Introduction to SQL Server 2005 Full Text Searching

Creating and Coding Against the Full Text Search Engine

About

- Robert C. Cain
- Southern Nuclear since 2005
- 10 years as a consultant in the B’ham Market
- Wide range of .Net applications, ASP & Win
- SQL Server 2005 Data Warehouse
- http://arcanecode.com

Thank You

“The successful people do what the failures won’t.”

--*Anthony Robbins*

What is Full Text Searching?

- A way to rapidly search unstructured data
- Find key words buried in free format text fields

What bits do you need?

- Optionally installed component on Standard and above. You must explicitly install.
- Not included with SQL Server Express.
- Instead, download SQL Server Express Edition with Advanced Services SP2 from http://msdn2.microsoft.com/en-us/express/bb410792.aspx (or see my blogpost from 10/4/07)

Basic steps for implementing FTS

1. Create the Catalog
2. Create the index for each table
3. Query against it

Creating the Catalog

```
create fulltext catalog my_catalog_name_here
in path ‘c:\mysqldata\somesubdirectory’
as default
```

- In line 1, enter the name of your catalog for “my_catalog_name_here”
- ‘in path’ is optional, if omitted it will simply place it where the database resides
- Adding ‘as default’ makes this the default catalog, meaning you won’t have to specify a catalog when you do FTS searches.

Demo 1

- Creating a Full Text Search Catalog

Modifications to Existing Catalogs

- `alter fulltext catalog my_catalog_name_here rebuild`
- `alter fulltext catalog my_catalog_name_here reorganize`
- `alter fulltext catalog my_catalog_name_here as default`

Modifications to Existing Catalogs

- `alter fulltext catalog my_catalog_name_here rebuild`

  Pros: Fast, efficient
  Cons: Takes the catalog offline, unavailable for full text search queries

Modifications to Existing Catalogs

- `alter fulltext catalog my_catalog_name_here reorganize`

  Pros: Rebuilds the indexes without taking the catalog offline. Still available for users to query.
  Cons: Slower, takes longer to update all the indexes.

Modifications to Existing Catalogs

- `alter fulltext catalog my_catalog_name_here as default`

  Makes this catalog the default

Catalogs – Best Practices

- Single Catalog
- Multiple Catalogs

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

Use the name of your table for my_table_name_here

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
   (column1, column2,…)
   key index my_tables_unique_index_name
   on my_catalog_name_here
   with change_tracking {manual | auto | off},
   no population
```

- Enter the name(s) of the fields to be indexed.
- Must be a valid data type.

Valid Data Types

- Char, nchar, varchar, nvarchar, text, ntext, xml, varbinary(max), and image are the valid data types for FTS.
- However, according to the documentation text, ntext, and image are going to be removed in a future version of SQL Server, so I’m going to avoid them and so should you.

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

- The Key Index must be a unique index for that row.
- You cannot create a Full Text Search index without first having a unique, primary key index.

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

- Use the name of your catalog for my_catalog_name_here
- Optional, if omitted the default catalog is used.

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

Change Tracking indicates how changes to the source table will be reflected in the Full Text Search index.

Change Tracking Options

- Manual – Changes are tracked, but are not sent to the FTS index until an alter command is done (more on that in a moment)
- Auto – The FTS index is automatically updated whenever the source table is updated.
- Off – Changes are not tracked. Only way to get it in sync is with a rebuild.

Create the Full Text Search Index

```
create fulltext index on my_table_name_here
  (column1, column2,…)
  key index my_tables_unique_index_name
  on my_catalog_name_here
  with change_tracking {manual | auto | off},
  no population
```

- Only valid when change_tracking is set to OFF
- Used to create a FTS index but not populate it

Create the Full Text Search Index

- Only one Full Text Search Index is allowed for any table.

- Demo 2

Best Practices with `change_tracking`

- In high use database, may wish to create the index using off, no population.
- Schedule a job to update during off peak hours (more on that in a moment)
- Alter the index to auto or manual afterward

Best Practices with `change_tracking`

- Use auto for databases with low updates
- Use manual for high updates, combined with a scheduled job to reorganize or rebuild

Altering an Existing Index

alter fulltext index on my_table_name_here
parameters here

- set change_tracking {off | auto | manual} – This works the same as with the create command, it lets you change the tracking mode.
- disable – Disables the full text search index, it's not used for searching nor is it updated. However the data is left intact, should you want to turn it back on.
- enable – Enables the full text search index after a disable.

Altering an Existing Index

alter fulltext index on my_table_name_here

parameters here

- add (column) – Adds the passed in column to the full text search index.
- drop (column) – Removes the passed in column from the full text search index.

Altering an Existing Index

alter fulltext index on my_table_name_here *parameters here*

- start full population –This rebuilds the index from the ground up.
- start incremental population –This will update the index since the last time it was updated. Note you must have a timestamp column on your table for this to work.
- start update population – Use with change_tracking manual option to update the index

Altering an Existing Index

alter fulltext index on my_table_name_here *parameters here*

- drop fulltext index on my_table_name_here

Querying the Full Text Search Index

Four new keywords

- Contains
- ContainsTable
- FreeText
- FreeTextTable

Contains

- Looks for an exact match

```
select col1, col2 from myTable
where contains(fts_column, ‘searchword’)
```

- Demo 3

FreeText

- Searches for variations of the word

```
select col1, col2 from myTable
where freetext(fts_column, ‘searchword’)
```

Demo 4

Advanced Options for Queries

- FORMS OF
- Phrases – wrap in double quotes
- NEAR
- OR
- Prefixed Terms (term plus wildcard - *)

Demo 5

Searching with Ranking

- FreeTextTable, ContainsTable
- Work identically except ContainsTable is specific match (just like Contains)
- Acts as a table that returns two fields: KEY and RANK
- KEY Is the primary key from the source table
- Rank is a number from 1 to 1000. The lower the number the better the match.

Searching with Ranking

- Demo 6 – Query with Ranking

Using inside an Application

- Demo 7 – Used inside an Application

Helpful System Queries

```
-- List names of all catalogs attached to the database
select * from sys.fulltext_catalogs

-- List names of all FTS indexes
select cat.[name] as CatalogName
     , object_name(object_id) as table_name
     , is_enabled
     , change_tracking_state_desc
  from sys.fulltext_indexes, sys.fulltext_catalogs cat

-- List table name and column number (ordinal position) for
   indexes
select object_name(object_id) table_name, column_id
  from sys.fulltext_index_columns
```

- Demo 8 – Helpful System Queries

Cleaning up the mess

```
drop fulltext index on [Production].[ProductDescription];
drop fulltext catalog AdvWorksCatalog;
```

- Demo 9 - Cleanup

Noise Words

- Finally, you should be aware that certain words are excluded from searches. Common words such as a, an, the, and so on. Microsoft refers to these as “Noise Words”. You can edit the list of noise words in case you have some words in your environment that wind up being Noise Words. Your company name might be one example.
- I found the file in the folder C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\FTData (your milage may vary). The file is named noiseenu.txt. (ENU is for English US, not to be confused with noiseeng.txt which our friends in the British Isles will be using.)
- This is also handy to know in case there is a reserved word you need to remove from the list. In our environment one of the reserve words is also an abbreviation for a piece of our equipment, so I would want to remove this from our list.

Thanks Again!

- Questions?
- All material available at http://arcanecode.com
- arcanecode@gmail.com