Text Search Fast Indexer

Contents

Searching on a Content Manager 8 repository using the fast indexer

The text-search fast indexer runs document retrieval and content conversion processes in parallel which significantly speeds up the process of full-text search indexing.

The capture process is managed by IBM Content Collector and scales horizontally by adding additional servers. Content Manager can store over 200 documents per second, so does not limit the required volume. It is the rate of the full-text indexing that is the decisive factor, which limits the daily volume of e-mails that can be indexed.

With its ability to run processes in parallel, the indexing rate achieved by the fast indexer is at least ten times greater than that achieved by a standard indexer that runs in one process.

Installing the fast indexer

The fast indexer file are provided when the text-user exit is installed. Unless you stated otherwise, the text-search user-exit is installed at C:\Program Files\IBM\ICCTextSearch on Windows® and at /opt/IBM/ICCTextSearch on AIX® or Solaris.

About this task

The text-search user-exit must be properly installed and configured before the fast indexer is installed.

The fast indexer files are copied to the fastidx directory under the ICCTextSearch directory when you run the IBM Content Collector text-search user-exit installation wizard.

The IBM Content Collector text-search user-exit installation wizard performs the following steps:

1. The installation wizard copies the following fast indexer shared libraries from the fastidx subdirectory to the DB2® function directory of the instance.
   - For AIX or Solaris, it copies:
     – ICMFetchFile
     – ICMFetchFilter1
     – GenTie2

     to, for example:
     `/home/db2inst1/sqllib/function`

     For Windows, it copies:
     – ICMFetchFile.dll
     – ICMFetchFilter1.dll
     – GenTie2.dll

     to, for example:

```
C:\Program Files\IBM\sqllib\function
```

- The installation wizard gives these files read and execution permissions.

2. The installation wizard also runs the following DB2 commands in order to connect the fast indexer with DB2.
   a. `db2 connect to <connection> user <user> using <password>`
   b. `db2 set current schema <schema>`
   c. `db2 bind GenTie2.bnd qualifier <user>`
   d. `db2 –tvf ICMFetchFilter1.ddl`
   e. `db2 –tvf ICMFetchFile.ddl`
   f. `db2 –tvf GenTie2.ddl`

Setting parameters and environment variables

The fast indexer uses the settings that are specified in the file called emsutils.properties. This file contains numerous setup parameters that control the indexing.

About this task

The installation package contains a sample file called emsutil.properties.original. For the fast indexer to find this property file, it must be copied and renamed, and an environment variable must point to the file's location.

1. Make a copy of the file emsutil.properties.original in the fastidx directory and name it emsutil.properties.
2. Edit the emsutil.properties file.

   Check that the following parameters are definitely set:

   - Set **`database.name`** to **ICMNLSDB**, or to the name of the library server database if you changed the default.
   - Set **`database.schema`** to **ICMADMIN**, or to the schema if you changed the default.
   - Set **`tempdir`** to the path where the XML output is written for each converted document, for example for Windows, to `C:\\temp\\items`, and for AIX or Solaris to `/temp/items`. To achieve the best performance, use cached or striped DASD so that the entire document batch that is being converted can be held in memory.

     Note: All Windows path names specified in parameters require two backward slashes (\\). AIX and Solaris path names only take a single forward slash (/).
   - Set **`debug`** and **`debug_udf`** to 0 unless debugging is needed.
   - Set **`model.file`** to
     – For Windows:

       ```
       "C:\\Program Files\\IBM\\ICCTextSearch\\model\\afu_mail_entire.xml"
       ```

       Note: Quotes (″) must be used if the path contains blanks.
     – For AIX and Solaris:

       ```
       /opt/IBM/ICCTextSearch/model/afu_mail_entire.xml
       ```

   Adapt the following parameter settings depending on your needs by changing the defaults:

   - Set **`batchsize`** to minimize the number of times the index is updated. The batch size should be set to a relatively high number such as 50,000 (default

size) or at least to 10% of the expected number of e-mails per day. Even if only a small number of documents are expected per day, **batchsize** should not be set to less than 5000 except for testing purposes.

- Set **reorg.count** to control the frequency of index reorganizations (when the secondary index is merged into the primary index). When the number of documents indexed since the last index reorganization is greater than this value, the next index update will include the reorganization option. The optimum value varies based on CPU and I/O speed, however, with document batches of 50 K, setting reorg.count to 1.5 million (default setting) is recommended.
- Set **reorg.time** to state the earliest time that an index reorganization can occur. The format for entering is hh:mm or an asterix (*) if reorganization is independent of the time of day. The default setting is 00.00.
- Set **reorg.last.time** to state the last time that an index reorganization can occur. The format for entering is hh:mm or an asterix (*) if reorganization is independent of the time of day. The default is 23:59. The window for reorganization can cross midnight.
- Set **retry.interval** to determine the number of hours to wait before retrying to index a document that had an indexing error. The default is 8 hours. Documents will be retried at most two times. If indexing is started only once a day, the retry interval should be small enough to include errors that occur near the end of the day.

  If you are running one batch at a time in an automated script (this is not the recommended way to index), then the retry interval should probably be at least 8 hours. This would allow time to discover that a resource manager is down and get it restarted before the automatic retries.

  Documents with a timeout, 608, or max clob error will not be retried automatically.
- Set **delete.retried.events** to determine whether to delete errors in the indexing event table for documents that have been successfully retrieved after the error was written to the event table. In addition, if the third and final attempt to index the document fails, the first two error messages will be deleted. Setting **delete.retried.events** to 0 keeps the event records, and setting it to 1 (the default) deletes the event records.
- Set **conversion.timeout** to limit the time to convert documents (e-mails and attachments) to text. The default is 2 minutes, but if many or large attachments are common and the library server is slow, increase this value to 5 minutes or even more.
- Set **exclude.binary.attachments** to avoid indexing large amounts of binary strings. Setting this parameter however does have the effect that if UTF8 documents were created using Notepad or other editors, they would be skipped although they could be correctly indexed.

  Note: DBCS or MBCS plain text attachments cannot be indexed, regardless of the value of this parameter.
- Set **object.size** to limit the maximum size of the XML output created after document conversion. The default size is 25 MB. This value must match the sizes in the ICMFetchFilter and ICMFetchFile UDFs.

  If you do alter this value in the two UDFs and the properties file:

  a. Run `db2 -tvf` against both DDL files.
  b. Restart DB2.

- Set **number.threads** and **number.processes** to improve the overall performance. If retrieval is not a bottleneck, then more than 8 threads does not significantly improve the speed (default number of threads is 4), but adding an additional process will improve performance (default number of processes is 1). After the retrieval and the conversion to XML is complete, the index update can run on only one CPU. The number of threads and processes configured in emsutil.properties should be just enough to keep indexing busy. For example, converting 100 documents per second when only 40 can be indexed in this time is not reasonable.

3. Create the directory specified in **tempdir** if it does not exist.
4. Set the EMSUTIL_PATH environment variable to the fastidx directory. You must set this variable twice, once as a system variable and once in the DB2 environment.

   Important: To specify EMSUTIL_PATH as a DB2 environment variable, you must add EMSUTIL_PATH as a value to the DB2ENVLIST variable as follows:

   ```
   set EMSUTIL_PATH=C:\Program files\IBM\ICCTextSearch\fastidx
   db2set DB2ENVLIST="EMSUTIL_PATH"
   ```

   On AIX or Solaris, EMSUTIL_PATH must be exported when DB2 is started. One way to do this, is to add EMSUTIL_PATH to the cstsenv.sh script (located at <install_dir>/cfg) which is executed before DB2 is started.
5. On AIX, it is recommended that the environment variable EXTSHM is set to ON if the indexing process is to be run with multiple threads. For example:

   ```
   export EXTSHM=ON
   ```

   If this environment variable is not set, an SQL 1224 error is created.
6. After having configured Content Manager and DB2 for fast indexing, stop and restart DB2 and the indexing engine:

   ```
   db2stop force
   db2start
   db2text stop
   db2text start
   ```

Verifying configuration settings

Before you can begin indexing, check that the database is configured to enable indexing for text-search.

About this task

Consider the following:

1. The database that you define for indexing must be a UTF-8 database. If the database was not created as a UTF-8 database, you must unload the database, recreate it, and reload the data using the `db2look` and `db2move` commands.
2. The directory in which the index is located must have at least twice as much free space as the expected index size. The work and index files must be on the same volume.

   If the directories are on different volumes, a large amount of data must be copied at the end of each index update, and the entire index must be copied when the index is reorganized. If the files are on the same volume, they are simply renamed instead of being copied. If the files are on different volumes, they can be moved using `db2text alter index`.

The total space in the file system must be at least 3 times the expected index size. The index size varies by customer, but for planning purposes assume that the index size will be at least 15% of the size of the total e-mail size that is indexed.

For example, if 1 TB of e-mail will be indexed, the index size might be as large as 150 GB. The total space in the file system must be at least 450 GB.

3. On AIX and Solaris, a 64–bit database instance is required. When running a 64–bit instance, all applications that run indexing commands must use 64–bit Java™.
4. Configure to run DB2 functions and procedures on AIX or Solaris under the ID of the instance owner, not a separate fenced ID. Change the fenced ID by setting the owner on /home/db2instl/sqllib/adm/.fenced
5. Verify that the posted date of a document is converted to UTC when the document is stored in Content Manager. This ensures that, when the document is searched and displayed, the timestamp is adjusted to the local timezone of the user and will match the timestamp seen by another client application.

   This setting is in the IBM Content Collector user-exit configuration file icmfce_config.ini and can be accessed using the environment variable ICMFCE_CFG. By default the file is located at:

   - C:\Program Files\IBM\ICCTextSearch\cfg on Windows
   - /opt/IBM/ICCTextSearch/cfg on AIX or Solaris

   In the settings section, add the configuration option:

   ```
   timestamps_in_utc = 1
   ```

6. It is recommended that you set the configuration option `ReceivedDateWithoutTime` in the user-exit configuration file icmfce_config.ini. In the settings section, add the configuration option:

   ```
   ReceivedDateWithoutTime = 1
   ```

   For details on the consequences of setting this option, refer to the section on configuration options in icmfce_config.ini below.
7. Verify that the index configuration file cteixcfg.ini contains the following settings: The index configuration file is located:

   - On Windows at:

     ```
     C:\Program Files\IBM\SQLLIB\db2ext
     ```

   - On AIX and Solaris at:

     ```
     /opt/IBM/db2/V9.5/db2ext
     ```

     where the version number (above V9.5) might differ depending on the installed version.

   The settings are:

   ```
   BlockMode=ON
   BlockSize=LARGE
   RespectCase=OFF
   TreatNumbersAsWords=FALSE
   SeparateParagraphs=OFF
   SeparateSentences=TRUE
   ```

   Note: For configuration options that can be enabled, use `1`, `ON`, `yes`, or `true`. To disable the option, set its value to anything else, for example `0` or `OFF`.

   These configuration settings cannot be changed after documents have been indexed.

If the configuration is not correct when the index is created, you must drop and recreate the index.

8. Check the index definitions by entering the following:

```
db2 "select indexname, updatefrequency, commitcount, reorganizationmode,
indexdirectory, workdirectory from db2ext.ttextindexes"
```

This query returns values related to the index. Check that the following values have been set as follows:

- `updatefrequency` must be NONE
- `commitcount` must be 0
- `reorganizationmode` must be 0
- `indexdirectory` and `workdirectory` should be the same value for performance reasons and to avoid error during the copying process.

Related concepts

Other configuration options in the configuration file icmfce_config.ini
You can set configuration options to control how the user-exit handles, for example, undefined attributes, the way MSG files are processed if the codepage is invalid or not supported, or how much memory to use for container attachments.

Running the query for item type information

Run this query to obtain parameter values related to an item type that are required when running various indexing commands.

About this task

To obtain information about the index name, the item type name, the table name, the index schema, the index directory, or the event and log table names:

Run the following query:

```
select
  k.keywordname as item_type,
  substr(t.tablename,1,25) as tablename,
  c.columnname,
  c.indexname,
  substr(t.eventviewname,1,13) as event_table,
  substr(tt.logviewname,1,13) as log_table,
  substr(t.indexdirectory,1,80) as indexdirectory,
  substr(t.workdirectory,1,80) as workdirectory,
  substr(t.format,1,10) as format,
  t.modelfile,
  c.udfname as CM_UDFNAME,
  substr(t.functionschema,1,20) as NSEUDFSchema,
  t.functionname as NSE_UDFName,
  t.number_docs as numdocs,
  t.updatefrequency as UPDFREQ
from
  icmsttextindexconf c,
  db2ext.textcolumns t,
  db2ext.ttextindexes tt,
  icmstnlskeywords k,
  icmstcompdefs d
where
  c.indexname=t.indexname
  and c.indexname=tt.indexname
  and c.componenttypeid=d.componenttypeid
```

```
  and keywordclass=18
  and keywordcode=d.itemtypeid
  and k.languagecode='ENU'
  order by k.keywordname;
```

The output of the query delivers the following information:

```
ITEM_TYPE
TABLENAME
INDEXNAME
FORMAT
CM_UDFNAME
NSEUDFSCHEMA
NSE_UDFNAME
NUMDOCS UPDFREQ
```

Verifying the fast indexer installation

After you have configured Content Manager and DB2, and before you can begin indexing the first time, you must verify that the fast indexer is called correctly and that the index definitions and configuration settings are correct.

About this task

Under DB2 using the user ID icmadmin or another user ID that has database administrator authority:

1. Enable the item type for indexing under Related tasks below.
2. Verify that the UDF name used to retrieve the documents to be indexed has been changed to ICMFetchFile by running:

   ```
   db2 "select functionname from db2ext.ttextcolumns
   where aliasname='<table name>'"
   ```

   If the output begins with ICMFETCHFILE, the fast indexer is loaded and processed correctly.
3. To verify that the item type is called correctly using the fast indexer, run:

   ```
   db2 "select itemid, icmfetchfilter(tieref) from <tablename>
   where tieref is not null fetch first row only"
   ```

   If the output is an XML document with sections appropriate for an e-mail, the fast indexer has been called correctly.

Related tasks

"Enabling an item type for indexing"
Before you can begin indexing, you must enable the item type for use with the fast indexer.

Enabling an item type for indexing

Before you can begin indexing, you must enable the item type for use with the fast indexer.

About this task

You only need to enable the item type once the very first time you run indexing on this item type.

Under DB2 using the user ID icmadmin or another user ID that has database administrator authority:

1. Run the query for information related to the new item type under Related tasks below. This query returns the item type name, the schema name, the database name, and the index name required in subsequent commands.
2. Run the following command to scan and index all entries in the component table and create triggers before running the fast indexer:

   ```
   db2text update index <schema name>.<index name> for text connect
   to <database name>
   ```

   If you do not run `db2text update` once for each item type , the entire table will be processed without using the log table and an error is returned for every document. The only way of recovering if this situation arises, is to recreate the index.
3. Run the following script to prepare for fast indexing. `fetchcontentsetup` lists several DB2 configuration parameters and their recommended values.
   - For AIX or Solaris, run:

     ```
     fetchcontentsetup.sh <item-type name> RUN [childtriggers]
     ```
   - For Windows, run:

     ```
     fetchcontentsetup.bat <item-type name> RUN [childtriggers]
     ```

   The argument `childtriggers` is used to create triggers on the DB2 database so that the full-text index is informed about changes that have occurred on the child component of the corresponding item type. In an archive enabled for single-instance storing (SIS), the index must be updated automatically whenever a new mailbox ID is added to a mail document (in the case where a user archives an e-mail, a new SIS child component is created) or removed from the corresponding document (in the case where a user deletes an archived document instance, a SIS child component is removed). The DB2 trigger ensures through automatic preprocessing that the full-text index is kept up-to-date.

   Important:
   - If you are enabling an item type that has been created using IBM Content Collector you MUST provide the `childtriggers` argument.
   - If you are enabling an item type that has been created manually and the item type uses the single-instance storing (SIS) feature defined for your archive you MUST provide the `childtriggers` argument.
   - If you are enabling an item type that has been created manually but the item type does NOT use the single-instance storing (SIS) feature defined for your archive, do NOT provide the `childtriggers` argument.

   For debugging purposes, you can run `fetchcontentsetup` with `LOG` before running the command with `RUN`. `RUN` applies the DB2 commands listed in `LOG`.

   The `fetchcontentsetup` script must be run in the same directory where the emsutil.properties file is, usually the fastidx directory.

Related tasks

"Running the query for item type information" on page 6
Run this query to obtain parameter values related to an item type that are required when running various indexing commands.

Indexing documents

If the item type has been enabled for indexing, run indexing on this item type to enable text-search.

Before you begin

The item type must only be enabled for use with the fast indexer once before indexing is run the first time.

Before you start indexing, make sure that enough disk space is allocated for the temporary XML files. To get an estimate of the XML file size, update the emsutil.properties file and set **debug_udf=performance**. Check the file size reported in icmserver.fetchfile, or the output object size reported in icmserver.fetchfilter1. Alternatively, you could simply monitor the total size and number of files in the directory containing the temporary files while `indexdocuments` is running.

About this task

To index documents for text-search:

Run the following script to retrieve, convert, and index the documents in the item type.

- For AIX or Solaris, run:

  ```
  indexdocuments.sh <item-type name> <maxruntime> <interval>
  ```

- For Windows, run:

  ```
  indexdocuments.bat <item-type name> <maxruntime> <interval>
  ```

`indexdocuments` must be run in the same directory where the emsutil.properties file is, usually the fastidx directory. On Windows, the `indexdocuments` script must be run from a DB2 command window and not a normal Command Prompt window. `<maxruntime>` sets the number of *hours* that `indexdocuments` runs and `<interval>` the time interval in *minutes* between two consecutive indexing runs.
Setting both values to zero is only recommended for testing purposes. If `<maxruntime>` is zero, only one iteration is run. If `<interval>` is zero, processing continues with no wait time until the Content Manager log table is empty. After that, the indexing process ends.
It is advisable that you run indexing for the same set period each day (for example, 18 hours) to allow indexing to end so that required backups or other activities can be performed with no indexing activity running on the system. The interval should be set to a time that would allow enough documents to arrive to start the batch.
Only the number of documents specified by the value of the **batchsize** parameter in the emsutil.properties file is processed. Indexing starts as soon as the number of retrieved, converted, and saved documents equals the value of **batchsize**.
On Windows, if you are using Windows Scheduler, you can create a bat file, for example runindexing.bat, containing the following two lines to run indexing:

```
cd %emsutil_path%
db2cmd db2setcp indexdocuments <itemtype> 18 30
```

This would make the indexing process run once a day, with a total run time of less than 24 hours.

Results

The processing output is recorded in the <item-type name>.log file, which is located in the directory where `indexdocuments` ran, usually the fastidx directory. Use the tail command to monitor this file. (The tail program for Windows is available from a variety of sources.)

Related tasks

"Running the query for item type information" on page 6
Run this query to obtain parameter values related to an item type that are required when running various indexing commands.

Reindexing documents with errors

To reindex documents that have failed due to an error, rows must be inserted into the log table for each error in the event table, then the log table must be updated, and finally the errors must be removed from the event table.

About this task

Note: If you are reindexing documents that produced conversion errors (indicated by TIEFLAG > 10000) the TIEFLAG value should be reset to mod(TIEFLAG,10000).

To reindex documents that have failed due to a particular error, first insert rows into the log table for each error in the event table that matches the given error string, update the log table, and finally remove the errors from the event table.

1. Run the following commands:

```
db2 "insert into icmut0####cminxlog (select 1, current timestamp,
pk01 from db2ext.teventix###### where msg like '%some string%'
and time between 'yyyy-mm-dd-hh.mm.ss' and 'yyyy-mm-dd-hh.mm.ss')"

db2 "update icmut0####001 set tieflag=mod(tieflag,10000) where
compkey in (select pk01 from db2ext.teventix###### where msg like
'%some string%' and time between 'yyyy-mm-dd-hh.mm.ss' and
'yyyy-mm-dd-hh.mm.ss')"

db2 "delete from db2ext.teventix###### where message like '%some string%'
and time between 'yyyy-mm-dd-hh.mm.ss' and 'yyyy-mm-dd-hh.mm.ss'"
```

   The 4–digit number (ICMUT0####...) is the Content Manager ComponentTypeID and is obtained using the following query:

```
db2 "select componenttypeid from icmstcompviewdefs where
componentviewname = 'CSLDMail001'"
```

   The 6–digit number (teventix######) is part of the index identifier and is obtained using the following query:

```
db2 "select indexidentifier from db2ext.ttextindexes where
indexname = 'ICMUU0####001TIE'"
```

   The string that is searched for must exist in the indexing event table exactly as it is used in the query. To find valid values, list rows from the indexing event table. Refer to the topic below about how to monitor indexing events.

2. After updating the CMINXLOG table, run `indexdocuments`.

Example

Another example could be to reindex documents created at a particular time, for example, documents created in January 2008. In this case, enter the following:

```
db2 "insert into icmut0####cminxlog (select 1, current timestamp,
compkey from icmut0####001)"

db2 "insert into icmut0####cminxlog (select 1, current timestamp,
compkey from icmut0####001
where createts between '2008-01-01-00.00.00' and '2008-01-31-00.00.00')"
```

Related reference

"Monitoring indexing events" on page 13
You can monitor the indexing events produced by the indexer (NetSearch Extender) during a particular time period.

Recreating an index

In most cases, you only need to prepare an existing index which is created when the item type is defined. However, if the index configuration (cteixcfg.ini) is not correct, or if the index is corrupted and cannot be restored from a backup, then the index must be recreated.

About this task

Important: Recreating an index is a very expensive procedure especially if a large amount of documents needs to be reindexed, Do not recreate an index unless there is absolute proof that the index is definitely corrupted.

To recreate the index:

- On AIX or Solaris, run

  ```
  recreateindex.sh <item-type name> <indexpath> RUN
  ```

- On Windows, run

  ```
  recreateindex.bat <item-type name> <indexpath> RUN
  ```

where `<indexpath>` is the directory where the index files are to be stored.

Note:

- `recreateindex` expects the emsutil.properties file to be in the same directory in which it runs. Ensure that emsutil.properties is available if you run `recreateindex` from a location other than the fastidx directory.
- The command shell environment must be configured with the proper path to the Java JRE binaries and the native DB2 SQL libraries. For example, on AIX, this can be done by setting (the paths may need to be adapted to your local environment):

  ```
  export PATH=/usr/java5_64/bin/:$PATH:.
  export LD_LIBRARY_PATH=/opt/IBM/db2/V9.5/lib64:$LD_LIBRARY_PATH
  ```

Related tasks

"Running the query for item type information" on page 6
Run this query to obtain parameter values related to an item type that are required when running various indexing commands.

Monitoring indexing

During indexing, all the encountered errors are written to a file named <item-type name>.log which is located in the same directory as the indexdocuments script.

About this task

This file contains a line for each 1000 documents that have been retrieved and converted. The number of documents listed in the file can be configured using the parameter **display.count** in the emsutil.properties file.

This file shows the number of documents that are available for processing and the number of documents in the indexing log table ready for updating. It also documents when indexing was started and if indexing was interrupted and what steps to take in this case.

For example, if a previous update was started and the system was rebooted or indexing was interrupted, a lock will be left which prevents the next update from starting. The log file suggests checking if the process cteprupx is still running, and if not, to clear the locks.

To verify if indexing is still running:

- For AIX or Solaris, enter:

  ```
  ps -eaf | grep cteprupx
  ```

- For Windows, check if the `cteprupx` process is running using the Task Manager

Additional monitoring information not directly related to indexing is logged in icmserver.fetchfilter1 by setting the parameter **debug_udf=performance** in the emsutil.properties file.

If **debug_udf=performance** is set, one line is written to icmserver.fetchfilter1 for each document retrieved and converted. For example:

```
2008/07/30 15:06:27.437
-- pid     7400
-- ItemID: A1001001A08A30B42525C80749
  Input size:    1092116
  Output size:     67630
  Pre-retrieve:      1
  Retrieve from RM:     58
  Post-retrieve:    164
  INSO:      0 (msec)

2008/08/05 06:03:37.375
1152 bytes from c:\temp\items\63\item.A1001001A08A30B42520A63295      1 msec
```

where `Input size` is the original e-mail size, `Output size` is the size of XML file returned from the text-search user-exit. `Post-retrieve` is the time in the text-search user-exit, including INSO document conversion. In addition, one line is written to ICMFetchFile for each indexed document.

Use the `NSEIndexingStats` script to monitor the performance of the index updates. `NSEIndexingStats` reports each update and each index reorganization that was done. Because the time to merge temporary files into the secondary index increases as the secondary index grows, it is important to monitor the speed of each update and determine when reorganization should be done.

For a very high volume system with large indexes, increasing **batchsize** to 150K or even more should be considered. Be sure that there is adequate disk or operating system cache to keep the entire batch in memory, because physical I/O might defeat the gains from larger batches. The file icmserver.fetchfile shows the elapsed time for reading each document. This should be zero or one msec if cache is being used. Also verify that the elapsed time between each recorded document is very low. If it is more than a few msec for most files, it is likely that the I/O performance creating temporary files is causing a problem.

To improve indexing performance:

- Place the index files and temporary XML files created during conversion on the fastest possible DASD.

- Minimize the number of index updates by increasing **batchsize**. Set **batchsize** to at least 10% of the daily e-mail volume.

For a high level view of the indexing progress, use the command:

```
db2text control list all locks for database <database name>
```

While the count is increasing, files are being read from disk and indexed. When the count stops increasing, the temporary files are being merged into the secondary index, or the secondary index is being merged into the main index.

With a large index, this could take hours. If you are concerned that the indexing process has stopped due to an error, monitor the file system where the index files are stored. If after several minutes, the file sizes have not changed, then indexing has probably encountered a problem. Before stopping any processes (particularly `cteprupx`) or running `db2text stop`, contact DB2 support to determine what information is needed to diagnose the situation.

Related tasks

"Running the query for item type information" on page 6
Run this query to obtain parameter values related to an item type that are required when running various indexing commands.

Monitoring index reorganization

You should monitor indexing performance using `nseindexingstats` to track how long reorganizing the index takes.

On a very large index, if reorganizing takes 3 hours and you want to be sure that indexing with reorganization is only allowed to start between, for example, 10:00 pm and 3:00 am, you must change the values of the reorg properties in the emsutil.properties file. To reflect the conditions in the example, set the following property values:

```
reorg.time=20.00
reorg.last.time=03.00
```

Monitoring indexing events

You can monitor the indexing events produced by the indexer (NetSearch Extender) during a particular time period.

To display the indexing events, connect to the database and enter the following command:

- For Windows:

  ```
  NSEEvents.bat <itemtype> x
  ```

- For AIX and Solaris:

  ```
  . NSEEvents.sh <itemtype> x
  ```

  Note: The dot (.) is required on UNIX® machines to pass the DB2 environment into the command.

Backing up the index files

Index files can be corrupted during the final merging or reorganization stages in the indexing process if either the `db2text` or `cteprupx` processes are stopped. The index can also be corrupted if files cannot be written to because there is not enough file system space. It is therefore important to design a strategy to restore a point-in-time backup and then to update this backup with documents that were indexed prior to the backup.

About this task

As each document is retrieved, converted, and written to the file system, a record is removed from the Content Manager log table and inserted into both the indexing log table (CMINXLOG) and the backup table (CMINXBKP):

- Copy the backup table together with the index log table to another location.
- Clear the backup table, or delete records dating from before the last backup. For example, to delete all records older than 30 days for the index log table icmut01031 (the log table name for your item type is obtained by running the query in Related topics below), enter:

  ```
  db2 "delete from icmut01031cminxlog where time < current timestamp – 30 days"
  ```

- Back up the index files after each `db2text update` process has finished running. To make regular copies of all the index files you can write your own backup script or alternatively use a regularly scheduled backup program that runs when indexing is not running. Which ever method you use, test your backup process carefully to make sure that the process works correctly.

  This is an example of what an indexing backup script for Windows could contain. Bear in mind that this is only a sample to use as a basis for writing any script code and must certainly be adapted to your environment:

  ```
  @rem  This is a sample script to back up NSE index files on Windows
  @rem  using IndexDocumentsExit from the fast indexing process.

  @rem Save files only if a reorg was done on the "db2text update" command
  @if " %2" == " 0" @goto :exit

  @setlocal
  @set BackupDirectory=e:\NSEBACKUPS

  @echo ====================== >> %emsutil_path%\IndexDocumentsExit.log
  @time /T >> %emsutil_path%\IndexDocumentsExit.log
  @echo Copy index files for %1 >> %emsutil_path%\IndexDocumentsExit.log

  @db2 connect to icmnlsdb
  @db2 set current schema icmadmin

  @db2 -x select '@SET INDEXID=' ^
             concat cast(t.indexidentifier as char(8)) ^
             from db2ext.ttextindexes t, db2ext.ttextcolumns c ^
             where t.indexidentifier = c.indexidentifier ^
             and c.tablename='%1' > ~tmp.bat

  @db2 -x select '@SET COPY1=' ^
             concat rtrim(indexdirectory) ^
             concat '\NODE0000\' ^
             concat rtrim(t.indexidentifier) ^
             concat '\*' ^
             from db2ext.ttextindexes t, db2ext.ttextcolumns c ^
             where t.indexidentifier = c.indexidentifier ^
             and c.tablename='%1' >> ~tmp.bat

  @db2 -x select '@SET COPY2=' ^
  ```

```
            concat rtrim(indexdirectory) ^
            concat '\NODE0000\' ^
            concat rtrim(t.indexidentifier) ^
            concat '.*' ^
            from db2ext.ttextindexes t, db2ext.ttextcolumns c ^
            where t.indexidentifier = c.indexidentifier ^
            and c.tablename='%1' >> ~tmp.bat


@call ~tmp.bat

xcopy %COPY1% %BackupDirectory%\%INDEXID%\* /sy  >> %emsutil_path%\IndexDocumentsExit.log
copy  %COPY2% %BackupDirectory%\%INDEXID% /y     >> %emsutil_path%\IndexDocumentsExit.log

@echo Backup complete
time /T >> %emsutil_path%\IndexDocumentsExit.log

:exit
```

If the backup must be restored, the files can be copied back again and the rows reinserted into the Content Manager log table. To copy the rows:

- Enter the following command (with sample cminxlog and cminxbkp tables):

  ```
  db2 "insert into icmut01031cminxlog select * from icmut01031cminxbkp
   where time ">" '2006-01-01-03.00.00'"
  ```

  The example assumes that the last good backup was on January 2, 2006, so any rows added to the CMINXBKP table after January 1 at 3:00 a.m. will be copied to the CMINXLOG table so that these documents will be reindexed.
- Run `indexdocuments` to re-index these documents.

Related tasks

"Running the query for item type information" on page 6
Run this query to obtain parameter values related to an item type that are required when running various indexing commands.