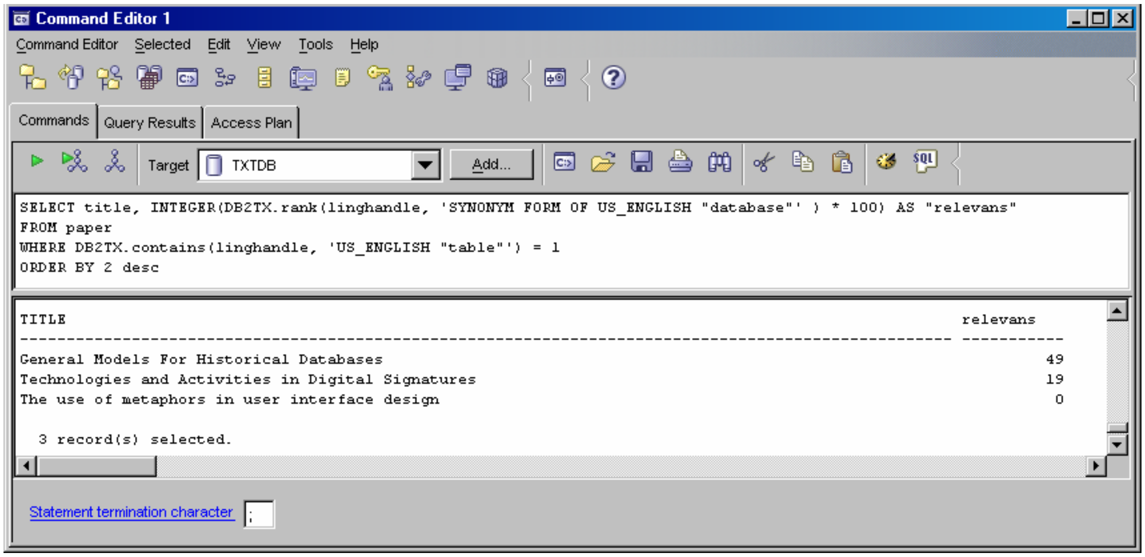
```
TITLE
----------------------------------------------------------------------------------------------------
A study of the Year 2000 problem in a multinational company
Technologies and Activities in Digital Signatures

  2 record(s) selected.
```

Statement termination character ;

You can use the function DB2TX.rank in order to get the database management system to calculate a ranking value per document according to a condition:

10. Retrieve all documents that contain the English word *table* and rank the documents according to relevance to the word *database*!

    *SELECT title, INTEGER(DB2TX.rank(linghandle, 'SYNONYM FORM OF US_ENGLISH "database"') * 100) AS "relevans"*
    FROM paper
    WHERE DB2TX.contains(linghandle, 'US_ENGLISH "table"') = 1
    ORDER BY 2 desc


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB Add...
Statement termination character ;

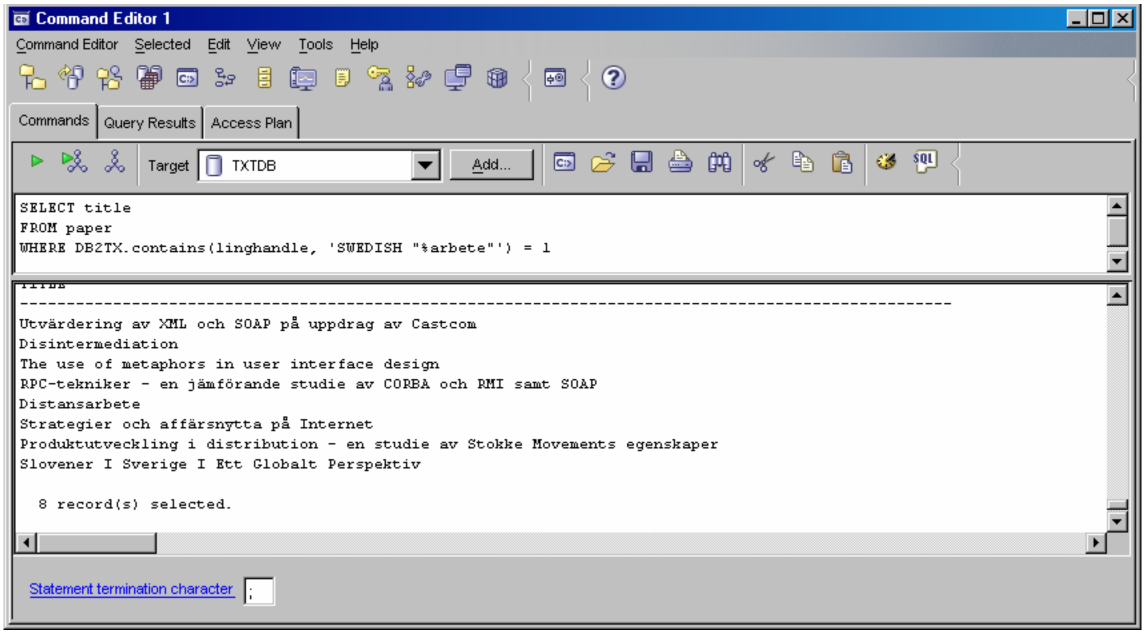
```
SELECT title, INTEGER(DB2TX.rank(linghandle, 'SYNONYM FORM OF US_ENGLISH "database"' ) * 100) AS "relevans"
FROM paper
WHERE DB2TX.contains(linghandle, 'US_ENGLISH "table"') = 1
ORDER BY 2 desc
```

```
TITLE                                                                                                  relevans
---------------------------------------------------------------------------------------------------- -----------
General Models For Historical Databases                                                                      49
Technologies and Activities in Digital Signatures                                                            19
The use of metaphors in user interface design                                                                 0

  3 record(s) selected.
```

Last, but not least we can look at an example with the use of wild-cards:

11. Show all documents that contain some word that ends with the Swedish word *arbete*!

```
SELECT title
FROM paper
WHERE DB2TX.contains(linghandle, 'SWEDISH "%arbete"') = 1
```

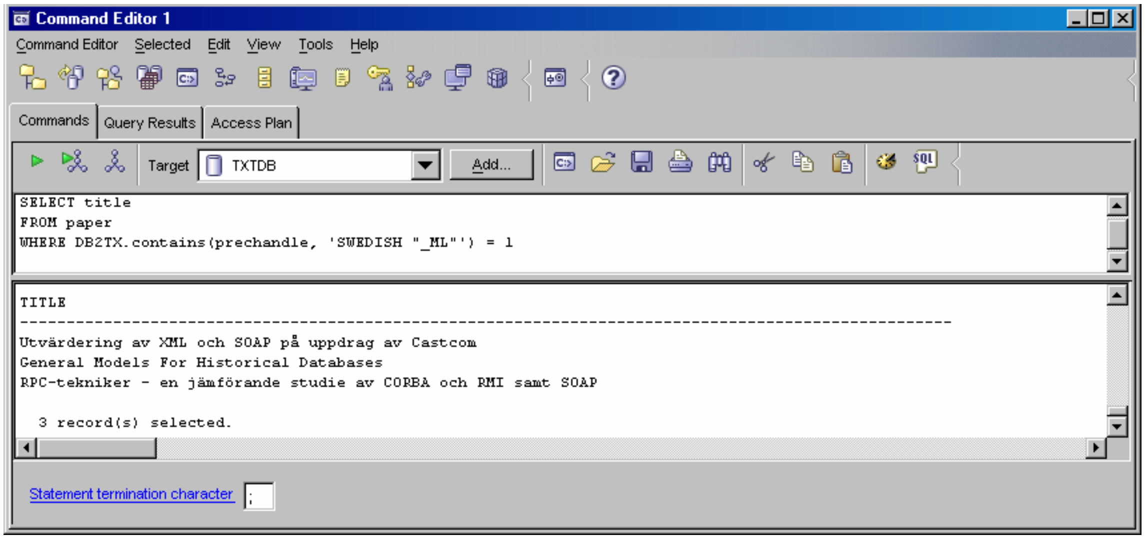

Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB Add...
Statement termination character ;

```
SELECT title
FROM paper
WHERE DB2TX.contains(linghandle, 'SWEDISH "%arbete"') = 1
```

```
TITLE
----------------------------------------------------------------------------------------------------
Utvärdering av XML och SOAP på uppdrag av Castcom
Disintermediation
The use of metaphors in user interface design
RPC-tekniker - en jämförande studie av CORBA och RMI samt SOAP
Distansarbete
Strategier och affärsnytta på Internet
Produktutveckling i distribution - en studie av Stokke Movements egenskaper
Slovener I Sverige I Ett Globalt Perspektiv

  8 record(s) selected.
```

12. Show all documents that contain an abbreviation of three letters that end with ML!

> *SELECT title*
> *FROM paper*
> *WHERE DB2TX.contains(prechandle, 'SWEDISH "_ML"') = 1*


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB
Add...
Statement termination character ;

```
SELECT title
FROM paper
WHERE DB2TX.contains(prechandle, 'SWEDISH "_ML"') = 1
```

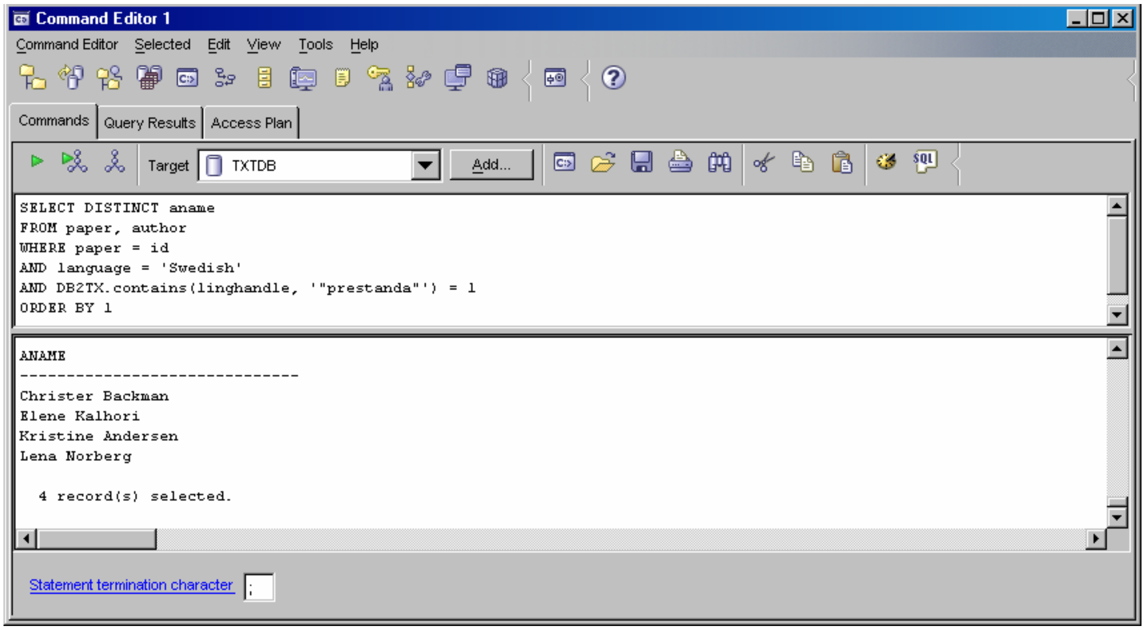
```
TITLE
----------------------------------------------------------------------------------------------------
Utvärdering av XML och SOAP på uppdrag av Castcom
General Models For Historical Databases
RPC-tekniker - en jämförande studie av CORBA och RMI samt SOAP

  3 record(s) selected.
```

(Two of them contain XML and one DML!)

Queries that combine all types of queries

1. Which authors have written a paper in Swedish that contains the Swedish word *prestanda*? Sort by name!

   SELECT DISTINCT aname
   FROM paper, author
   WHERE paper = id
   AND language = 'Swedish'
   AND DB2TX.contains(linghandle, '"prestanda"') = 1
   ORDER BY 1



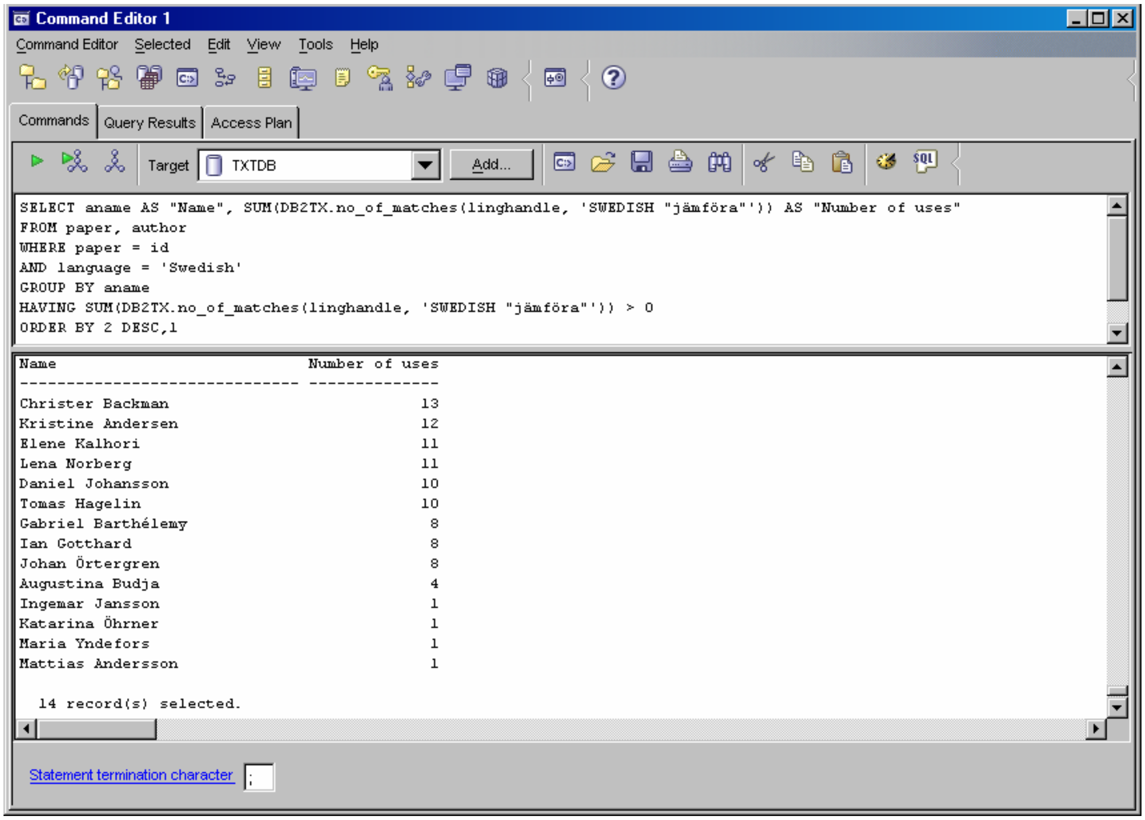
```
SELECT DISTINCT aname
FROM paper, author
WHERE paper = id
AND language = 'Swedish'
AND DB2TX.contains(linghandle, '"prestanda"') = 1
ORDER BY 1
```

```
ANAME
------------------------------
Christer Backman
Elene Kalhori
Kristine Andersen
Lena Norberg

  4 record(s) selected.
```

Statement termination character ;

2. For each author, show the name and the amount of times this author has used the Swedish word *jämföra* (in some inflected form) in Swedish papers! Authors that have not used this word should not be shown in the result. The author that has used the word most should be shown first. If more than one author has used the word equal amount of times they should be sorted on name.

   SELECT aname AS "Name", SUM(DB2TX.no_of_matches(linghandle, 'SWEDISH "jämföra"'))
   AS "Number of uses"
   FROM paper, author
   WHERE paper = id
   AND language = 'Swedish'
   GROUP BY aname
   HAVING SUM(DB2TX.no_of_matches(linghandle, 'SWEDISH "jämföra"')) > 0
   ORDER BY 2 DESC,1


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands Query Results Access Plan
Target TXTDB Add...

```
SELECT aname AS "Name", SUM(DB2TX.no_of_matches(linghandle, 'SWEDISH "jämföra"')) AS "Number of uses"
FROM paper, author
WHERE paper = id
AND language = 'Swedish'
GROUP BY aname
HAVING SUM(DB2TX.no_of_matches(linghandle, 'SWEDISH "jämföra"')) > 0
ORDER BY 2 DESC,1
```

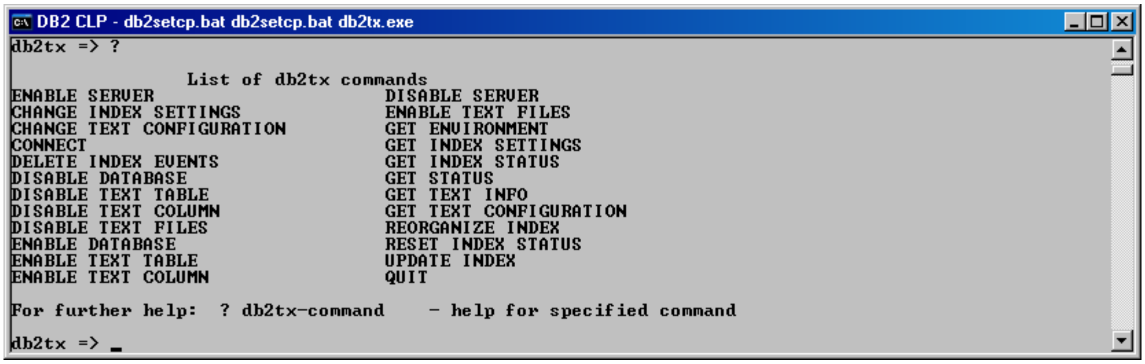
```
Name                           Number of uses
------------------------------ --------------
Christer Backman                           13
Kristine Andersen                          12
Elene Kalhori                              11
Lena Norberg                               11
Daniel Johansson                           10
Tomas Hagelin                              10
Gabriel Barthélemy                          8
Ian Gotthard                                8
Johan Örtergren                             8
Augustina Budja                             4
Ingemar Jansson                             1
Katarina Öhrner                             1
Maria Yndefors                              1
Mattias Andersson                           1

  14 record(s) selected.
```

When something has gone wrong

Here is a list of commands that you can use to undo the changes in the database when something goes wrong:

Use this command	**To undo the following command**
TXSTOP	TXSTART
DISCONNECT	CONNECT TO <*database*>
DISABLE DATABASE	ENABLE DATABASE
DISABLE TEXT COLUMN <*table*> HANDLE <*handle*>	ENABLE TEXT COLUMN <*table*> <*column*> HANDLE <*handle*> INDEXTYPE <*indextype*>

For more information you can write *?* in **DB2TX Command Line Processor:**



```
db2tx => ?

                List of db2tx commands
ENABLE SERVER                     DISABLE SERVER
CHANGE INDEX SETTINGS             ENABLE TEXT FILES
CHANGE TEXT CONFIGURATION         GET ENVIRONMENT
CONNECT                           GET INDEX SETTINGS
DELETE INDEX EVENTS               GET INDEX STATUS
DISABLE DATABASE                  GET STATUS
DISABLE TEXT TABLE                GET TEXT INFO
DISABLE TEXT COLUMN               GET TEXT CONFIGURATION
DISABLE TEXT FILES                REORGANIZE INDEX
ENABLE DATABASE                   RESET INDEX STATUS
ENABLE TEXT TABLE                 UPDATE INDEX
ENABLE TEXT COLUMN                QUIT

For further help:  ? db2tx-command    - help for specified command

db2tx => _
```

Assignments

The following assignments should be solved and then sent in to the conference "RELDES Assignments" together with the execution results.

1. Which documents (the title of the papers) contain the Swedish words *Internet* or *hårddisk* and not the words *intranet* or *databas*?

2. Retrieve the title of papers that contain an exact correspondence to the Swedish word *dator* (i.e. inflected forms of the word such as *datorer, datorn* etc are not valid) and whose title contains the word *och*.

3. Which Swedish papers (titles) are written after 1999 and contain the word *term*? Show only the papers that have been written by at least 2 authors!

4. Show all papers that contain at least 3 occurrences of synonyms of the word *population* and that have the word (the synonym) in the same paragraph as the word *kunskap*.

5. Which documents are at least as relevant to the word *teknik* (or inflections of it) as the average value for Swedish documents that contain the exact word *protokoll* and the exact phrase *höga krav*? Show the title, year and amount of authors! In other words: If Swedish documents that contain *protokoll* and *höga krav* have a average relevance to the word *teknik* of value *X*, then the final result should contain documents that have a higher relevance to the word *teknik* than *X*.

Reference materials

Specifying search arguments

Search arguments are used in CONTAINS, NO_OF_MATCHES and RANK. This section uses the CONTAINS function to show different examples of search arguments in UDFs.

Searching for several terms

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

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

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

Searching with the Boolean operators

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

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

You can combine several terms using Boolean operators:

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

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

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

is evaluated as:

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

So in the following example you must include the parentheses:

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

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

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

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

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

Searching for variations of a term

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

If you are using a **linguistic** index, Text Extender searches also for variations of the terms, such as the plural of a noun, or a different tense of a verb. For example, the term drive finds text that contains "drive", "drives", "driving", "drove", and "driven":

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
             'PRECISE FORM OF "utility"') = 1
```

By contrast, this example finds occurrences of "utility" and "utilities":

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

Searching for parts of a term (character masking)

Masking characters, otherwise known as "wildcard" characters, offer a way to make a search more flexible. They represent optional characters at the front, middle, or end of a search term. They increase the number of text documents found by a search.

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

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

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

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

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

This search term finds text documents containing, for example, "username", "filename", and "table-name". % can also represent a **whole word**. The following example finds text documents containing phrases such as "graphic function" and "query function":

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

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

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

Searching for terms that already contain a masking character

If you want to search for a term that contains the "%" character or the "_" character, you must precede the character by a so-called *escape* character, and then identify the escape character using the ESCAPE keyword. For example, to search for "10% interest":

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"10!% interest" ESCAPE "!"') = 1
```

The escape character in this example is "!".

Searching for terms in any sequence

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

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

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

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

Searching for terms in the same sentence or paragraph

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

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

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

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

Searching for synonyms of terms

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

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

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

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

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

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

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

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

Making a linguistic search

Text Extender offers powerful linguistic processing for making a search based on the search terms that you provide. The linguistic functions are applied when the index is linguistic.

An example of this is searching for a plural form, such as "utilities", and finding "utility". The plural is reduced to its base form utility, using an English dictionary, before the search begins. The English dictionary, however, does not have the information for reducing variations of terms in other languages to their base form. To search for the plural of a term in a different language you must use the dictionary for that language.

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

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

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

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

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

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

Searching with the Boolean operator NOT

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

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

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

You cannot use the NOT operator in combination with IN SAME SENTENCE AS or IN SAME PARAGRAPH AS, neither can you use it with SYNONYM FORM OF . You can use the NOT operator only with a search-primary, that is, you cannot freely combine the &, |, and NOT operators.

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

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

Allowed is:

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

Searching for similar-sounding words

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

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

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

Indexing

An IR-system (information retrieval system) performs searches by matching the search argument against words that exists in a pre-defined index. It would take way too much time to sequentially scan through all words in the documents for every search. The index contains relevant words/terms that have been extracted from the text documents and every word/term is stored together with information about the documents in which they reside. In this way it is easy to localize the documents that match the search argument.

By saying relevant words/terms we mean those that are typical for the document. Words that are not typical for the document (i.e. often occurring words like prepositions and pronouns – *and, is, with, without* etc.) will not be indexed. These words exist in a *Stop Word List* that is used to filter out irrelevant words before words/terms are stored in the index.

There are different types of indexes that can be used. These affect among other things which different types of searches that can be performed. The most usual index types are *precise*, *linguistic*, and *ngram*.

Linguistic index

Before words/terms are stored in an index the documents undergo a linguistic analysis. This is also done with search arguments before a search. The most important steps in this process can be divided into four phases:

1. **Basic text analysis:**
    - The text is analyzed so that words that contain non alphanumerical characters will be stored in the index as one term (e.g. ”mother-in-law”, ”$12,234”).
    - Upper-case letters are changed to lower-case letters (e.g.”About” is changed to ”about”).
    - The text of the document is analyzed to identify where each sentence starts and ends. This is done so that searching for occurrences of terms within the same sentence can be performed.
2. **Reduction to base form:** All words are transformed into base form.
3. **Decomposition:** Compound words (e.g. “sometime”) are indexed in their entirety, but also as their elements (“some” and “time”).
4. **Stop-word filtering:** Irrelevant words are filtered out by comparing the document text or search argument with a *Stop Word List* containing often occurring words.

Below you’ll find a summarization of the indexing process for a *linguistic index*.

Table 1. Term extraction for a linguistic index

Document text	Term in index	Linguistic processing
Hat	Hat	**Basic text analysis** (normalization)
mice swam stole	mouse, swim, steal	**Reduction to base form**
butterfly homebase	butterfly, butter, fly, homebase, home, base	**Decomposition**
A report about Mars	report, Mars	**Stop-word filtering**. The stopwords are: a, about

Precise index

Words/terms are stored in the index exactly as they occur in the document text. The same goes for search arguments, i.e. only exact correspondence between the indexed word and the search argument results in a match. The advantage with this type of index is that matching becomes more precise and that indexing and searching is faster.

The process of indexing and searching with this type of index contains the following phases:

- **Word and sentence separation.** Words and sentences are identified by analysis.
- **Stop-word filtering.**

Below you'll find a summarization of the indexing process for a precise *index*.

Table 2. Term extraction for a precise index

Document text	Term in index	Linguistic processing
Hat	Hat	**No normalization**
mice swam stole	Mice, Swam, stole	**No reduction to base form**
butterfly homebase	Butterfly, homebase	**No decomposition**
A report about Mars	report, Mars	**Stop-word filtering.** The stopwords are: a, about