عنوان: **نصب و راه اندازی مقدماتی Full Text Search**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۵ ۲۰:۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server

با استفاده از امکانات ابتدایی T-SQL مانند like می‌توان جستجوهایی را برای یافتن موارد مشابه با عبارتی خاص انجام داد، اما این جستجوها بسیار هزینه‌بر و کند هستند. در SQL Server برای مدیریت جستجوهای سریع و پیشرفته بر روی متون، افزونه‌های توکاری مانند Full text search، Semantic search، Term extraction و Term lookup تدارک دیده شده‌اند. Semantic search از نگارش 2012 آن افزوده شده‌است و مابقی در نگارش‌های پیشین آن نیز وجود داشته‌اند.

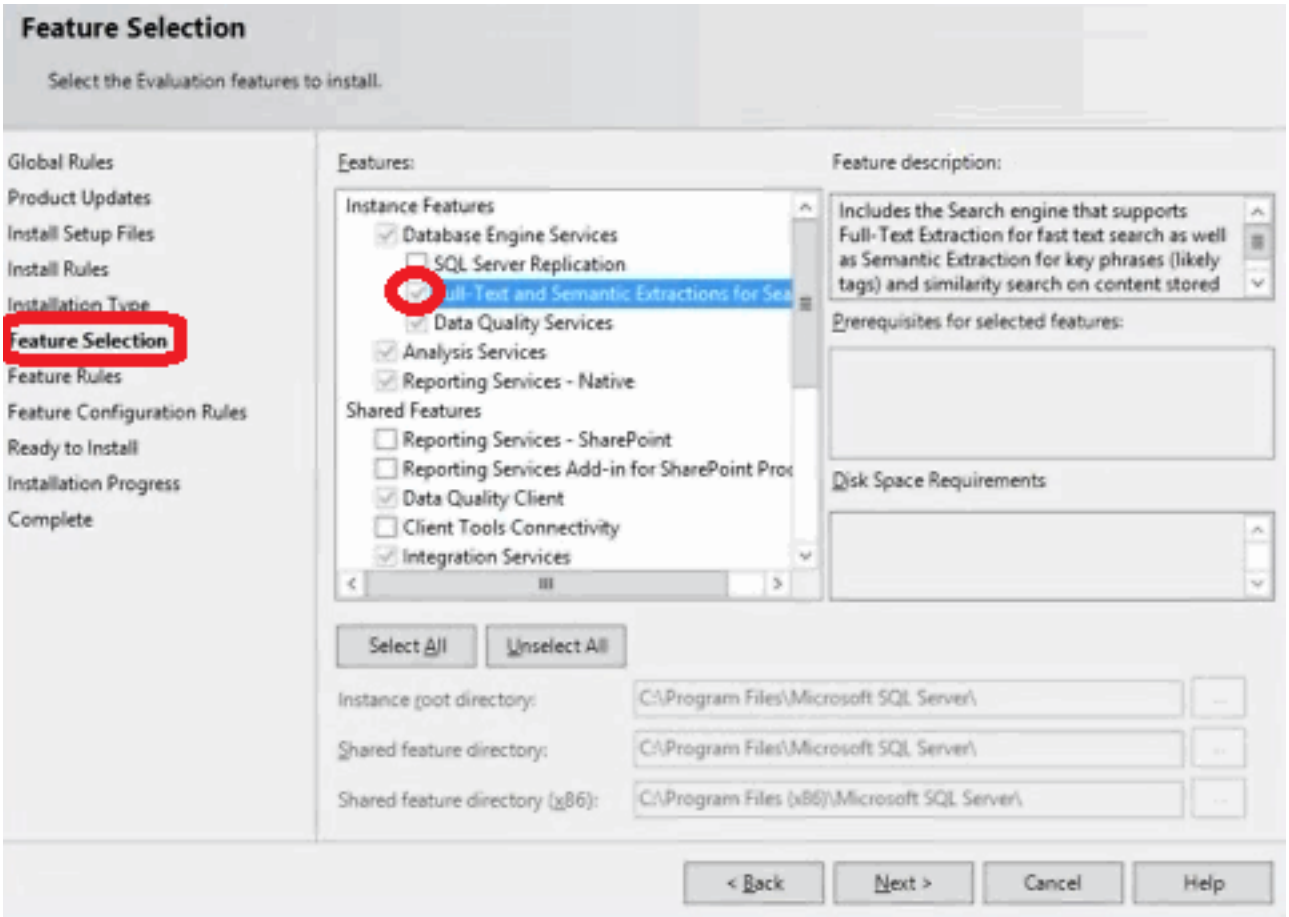
بررسی‌های مقدماتی

ابتدای کار نیاز است بررسی کنیم آیا افزونه‌ی Full Text Search، به همراه SQL Server نصب شده‌است یا خیر. برای این منظور کوئری ذیل را اجرا کنید:

```
select SERVERPROPERTY('IsFullTextInstalled');
```

اگر خروجی این کوئری عدد 1 بود، یعنی FTS نصب شده‌است؛ اگر خیر، مجددا برنامه‌ی نصاب SQL Server را اجرا کرده و زمانیکه به قسمت feature selection رسیدید، گزینه‌ی ذیل را باید انتخاب کنید:

```
instance features -> database engine services -> Full Text
```



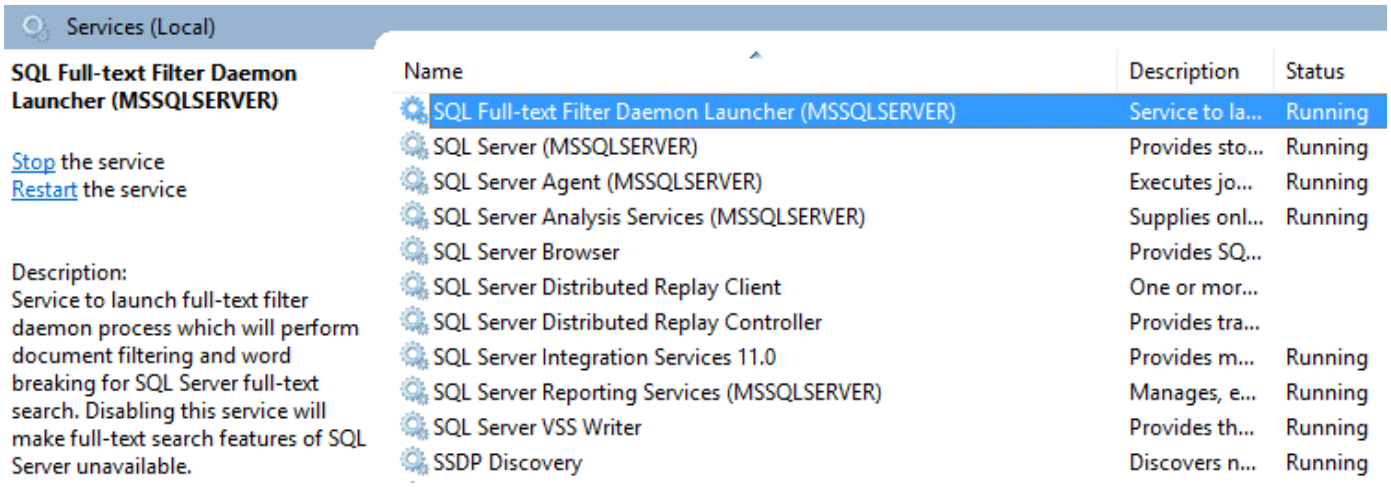
راه اندازی سرویس Full Text Search

پیش از ادامه‌ی بحث، به کنسول سرویس‌های ویندوز مراجعه کرده و مطمئن شوید که سرویس SQL Full-text Filter Daemon Launcher MSSQLSERVER نیز در حال اجرا است. در غیراینصورت با خطای ذیل مواجه خواهید شد:

```
SQL Server encountered error 0x80070422 while communicating with full-text filter daemon host (FDHost) process.
```

اگر این سرویس در حال اجرا است و باز هم خطای فوق ظاهر شد، مجددا به کنسول سرویس‌های ویندوز مراجعه کرد، در برگه‌ی خواص سرویس SQL Full-text Filter Daemon Launcher MSSQLSERVER، گزینه‌ی logon را یافته و آن‌را به local system account تغییر دهید. سپس سرویس را ری استارت کنید. پس از آن نیاز است دستور ذیل را نیز اجرا کنید:

```
sp_fulltext_service 'restart_all_fdhosts'
```



چه نوع داده‌هایی را می‌توان توسط FTS ایندکس کرد؟

با استفاده از امکانات FTS می‌توان کلیه ستون‌هایی را که دارای نوع‌های ذیل باشند، ایندکس کرد:

```
char, nchar, varchar, nvarchar, text, ntext, image, xml, varbinary(max)
```

البته نوع باینری را که ملاحظه می‌کنید مانند image و varbinary max، نیاز به یک ستون اضافی، برای ذخیره سازی پسوند فایل‌های ذخیره شده در آن‌ها مانند xlsx ، pdf، docx و امثال آن نیز دارند. برای مثال ابتدا یک فایل word را در ستونی از نوع varbinary max ذخیره می‌کنید و سپس نیاز است در همانجا در ستونی دیگر، پسوند این فایل را نیز قید نمائید.
همچنین FTS برای پردازش این فایل‌های باینری و ایندکس کردن اطلاعات آن‌ها، نیاز به افزونه‌هایی به نام IFilters دارد. کار این فیلترها استخراج متن بدون فرمت، از فایل‌های باینری مرتبط و ارائه‌ی آن‌ها به موتور FTS می‌باشد.

نصب فیلترهای مخصوص FTS آفیس

اگر علاقمند هستید که بدانید در حال حاضر چه تعداد فیلترهای FTS بر روی سیستم شما نصب شده‌است، کوئری ذیل را اجرا

نمائید:

```
exec sys.sp_help_fulltext_system_components 'filter';
```

برای نمونه اگر آفیس بر روی سیستم شما نصب باشد، در حاصل کوئری فوق، فیلتری مانند offfilt.dll را نیز مشاهده خواهید کرد که به پسوندهایی مانند xls، ppt، doc و امثال آن انتساب داده شده‌است.
فیلترهای آفیس را جداگانه نیز می‌توانید دریافت و نصب کنید (بدون نیاز به نصب کامل آفیس بر روی سرور):
Microsoft Office 2010 Filter Packs
این فیلترها تا نگارش 2013 آفیس را نیز پشتیبانی می‌کنند و اگر آپدیت ویندوز نیز روشن باشد، سرویس پک 2 آن را نیز دریافت خواهید کرد.

پس از اینکه فیلترها را نصب کردید، باید آن‌ها را در وهله‌ی جاری SQL Server ثبت کرد:

```
exec sys.sp_fulltext_service 'load_os_resources', 1;
EXEC sp_fulltext_service 'update_languages';
EXEC sp_fulltext_service 'restart_all_fdhosts';
```

اکنون اگر مجددا کوئری sys.sp_help_fulltext_system_components یاد شده را اجرا کنید. خروجی آن حدودا 50 سطر خواهد بود؛ این اطلاعات را از کوئری ذیل نیز می‌توان بدست آورد:

```
select * from sys.fulltext_document_types;
```

اگر پس از نصب و همچنین ثبت و معرفی فیلترهای آفیس 2010 به بعد، هنوز تعداد 50 ردیف را ملاحظه می‌کنید (اکنون باید بیشتر از 160 مورد باشند)، نیاز است یکبار وهله‌ی جاری SQL Server را ری استارت کنید. برای اینکار در management studio بر روی وهله‌ی جاری، کلیک راست کرده و گزینه‌ی Restart را انتخاب کنید.

فیلترهای فوق علاوه بر اینکه امکان FTS را بر روی کلیه فایل‌های مجموعه آفیس میسر می‌کنند، امکان جستجو FTS را بر روی خواص ویژه اضافی آن‌ها، مانند نام نویسنده، واژه‌های کلیدی، تاریخ ایجاد و امثال آن نیز به همراه دارند.

FTS چگونه کار می‌کند؟

زبان‌های پشتیبانی شده توسط FTS را توسط کوئری ذیل می‌توانید مشاهده کنید:

```
select lcid, name from sys.fulltext_languages order by name;
```

کار FTS با word-breakers و stemmers شروع می‌شود. این‌ها کار آنالیز متن را بر اساس زبانی مشخص انجام می‌دهند. اگر زبان مدنظر توسط FTS پشتیبانی نمی‌شود، می‌توان از زبان انگلیسی و یا همچنین Neutral نیز برای آنالیز آن استفاده کرد. زبان Neutral جزو خروجی کوئری فوق با شماره آی دی صفر است.
word-breakers تک تک کلمات را (که به آن‌ها token نیز گفته می‌شود) تشخیص داده و سپس FTS آن‌ها را با فرمتی فشرده شده، درون ایندکس‌های مخصوص خود ذخیره می‌کند.کار stemmers تولید حالات inflectional (صرفی) یک کلمه بر اساس دستور زبانی مشخص است.
اهمیت آنالیز inflectional، در اینجا است که برای مثال اگر در متنی واژه‌ی jumps وجود داشت و کاربر در حین جستجو، jumped را وارد کرد، FTS بر اساس دستور زبان مورد استفاده، پیشتر، حالات مختلف صرفی jump را ذخیره کرده‌است و امکان انجام یک چنین کوئری پیشرفته‌ای را پیدا می‌کند.

نصب و فعال سازی Semantic Language Database

کار TFS تنها به خرد کردن واژه‌ها و آنالیز صرفی آن‌ها خلاصه نمی‌شود. در مرحله‌ی بعد، انجام Statistical semantic search میسر می‌شود. در اینجا SQL Server بر اساس آمار واژه‌های کلیدی استخراج شده، توانایی یافتن متونی مشابه و یا مرتبط را پیدا می‌کند. Semantic Search جزو تازه‌های SQL Server 2012 است.

برای اینکار نیاز است بانک اطلاعاتی Semantic language statistics نیز نصب شود. برای اطمینان از نصب بودن آن، کوئری ذیل را اجرا کنید:

```
select * from sys.fulltext_semantic_language_statistics_database;
```

اگر حاصل آن خالی بود، نیاز است مستقلا نصب شود. این بانک اطلاعاتی ویژه را در یکی از دو مسیر ذیل

```
x64\Setup\SemanticLanguageDatabase.msi
x86\Setup\SemanticLanguageDatabase.msi
```

در DVD یا فایل ISO نصب SQL Server 2012 می‌توانید پیدا کنید. فایل نصاب msi آن‌را اجرا کنید، دو فایل mdf و ldf را در مسیری که مشخص می‌کنید، کپی می‌کند.
پس از آن نیاز است این بانک اطلاعاتی را Attach و همچنین ثبت کرد:

```
CREATE DATABASE semanticsdb
    ON ( FILENAME = 'D:\SQL_Data\SemanticLanguageDatabase\semanticsdb.mdf' )
    LOG ON ( FILENAME = 'D:\SQL_Data\SemanticLanguageDatabase\semanticsdb_log.ldf' )
    FOR ATTACH
GO

EXEC sp_fulltext_semantic_register_language_statistics_db @dbname = N'semanticsdb'
GO
```

زمانیکه این بانک اطلاعاتی کپی می‌شود، دسترسی Write کاربر وارد شده به سیستم را در برگه‌ی Security فایل‌های mdf و ldf آن ندارد. به همین جهت ممکن است در حین Attach، پیام عدم دسترسی را دریافت کنید که با مراجعه به خواص فایل‌ها و تنظیم دسترسی Write کاربر جاری، مشکل برطرف می‌شود.
پس از مراحل فوق، اگر مجددا کوئری یاد شده بر روی sys.fulltext_semantic_language_statistics_database را اجرا کنید، یک سطر خروجی خواهد داشت.

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۱۲/۱۶ ۰:۴۹

در مقایسه با لوسیون کدام قوی تره؟ و یا پیشنهاد میشه؟
و بنظرتون معیارهای انتخاب برای استفاده از هرکدوم چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۶ ۱:۱۴

از لوسین برای بانک‌های اطلاعاتی سبکی که قابلیت‌های Full text search ندارند، بهتر است استفاده شود. برای مثال اگر از SQLite استفاده می‌کنید یا حتی SQL Server CE (سبک‌ترین نسخه‌ی SQL Server که یک بانک اطلاعاتی embedded محسوب می‌شود)، لوسین بسیار مناسب است.
برای نمونه در سایت جاری از آن برای تهیه موتور جستجوی سایت استفاده شده و یا حتی برنامه‌ی سبک Viewer بانک اطلاعاتی سایت که با فرمت XML است، از لوسین استفاده می‌کند.
به صورت خلاصه برای کارهای سبک و یا بانک‌های اطلاعاتی embedded، استفاده از لوسین فوق العاده است.
اما برای کار با SQL Server کامل، واقعا نیازی به لوسین نیست. یک زیرساخت کامل و توکار برای Full Text Search دارد که با زبان T-SQL آن یکپارچه است. نگهداری و به روز رسانی آن توسط برنامه نویس در حد صفر است و یکبار که تعریف شد، به خوبی کار می‌کند. نگهداری ایندکس‌های لوسین خودکار نیست و باید توسط برنامه نویس به صورت مجزا هربار که اطلاعات تغییر می‌کند انجام شود.

جستجو بر روی خواص و متادیتای اسناد آفیس

همانطور که در قسمت قبل نیز عنوان شد، فیلترهای FTS آفیس، علاوه بر اینکه امکان جستجوی پیشرفته FTS را بر روی کلیه فایل‌های مجموعه آفیس میسر می‌کنند، امکان جستجوی FTS را بر روی خواص ویژه اضافی آن‌ها، مانند نام نویسنده، واژه‌های کلیدی، تاریخ ایجاد و امثال آن نیز به همراه دارند.
اینکه چه خاصیتی را بتوان جستجو کرد نیز بستگی به نوع فیلتر نصب شده دارد. برای تعریف خواص قابل جستجوی یک سند، باید یک SEARCH PROPERTY LIST را ایجاد کرد:

```
CREATE SEARCH PROPERTY LIST WordSearchPropertyList;
GO

ALTER SEARCH PROPERTY LIST WordSearchPropertyList
   ADD 'Authors'
   WITH (PROPERTY_SET_GUID = 'F29F85E0-4FF9-1068-AB91-08002B27B3D9',
             PROPERTY_INT_ID = 4,
             PROPERTY_DESCRIPTION = 'System.Authors - authors of a given item.');
GO
```

در این تعریف، PROPERTY_INT_ID و PROPERTY_SET_GUIDها استاندارد بوده و لیست آن‌ها را در آدرس ذیل می‌توانید مشاهده نمائید:

Find Property Set GUIDs and Property Integer IDs for Search Properties

بهبود کیفیت جستجو توسط Stop lists و Stop words

به یک سری از کلمات و حروف، اصطلاحا noise words گفته می‌شود. برای مثال در زبان انگلیسی حروف و کلماتی مانند a، is، the و and به صورت خودکار از FTS حذف می‌شوند؛ چون جستجوی آن‌ها بی‌حاصل است. به این‌ها stop words نیز می‌گویند.
با استفاده از کوئری ذیل می‌توان لیست stop words تعریف شده در بانک اطلاعاتی جاری را مشاهده کرد:

```
-- Check the Stopwords list
SELECT w.stoplist_id,
   l.name,
   w.stopword,
   w.language
FROM sys.fulltext_stopwords AS w
   INNER JOIN sys.fulltext_stoplists AS l
     ON w.stoplist_id = l.stoplist_id;
```

و برای تعریف stop words از دستورات ذیل کمک گرفته می‌شود:

```
-- Stopwords list
CREATE FULLTEXT STOPLIST SQLStopList;
GO

-- Add a stopword
ALTER FULLTEXT STOPLIST SQLStopList
 ADD 'SQL' LANGUAGE 'English';
GO
```

کاتالوگ‌های Full Text Search

ایندکس‌های ویژه‌ی FTS، در مکان‌هایی به نام Full Text Catalogs ذخیره می‌شوند. این کاتالوگ‌ها صرفا یک شیء مجازی بوده و تنها برای تعریف ظرفی دربرگیرنده‌ی ایندکس‌های FTS تعریف می‌شوند. در نگارش‌های پیش از 2012 اس کیوال سرور، این کاتالوگ‌ها اشیایی فیزیکی بودند؛ اما اکنون تبدیل به اشیایی مجازی شده‌اند.
حالت کلی تعریف یک fulltext catalog به نحو ذیل است:

```
create fulltext catalog catalog_name
on filegroup filegroup_name
in path  'rootpath'
with some_options
as default
authoriztion owner_name
accent_sensivity = {on|off}
```

اما اکثر گزینه‌های آن مانند on filegroup و in path صرفا برای حفظ سازگاری با نگارش‌های قبلی حضور دارند و دیگر نیازی به ذکر آن‌ها نیست؛ چون تعریف کننده‌ی ماهیت فیزیکی این کاتالوگ‌ها می‌باشند.
به صورت پیش فرض حساسیت به لهجه یا accent_sensivity خاموش است. اگر روشن شود، باید کل ایندکس مجددا بازسازی شود.

ایجاد ایندکس‌های Full Text

پس از ایجاد یک fulltext catalog، اکنون نوبت به تعریف ایندکس‌هایی فیزیکی هستند که داخل این کاتالوگ‌ها ذخیره خواهند شد:

```
-- Full-text catalog
CREATE FULLTEXT CATALOG DocumentsFtCatalog;
GO

-- Full-text index
CREATE FULLTEXT INDEX ON dbo.Documents
(
  docexcerpt Language 1033,
  doccontent TYPE COLUMN doctype
  Language 1033
  STATISTICAL_SEMANTICS
)
KEY INDEX PK_Documents
ON DocumentsFtCatalog
WITH STOPLIST = SQLStopList,
  SEARCH PROPERTY LIST = WordSearchPropertyList,
  CHANGE_TRACKING AUTO;
GO
```

در اینجا توسط KEY INDEX نام منحصربفرد ایندکس مشخص می‌شود.
CHANGE_TRACKING AUTO به این معنا است که SQL Server به صورت خودکار کار به روز رسانی این ایندکس را با تغییرات رکوردها انجام خواهد داد.
ذکر STATISTICAL_SEMANTICS، منحصر به SQL Server 2012 بوده و کار آن تشخیص واژه‌های کلیدی و ایجاد ایندکس‌های یافتن اسناد مشابه است. برای استفاده از آن حتما نیاز است مطابق توضیحات قسمت قبل، Semantic Language Database پیشتر نصب شده باشد.
توسط STOPLIST، لیست واژه‌هایی که قرار نیست ایندکس شوند را معرفی خواهیم کرد. SQLStopList را در ابتدای بحث ایجاد کردیم.
1033 Language به معنای استفاده از زبان US English است.
نحوه‌ی استفاده از SEARCH PROPERTY LIST ایی که پیشتر تعریف کردیم را نیز در اینجا ملاحظه می‌کنید.

مثالی برای ایجاد ایندکس‌های FTS

برای اینکه ربط منطقی نکات عنوان شده را بهتر بتوانید بررسی و آزمایش کنید، مثال ذیل را درنظر بگیرید.

ابتدا جدول Documents را برای ذخیره سازی تعدادی سند، ایجاد می‌کنیم:

```
CREATE TABLE dbo.Documents
(
  id INT IDENTITY(1,1) NOT NULL,
  title NVARCHAR(100) NOT NULL,
  doctype NCHAR(4) NOT NULL,
  docexcerpt NVARCHAR(1000) NOT NULL,
  doccontent VARBINARY(MAX) NOT NULL,
  CONSTRAINT PK_Documents
PRIMARY KEY CLUSTERED(id)
);
```

اگر به این جدول دقت کنید، هدف از آن ذخیره‌ی اسناد آفیس است که فیلترهای FTS آن‌را در قسمت قبل نصب کردیم. ستون doctype، معرف نوع سند و doccontent ذخیره کننده‌ی محتوای کامل سند خواهند بود.

سپس اطلاعاتی را در این جدول ثبت می‌کنیم:

```
-- Insert data
-- First row
INSERT INTO dbo.Documents
(title, doctype, docexcerpt, doccontent)
SELECT N'Columnstore Indices and Batch Processing',
 N'docx',
 N'You should use a columnstore index on your fact tables,
   putting all columns of a fact table in a columnstore index.
   In addition to fact tables, very large dimensions could benefit
   from columnstore indices as well.
   Do not use columnstore indices for small dimensions. ',
 bulkcolumn
FROM OPENROWSET
 (BULK 'C:\Users\Vahid\Desktop\Updates\fts_docs\ColumnstoreIndicesAndBatchProcessing.docx',
  SINGLE_BLOB) AS doc;

-- Second row
INSERT INTO dbo.Documents
(title, doctype, docexcerpt, doccontent)
SELECT N'Introduction to Data Mining',
 N'docx',
 N'Using Data Mining is becoming more a necessity for every company
   and not an advantage of some rare companies anymore. ',
 bulkcolumn
FROM OPENROWSET
 (BULK 'C:\Users\Vahid\Desktop\Updates\fts_docs\IntroductionToDataMining.docx',
  SINGLE_BLOB) AS doc;

-- Third row
INSERT INTO dbo.Documents
(title, doctype, docexcerpt, doccontent)
SELECT N'Why Is Bleeding Edge a Different Conference',
 N'docx',
 N'During high level presentations attendees encounter many questions.
   For the third year, we are continuing with the breakfast Q&A session.
   It is very popular, and for two years now,
   we could not accommodate enough time for all questions and discussions! ',
 bulkcolumn
FROM OPENROWSET
 (BULK 'C:\Users\Vahid\Desktop\Updates\fts_docs\WhyIsBleedingEdgeADifferentConference.docx',
  SINGLE_BLOB) AS doc;

-- Fourth row
INSERT INTO dbo.Documents
(title, doctype, docexcerpt, doccontent)
SELECT N'Additivity of Measures',
 N'docx',
 N'Additivity of measures is not exactly a data warehouse design problem.
   However, you have to realize which aggregate functions you will use
   in reports for which measure, and which aggregate functions
   you will use when aggregating over which dimension.',
 bulkcolumn
FROM OPENROWSET
 (BULK 'C:\Users\Vahid\Desktop\Updates\fts_docs\AdditivityOfMeasures.docx',
  SINGLE_BLOB) AS doc;
GO
```

4 ردیف ثبت شده در جدول اسناد، نیاز به 4 فایل docx نیز دارند که آن‌ها را از آدرس ذیل می‌توانید برای تکمیل ساده‌تر آزمایش دریافت کنید:

fts_docs.zip

در ادامه می‌خواهیم قادر باشیم تا بر روی متادیتای نویسنده‌ی این اسناد نیز جستجوی کامل FTS را انجام دهیم. به همین جهت SEARCH PROPERTY LIST آن‌را نیز ایجاد خواهیم کرد:

```
-- Search property list
CREATE SEARCH PROPERTY LIST WordSearchPropertyList;
GO
ALTER SEARCH PROPERTY LIST WordSearchPropertyList
 ADD 'Authors'
 WITH (PROPERTY_SET_GUID = 'F29F85E0-4FF9-1068-AB91-08002B27B3D9',
PROPERTY_INT_ID = 4,
PROPERTY_DESCRIPTION = 'System.Authors - authors of a given item.');
GO
```

همچنین می‌خواهیم از واژه‌ی SQL در این اسناد، در حین ساخت ایندکس‌های FTS صرفنظر شود. برای این منظور یک FULLTEXT STOPLIST را به نام SQLStopList ایجاد کرده و سپس واژه‌ی مدنظر را به آن اضافه می‌کنیم:

```
-- Stopwords list
CREATE FULLTEXT STOPLIST SQLStopList;
GO
-- Add a stopword
ALTER FULLTEXT STOPLIST SQLStopList
 ADD 'SQL' LANGUAGE 'English';
GO
```

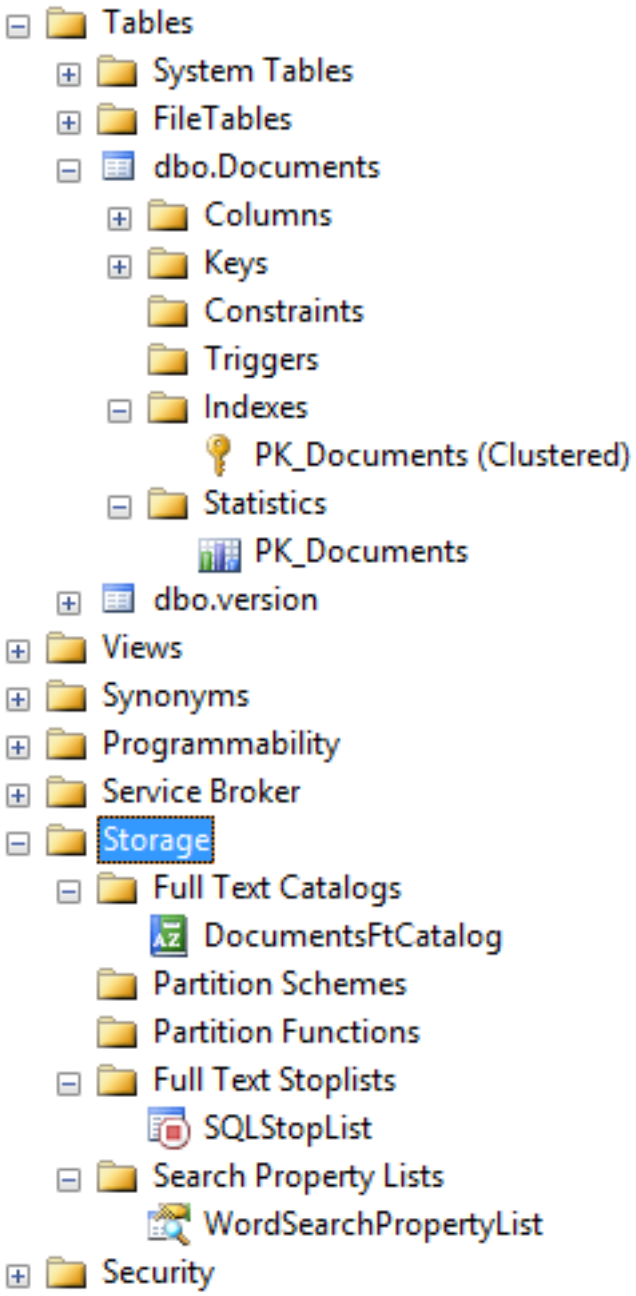
صحت عملیات آن‌را توسط کوئری «Check the Stopwords list» ذکر شده در ابتدای بحث می‌توانید بررسی کنید.

اکنون زمان ایجاد یک کاتالوگ FTS است:

```
-- Full-text catalog
CREATE FULLTEXT CATALOG DocumentsFtCatalog;
GO
```

با توجه به اینکه در نگارش‌های جدید SQL Server این کاتالوگ صرفا ماهیتی مجازی دارد، ساده‌ترین Syntax آن برای کار ما کفایت می‌کند.
و در آخر ایندکس FTS ایی را که پیشتر در مورد آن بحث کردیم، ایجاد خواهیم کرد:

```
-- Full-text index
CREATE FULLTEXT INDEX ON dbo.Documents
(
  docexcerpt Language 1033,
  doccontent TYPE COLUMN doctype
  Language 1033
  STATISTICAL_SEMANTICS
)
KEY INDEX PK_Documents
ON DocumentsFtCatalog
WITH STOPLIST = SQLStopList,
  SEARCH PROPERTY LIST = WordSearchPropertyList,
  CHANGE_TRACKING AUTO;
GO
```



در این تصویر محل یافتن اجزای مختلف Full text search را در management studio مشاهده می‌کنید.

یک نکته‌ی تکمیلی

برای زبان فارسی نیز یک سری stop words وجود دارند. لیست آن‌ها را از اینجا می‌توانید دریافت کنید:

stopwords.sql

متاسفانه زبان فارسی جزو زبان‌های پشتیبانی شده توسط FTS در SQL Server نیست (نه به این معنا که نمی‌توان با آن کار کرد؛ به این معنا که برای مثال دستورات صرفی زبان را ندارد) و به همین جهت از زبان انگلیسی در اینجا استفاده شده‌است.

عنوان: تهیه کوئری بر روی ایندکس‌های Full Text Search
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۶ ۱۸:۱۹
آدرس: www.dotnettips.info
گروه‌ها: SQL Server

در دو قسمت قبل ابتدا سیستم FTS را نصب و فعال کردیم و سپس تعدادی رکورد را ثبت کرده، کاتالوگ‌های FTS، ایندکس‌ها و Stop words متناظری را ایجاد کردیم. در این قسمت قصد داریم از این اطلاعات ویژه، استفاده کرده و کوئری بگیریم. مواردی که بررسی خواهند شد اصطلاحا Predicates نام داشته و شامل توابع مخصوصی مانند Contains و Freetext می‌شوند.

با استفاده از Contains predicate چه اطلاعاتی را می‌توان جستجو کرد؟

متد Contains مخصوص FTS، قابلیت یافتن کلمات و عبارات، تطابق کامل با عبارت در حال جستجو و یا حتی جستجوهای فازی را دارد. همچنین حالات مختلف صرفی یا inflectional یک کلمه را نیز می‌تواند جستجو کند (مانند jumps ،jump و jumped). البته این مورد وابسته است به زبانی که در حین ایجاد ایندکس مشخص می‌شود. امکان یافتن کلماتی نزدیک و مشابه به کلماتی دیگر نیز پیش بینی شده‌است. پیشوندها و پسوندها را نیز می‌توان جستجو کرد. امکان تعیین وزن و اهمیت کلمات در حال جستجو وجود دارند (برای مثال در این جستجوی خاص، کلمه‌ی ویژه اهمیت بیشتری نسبت به بقیه دارد). متد Contains امکان جستجوی Synonyms را نیز دارد. برای مثال یافتن رکوردهایی که معنایی مشابه need دارند اما دقیقا حاوی کلمه‌ی need نیستند.

بررسی ریز جزئیات توانمندی‌های Contains predicate

1) جستجوی کلمات ساده

```
-- Simple term
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'data');
```

در این کوئری که بر روی جدول Documents قسمت قبل انجام می‌شود، به دنبال عین واژه‌ی در حال جستجو هستیم.
باید دقت داشت که این نوع کوئری‌ها، حساس به حروف کوچک و بزرگ نیستند.
همچنین عبارت وارد شده از نوع یونیکد است. به همین جهت برای جلوگیری از تغییر encoding رشته وارد شده (و تفسیر آن بر اساس Collation بانک اطلاعاتی)، یک N به ابتدای عبارت افزوده شده‌است.

2) جستجوی عبارات

```
-- Simple term - phrase
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'"data warehouse"');
```

اگر نیاز به یافتن عین عبارتی که از چند کلمه تشکیل شده‌است می‌باشد، نیاز است آن‌را با "" محصور کرد.

3) استفاده از عملگرهای منطقی مانند OR و AND

```
-- Simple terms with logical OR
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'data OR index');
```

در این کوئری نحوه‌ی استفاده از عملگر منطقی OR را مشاهده می‌کنید.
و یا نحوه‌ی بکارگیری AND NOT در کوئری ذیل مشخص شده‌است:

```
-- Simple terms with logical AND NOT
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'data AND NOT mining');
```

در این کوئری به دنبال رکوردهایی هستیم که docexcerpt آن‌ها دارای کلمه‌ی data بوده، اما شامل mining نمی‌شوند.
به علاوه با استفاده از پرانتزها می‌توان تقدم و تاخر عملگرهای منطقی را بهتر مشخص کرد:

```
-- Simple terms with mny logical operators, order defined with parentheses
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'data OR (fact AND warehouse)');
```

4) جستجوی پیشوندها

```
-- Prefix
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'"add*"');
```

در کوئری فوق به دنبال رکوردهایی هستیم که docexcerpt آن‌ها با کلمه‌ی add شروع می‌شوند. در این حالت نیز استفاده از "" اجباری است. اگر از "" استفاده نشود، FTS به دنبال تطابق عینی با عبارت وارد شده خواهد گشت.

5) جستجوهای Proximity

Proximity در اینجا به معنای یافتن واژه‌هایی هستند که نزدیک (از لحاظ تعداد فاصله بر حسب کلمات) به واژه‌ای دیگر می‌باشند.

```
-- Simple proximity
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'NEAR(problem, data)');
```

برای این منظور از واژه‌ی NEAR استفاده می‌شود؛ به همراه ذکر دو واژه‌ای که به دنبال آن‌ها هستیم. معنای کوئری فوق این است: رکوردهایی را پیدا کن که در آن در یک جایی از خلاصه سند، کلمه‌ی problem وجود دارد و در جایی دیگر از آن خلاصه‌ی سند، کلمه‌ی data.
همچنین می‌توان مشخص کرد که این نزدیک بودن دقیقا به چه معنایی است:

```
-- Proximity with max distance 5 words
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'NEAR((problem, data),5)');

-- Proximity with max distance 1 word
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'NEAR((problem, data),1)');
```

در این کوئری‌ها اعداد 1 و 5، بیانگر فاصله‌ی بین دو کلمه‌ای هستند (فاصله بر اساس تعداد کلمه) که قرار است در نتایج جستجو حضور داشته باشند. مقدار پیش فرض آن Max است؛ یعنی در هر جایی از سند.
همچنین می‌توان مشخص کرد که ترتیب جستجو باید دقیقا بر اساس نحوه‌ی تعریف این کلمات در کوئری باشد:

```
-- Proximity with max distance and order
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'NEAR((problem, data),5, TRUE)');
GO
```

پارامتر آخر یا flag، به صورت پیش فرض false است. به این معنا که ترتیب این دو کلمه در جستجو اهمیتی ندارند.

6) جستجوی بر روی بیش از یک فیلد

در قسمت قبل، FULLTEXT INDEX انتهای بحث را بر روی دو فیلد docexcerpt و doccontent تهیه کردیم. اگر نیاز باشد تا جستجوی انجام شده هر دو فیلد را شامل شود می‌توان به نحو ذیل عمل کرد:

```
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS((docexcerpt,doccontent), N'data');
```

در این حالت تنها کافی است دو فیلد را داخل یک پرانتز قرار داد.

یک نکته: اگر تعداد ستون‌های ایندکس شده زیاد است و نیاز داریم تا بر روی تمام آن‌ها FTS انجام شود، تنها کافی است پارامتر اول متد Contains را * وارد کنیم. * در اینجا به معنای تمام ستون‌هایی است که در حین تشکیل FULLTEXT INDEX ذکر شده‌اند.

7) جستجوهای صرفی یا inflectional

FTS بر اساس زبان انتخابی، در حین تشکیل ایندکس‌های خاص خودش، یک سری آنالیزهای دستوری را نیز بر روی واژه‌ها انجام می‌دهد. همچنین امکان تعریف زبان مورد استفاده در حین استفاده از متد Contains نیز وجود دارد.

```
-- Inflectional forms

-- The next query does not return any rows
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'presentation');

-- The next query returns a row
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'FORMSOF(INFLECTIONAL, presentation)');
GO
```

در این مثال در کوئری اول به دنبال عین واژه‌ی وارد شده هستیم که با توجه به تنظیمات قسمت قبل و داده‌های موجود، خروجی را به همراه ندارد.
اکنون اگر کوئری دوم را که از FORMSOF جهت تعیین روش INFLECTIONAL استفاده کرده است، اجرا کنیم، به یک رکورد خواهیم رسید که در آن جمع واژه‌ی presentation وجود دارد.

8) جستجو برای یافتن متشابهات

برای نمونه اگر SQL Server 2012 بر روی سیستم شما نصب باشد، محل نصب واژه‌نامه‌های Synonyms یا واژه‌هایی همانند از لحاظ معنایی را در مسیر زیر می‌توانید مشاهده کنید:

```
C:\...\MSSQL11.MSSQLSERVER\MSSQL\FTData
```

این‌ها یک سری فایل XML هستند با ساختار ذیل:

```
<XML ID="Microsoft Search Thesaurus">
    <thesaurus xmlns="x-schema:tsSchema.xml">
<diacritics_sensitive>0</diacritics_sensitive>
        <expansion>
            <sub>Internet Explorer</sub>
            <sub>IE</sub>
            <sub>IE5</sub>
        </expansion>
        <replacement>
            <pat>NT5</pat>
            <pat>W2K</pat>
```

```
            <sub>Windows 2000</sub>
        </replacement>
        <expansion>
            <sub>run</sub>
            <sub>jog</sub>
        </expansion>
        <expansion>
            <sub>need</sub>
            <sub>necessity</sub>
        </expansion>
    </thesaurus>
</XML>
```

در اینجا diacritics_sensitive به معنای حساسیت به لهجه است که به صورت پیش فرض برای تمام زبان‌ها خاموش است.
سپس یک سری expansion و replacement را مشاهده می‌کنید.
فایل tsenu.xml به صورت پیش فرض برای زبان انگلیسی آمریکایی مورد استفاده قرار می‌گیرد. اگر محتویات آن‌را برای مثال با محتویات XML ایی فوق جایگزین کنید (در حین ذخیره باید دقت داشت که encoding فایل نیاز است Unicode باشد)، سپس باید SQL Server را از این تغییر نیز مطلع نمائیم:

```
-- Load the US English file
EXEC sys.sp_fulltext_load_thesaurus_file 1033;
GO
```

عدد 1033، عدد استاندارد زبان US EN است.
البته اگر اینکار را انجام ندهیم، به صورت خودکار، اولین کوئری که از THESAURUS انگلیسی استفاده می‌کند، سبب بارگذاری آن خواهد شد.

```
-- Synonyms

-- The next query does not return any rows
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'need');

-- The next query returns a row
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(docexcerpt, N'FORMSOF(THESAURUS, need)');
GO
```

در اولین مثال به دنبال عین واژه‌ی need در رکوردهای موجود هستیم که خروجی را بر نمی‌گرداند.
در ادامه اگر کوئری دوم را که از FORMSOF جهت تعیین روش THESAURUS استفاده کرده است، اجرا کنیم، به یک رکورد خواهیم رسید که در آن واژه‌ی necessity به کمک محتویات فایل tsenu.xml که پیشتر تهیه کردیم، بجای need وجود دارد.

9) جستجو بر روی خواص و متادیتای فایل‌ها

```
-- Document properties
SELECT id, title, docexcerpt
FROM dbo.Documents
WHERE CONTAINS(PROPERTY(doccontent,N'Authors'), N'Test');
```

در اینجا نحوه‌ی جستجوی خواص فایل‌های docx ذخیره شده در قسمت قبل را مشاهده می‌کنید که شامل ذکر PROPERTY و ستون FTS مورد نظر است، به همراه نام خاصیت و عبارت جستجو.

کار با FREETEXT

```
-- FREETEXT
SELECT *
FROM dbo.Documents
WHERE FREETEXT(docexcerpt, N'data presentation need');
```

FREETEXT عموما ردیف‌های بیشتری را نسبت به Contains بر می‌گرداند؛ چون جستجوی عمومی‌تری را انجام می‌دهد. در اینجا جستجو بر روی معنای عبارات انجام می‌شود و نه صرفا یافتن عباراتی دقیقا همانند عبارت در حال جستجو. در اینجا مباحث Synonyms و Inflectional ایی که پیشتر یاد شد، به صورت خودکار اعمال می‌شوند.
در کوئری فوق، کلیه رکوردهایی که با سه کلمه‌ی وارد شده (به صورت مجزا) به نحوی تطابق داشته باشند (تطابق کامل یا بر اساس تطابق‌های معنایی یا دستوری) باز گردانده خواهند شد.

نظرات خوانندگان

نویسنده: میلاد رسولی ا
تاریخ: ۱۳۹۴/۰۱/۰۸ ۲۱:۱۷

بنده در حال ساخت جستجویی برای وب سایتی هستم. این جستجو بر روی جداول کتاب، نویسنده، مترجم و انتشارات انجام میشه و در صورتی که کاربر قسمتی از نام کتاب و نام نویسنده را وارد کند جستجو بر روی این دو فیلد که از دو جدول متفاوت هستند انجام می‌شود.
مشکل اینجاست که از آنجایی که دستوارت FTS بر روی یک جدول عمل می‌کنند و با توجه به پیچیدگی جستجو، شما چه راهی را برای کوئری گرفتن از چندین جدول (که ممکن است یک کتاب چند نویسنده هم داشته باشد) پیشنهاد می‌کنید.
بنده در حال حاضر تمام این جداول را در یک View قرار داده و فیلدهای چندمقداری را با Concat بوسیله " ، " در یک فیلد جای داده‌ام.
ممنون از راهنماییتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۰۸ ۲۲:۱۶

دو متد CONTAINSTABLE و FREETEXTTABLE امکان join را نیز میسر می‌کنند.

```
SELECT b.Name, a.Name, bkt.[Rank] + akt.[Rank]/2 AS [Rank]
FROM Book b
INNER JOIN Author a ON b.AuthorID = a.AuthorID
INNER JOIN FREETEXTTABLE(Book, Name, @criteria) bkt ON b.ContentID = bkt.[Key]
LEFT JOIN FREETEXTTABLE(Author, Name, @criteria) akt ON a.AuthorID = akt.[Key]
ORDER BY [Rank] DESC
```

عنوان: بررسی Semantic Search و FTS Table-valued functions
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۷ ۱:۳۶
آدرس: www.dotnettips.info
گروه‌ها: SQL Server

Semantic Search جزو تازه‌های SQL Server 2012 است و مقدمات نصب و فعال سازی آن‌را در قسمت اول بررسی کردیم.
توابع Predicates مختص به FTS مانند Contains و Freetext، تنها ردیف‌های متناظر با جستجوی انجام شده را باز می‌گردانند و رتبه‌ای به نتایج جستجو اعمال نمی‌گردد. برای مثال، مشخص نیست اولین ردیف بازگشت داده شده بهترین تطابق را با جستجوی انجام شده دارد یا بدترین نتیجه‌ی ممکن است. برای رفع این مشکل FTS table-valued functions معرفی شده‌اند. حاصل این‌ها یک جدول با دو ستون است. ستون اول کلید متناظر با جدول تطابق یافته بوده و ستون دوم، Rank نام دارد که بیانگر میزان مفید بودن و درجه‌ی اعتبار ردیف بازگشت داده شده‌است.
Semantic Search نیز به کمک سه table-valued functions پیاده سازی می‌شود. همچنین باید دقت داشت که تمام زبان‌های پشتیبانی شده توسط FTS در حالت Semantic Search پشتیبانی نمی‌شوند. برای بررسی این مورد، دو کوئری ذیل را اجرا نمائید:

```
-- Full text Languages
SELECT *
FROM sys.fulltext_languages
ORDER BY name;

-- Semantic Search Languages
SELECT *
FROM sys.fulltext_semantic_languages
ORDER BY name;
GO
```

بررسی table-valued functions مختص به FTS

دو متد ویژه‌ی CONTAINSTABLE و FREETEXTTABLE خروجی از نوع جدول دارند؛ با ستون‌هایی به نام‌های key و rank. اگر قسمت ایجاد کاتالوگ FTS و ایندکس آن‌را بخاطر داشته باشید، در حین ایجاد ایندکس FTS می‌بایستی KEY INDEX PK_Documents را نیز ذکر کرد. کاربرد آن در همین table-valued functions است.
مقدار rank، عددی است بین 0 و 1000 که هر چقدر مقدار آن بیشتر باشد، یعنی نتیجه‌ای نزدیک‌تر، به عبارت جستجو شده، یافت گردیده‌است. باید دقت داشت که این عدد فقط در زمینه‌ی یک کوئری معنا پیدا می‌کند و مقایسه‌ی rank دو کوئری مختلف با هم، بی‌معنا است.
عملکرد CONTAINSTABLE بسیار شبیه به متد Contains است با این تفاوت که قابلیت‌های بیشتری دارد. برای مثال در اینجا می‌توان برای قسمتی از جستجو، وزن و اهمیت بیشتری را قائل شد و این حالت تنها زمانی معنا پیدا می‌کند که خروجی جستجو، دارای rank باشد.
متد FREETEXTTABLE نیز بسیار شبیه به FREETEXT عمل کرده و نسبت به CONTAINSTABLE بسیار ساده‌تر است. برای نمونه امکان تعریف وزن، near، formsof و غیره در اینجا وجود ندارد. به علاوه عملگرهای منطقی مانند and و or نیز در اینجا کاربردی نداشته و صرفا یک noise word درنظر گرفته می‌شوند.

چند مثال جهت بررسی عملکرد دو متد CONTAINSTABLE و FREETEXTTABLE

استفاده از متد CONTAINSTABLE

```
-- Rank with CONTAINSTABLE
SELECT D.id, D.title, CT.[RANK], D.docexcerpt
FROM CONTAINSTABLE(dbo.Documents, docexcerpt,
N'data OR level') AS CT
 INNER JOIN dbo.Documents AS D
  ON CT.[KEY] = D.id
ORDER BY CT.[RANK] DESC;
```

چون متد CONTAINSTABLE خروجی از نوع table دارد، در قسمت from ذکر شده‌است.

این متد ابتدا نام جدول مورد بررسی را دریافت می‌کند. سپس ستونی که باید جستجو بر روی آن انجام شود و در ادامه عبارت جستجو شونده، مشخص می‌گردد. اگر این متد را به تنهایی اجرا کنیم:

```
SELECT * FROM CONTAINSTABLE(dbo.Documents, docexcerpt, N'data OR level')
```

همانطور که عنوان شد، صرفا یک سری ردیف اشاره کننده به id و rank را بازگشت می‌دهد. به همین جهت join نوشته شده‌است تا بتوان رکوردهای اصلی را نیز در همینجا به همراه rank متناظر، نمایش داد.

استفاده از متد FREETEXTTABLE

```
-- Rank with FREETEXTTABLE
SELECT D.id, D.title, FT.[RANK], D.docexcerpt
FROM FREETEXTTABLE (dbo.Documents, docexcerpt,
N'data level') AS FT
 INNER JOIN dbo.Documents AS D
  ON FT.[KEY] = D.id
ORDER BY FT.[RANK] DESC;
```

کلیات عملکرد متد FREETEXTTABLE بسیار شبیه است به متد CONTAINSTABLE؛ با این تفاوت که ساده‌تر بوده و بسیاری از قابلیت‌های پیشرفته و سفارشی CONTAINSTABLE را به صورت خودکار و یکجا اعمال می‌کند. به همین جهت دقت آن، اندکی کمتر بوده و عمومی‌تر عمل می‌کند.
در اینجا اگر نیاز باشد تا تعداد نتایج را شبیه به کوئری‌های top n محدود نمود، می‌توان از پارامتر عددی بعدی که برای نمونه به 2 تنظیم شده‌است، استفاده کرد:

```
-- Rank with FREETEXTTABLE and top_n_by_rank
SELECT D.id, D.title, FT.[RANK], D.docexcerpt
FROM FREETEXTTABLE (dbo.Documents, docexcerpt,
N'data level', 2) AS FT
 INNER JOIN dbo.Documents AS D
  ON FT.[KEY] = D.id
ORDER BY FT.[RANK] DESC;
```

در این کوئری تنها 2 ردیف بازگشت داده می‌شود.

تعیین وزن و اهمیت کلمات در حال جستجو

```
-- Weighted terms
SELECT D.id, D.title, CT.[RANK], D.docexcerpt
FROM CONTAINSTABLE
(dbo.Documents, docexcerpt,
N'ISABOUT(data weight(0.8), level weight(0.2))') AS CT
 INNER JOIN dbo.Documents AS D
  ON CT.[KEY] = D.id
ORDER BY CT.[RANK] DESC;
```

با استفاده از واژه کلیدی ISABOUT، امکان تعیین وزن، برای واژه‌های در حال جستجو ممکن می‌شوند. در این کوئری اهمیت واژه data بیشتر از اهمیت واژه level تعیین شده‌است.

انجام جستجوهای Proximity

```
-- Proximity term
SELECT D.id, D.title, CT.[RANK]
FROM CONTAINSTABLE (dbo.Documents, doccontent,
N'NEAR((data, row), 30)') AS CT
 INNER JOIN dbo.Documents AS D
  ON CT.[KEY] = D.id
```

```
ORDER BY CT.[RANK] DESC;
GO
```

در اینجا مانند متد CONTAINS، امکان انجام جستجوهای Proximity نیز وجود دارد. برای مثال در کوئری فوق به دنبال رکوردهایی هستیم که در آن‌ها واژه‌های data و row وجود دارند، با فاصله‌ای کمتر از 30 کلمه.

بررسی Semantic Search key valued functions

متد SEMANTICKEYPHRASETABLE کار بازگشت واژه‌های کلیدی آنالیز شده توسط FTS را انجام داده و جدولی حاوی 4 ستون را باز می‌گرداند. این چهار ستون عبارتند از:
- column_id: شماره ستون واژه کلیدی یافت شده‌است. تفسیر آن نیاز به استفاده از تابع سیستمی COL_NAME دارد (مانند مثال زیر).
- document_key: متناظر است با کلید اصلی جدولی که بر روی آن کوئری گرفته می‌شود.
- keyphrase: همان واژه کلیدی است.
- score: رتبه‌ی واژه کلیدی است در بین سایر واژه‌هایی که بازگشت داده شده و عددی است بین صفر تا یک.

مثالی از آن‌را در ادامه ملاحظه می‌کنید:

```
-- Top 100 semantic key phrases
SELECT TOP (100)
 D.id, D.title,
 SKT.column_id,
 COL_NAME(OBJECT_ID(N'dbo.Documents'), SKT.column_id) AS column_name,
 SKT.document_key,
 SKT.keyphrase, SKT.score
FROM SEMANTICKEYPHRASETABLE
(dbo.Documents, doccontent) AS SKT
 INNER JOIN dbo.Documents AS D
  ON SKT.document_key = D.id
ORDER BY SKT.score DESC;
```

Results | Messages

	id	title	column_id	column_name	document_key	keyphrase	score
1	4	Additivity of Measures	5	doccontent	4	additive	0.7003461
2	2	Introduction to Data Mining	5	doccontent	2	undirected	0.6858258
3	4	Additivity of Measures	5	doccontent	4	additivity	0.6471558
4	4	Additivity of Measures	5	doccontent	4	aggregate	0.6166306
5	2	Introduction to Data Mining	5	doccontent	2	clustering	0.6092916
6	3	Why Is Bleeding Edge a Different Conference	5	doccontent	3	presentations	0.5764243

در متد جدولی SEMANTICKEYPHRASETABLE، ابتدا جدول مورد نظر و سپس ستونی که نیاز است واژه‌های کلیدی آنالیز شده‌ی آن بازگشت داده شوند، قید می‌گردند. document_key آن به تنهایی شاید مفید نباشد. به همین جهت join شده‌است به جدول اصلی، تا بتوان رکوردهای متناظر را نیز بهتر تشخیص داد.
به این ترتیب مهم‌ترین واژه‌های کلیدی ستون doccontent را به همراه درجه‌ی اهمیت و رتبه‌ی آن‌ها، می‌توان گزارش گرفت.

متد SEMANTICSIMILARITYTABLE برای یافتن سندهای مشابه با یک سند مشخص بکار می‌روند؛ چیزی شبیه به گزارش «مقالات مشابه مطلب جاری» در بسیاری از سایت‌های ارائه‌ی محتوا. ستون‌های خروجی آن عبارتند از:
- source_column_id: شماره ستون منبع انجام کوئری.

- matched_column_id: شماره ستون سند مشابه یافت شده.
- matched_document_key: متناظر است با کلید اصلی جدولی که بر روی آن کوئری گرفته می‌شود.
- score: رتبه‌ی نسبی سند مشابه یافت شده.

```
-- Documents that are similar to document 1
SELECT S.source_document_title,
 SST.matched_document_key,
 D.title AS matched_document_title,
 SST.score
FROM
 (SEMANTICSIMILARITYTABLE
  (dbo.Documents, doccontent, 1) AS SST
  INNER JOIN dbo.Documents AS D
ON SST.matched_document_key = D.id)
 CROSS JOIN
  (SELECT title FROM dbo.Documents WHERE id=1)
AS S(source_document_title)
ORDER BY SST.score DESC;
```

Results Messages

	source_document_title	matched_document_key	matched_document_title	score
1	Columnstore Indices and Batch Processing	4	Additivity of Measures	0.1531117
2	Columnstore Indices and Batch Processing	2	Introduction to Data Mining	0.07079753
3	Columnstore Indices and Batch Processing	3	Why Is Bleeding Edge a Different Conference	0.02776711

در این کوئری، اسناد مشابه با سند شماره 1 یافت شده‌اند. مبنای جستجو نیز ستون doccontent، جدول dbo.Documents است.
از join بر روی matched_document_key و id جدول اصلی، مشخصات سند یافت شده را می‌توان استخراج کرد. کار CROSS JOIN تعریف شده، صرفا افزودن یک ستون مشخص به نتیجه‌ی خروجی کوئری است.
همانطور که در تصویر مشخص است، سند شماره 4 بسیار شبیه است به سند شماره 1. در ادامه قصد داریم بررسی کنیم که علت این شباهت چه بوده‌است؟

متد SEMANTICSIMILARITYDETAILSTABLE واژه‌های کلیدی مهم مشترک بین دو سند را بازگشت می‌دهد (سند منبع و سند مقصد). به این ترتیب می‌توان دریافت، چه واژه‌های کلیدی سبب شده‌اند تا این دو سند به هم شبیه باشند. ستون‌های خروجی آن عبارتند از:

- keyphrase: واژه‌ی کلیدی
- score: رتبه‌ی نسبی واژه‌ی کلیدی

```
-- Key phrases that are common across two documents
SELECT SSDT.keyphrase, SSDT.score
FROM SEMANTICSIMILARITYDETAILSTABLE
(dbo.Documents, doccontent, 1,
doccontent, 4) AS SSDT
ORDER BY SSDT.score DESC;
```

Results | Messages

	keyphrase	score
1	metadata	0.2517761
2	dimensions	0.1598883
3	data	0.1207339
4	queries	0.09310722
5	use	0.08706871
6	you	0.08695035
7	server	0.07584953
8	includes	0.0703855
9	processing	0.06637077
10	can	0.06478205
11	vahid	0.06091693
12	test	0.05837551
13	tables	0.05781268
14	aggregation	0.05597205
15	storage	0.05366294
16	called	0.05159317

در کوئری فوق قصد داریم بررسی کنیم چه واژه‌های کلیدی، سبب مشابهت سندهای شماره 1 و 4 شده‌اند و بین آن‌ها مشترک می‌باشند.

نظرات خوانندگان

نویسنده: ایمان دارابی
تاریخ: ۱۳۹۳/۰۱/۲۳ ۱۴:۱۱

با سلام
SEMANTICSIMILARITYTABLE آیا برای متون فارسی هم کار می‌کند.
من تست کردم نتیجه‌ای برای رکورهایی که با متون فارسی پر شدن بر نمی‌گردونه!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۳ ۱۵:۱

در ابتدای متن توضیح دادم: «همچنین باید دقت داشت که تمام زبان‌های پشتیبانی شده توسط FTS در حالت Semantic Search پشتیبانی نمی‌شوند. برای بررسی این مورد، دو کوئری ذیل را اجرا نمائید». فقط زبان‌هایی که حاصل گزارش زیر هستند Semantic Search در مورد آن‌ها صادق است:
(زبان عربی در FTS پشتیبانی می‌شود؛ اما نه در Semantic Search)

```
SELECT * FROM sys.fulltext_semantic_languages ORDER BY name
```

نویسنده: امیران
تاریخ: ۱۳۹۳/۰۹/۲۳ ۱۸:۲

ممنون جناب نصیری
مدتها درگیر Semantec Search با در نظر گرفتن Stemming برای زبان فارسی با استفاده از لوسین و زبان جاوا بودم.
سوالم این است که فیلترهای آفیس یا پی دی اف Adobe هنگام fulltext search زبان فارسی رو پشتیبانی می‌کند؟ یعنی با استفاده از این فیلترها امکان جستجوی فارسی در فایلهای آفیس یا پی دی اف وجود دارد؟
خود FTS در حالت جستجو در nvarchar(max) بصورت کامل از فارسی پشتیبانی می‌کند آیا امکان جستجوی فارسی در تایپهای varbinary(max) و فایلهای آفیس یا پی دی اف هم وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۳ ۱۸:۵

بله. از این لحاظ مشکلی نیست: « استفاده از Adobe iFilter برای جستجوی Full Text در فایل‌های PDF »

نویسنده: امیران
تاریخ: ۱۳۹۳/۰۹/۲۴ ۱۵:۲۳

ممنون جناب نصیری
امکان دریافت شماره صفحه ای که عبارت مورد جستجو در آن یافت شده هم وجود دارد؟ بعبارتی آیا می‌توان به صفحه ای که عبارت جستجو شده در آن وجود دارد به نحوی دسترسی داشت؟
تا جایی که من بررسی کردم در word اطلاعات صفحه (مثلا شروع یا پایان صفحه) در ساختار xml فایلهای word نگهداری نمی‌شود (بررسی شده با openxml) و هنگام نمایش در محیط Word صفحه ارایی انجام می‌گیرد. البته pdf را بررسی نکرده ام. در صورتیکه این امکان وجود داشته باشد جستجو در فایلهای داده مانند مجموعه آفیس و پی دی اف بسیار ساده خواهد بود. و نکته بعدی اینکه ما مدتها روی ریشه کلمات فارسی کار کرده ایم (wordnet فارسی برای کلمات متداول) و الگوریتمی هم برای آن تهیه کرده ایم که درصد خطای بسیار پایینی دارد آیا امکان توسعه semantic language برای پشتیبانی از زبان فارسی وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۴ ۱۷:۵۹

خیر. فقط بر روی روی خواص و متادیتای فایل‌ها می‌توان جستجوی تکمیلی را انجام داد. نمونه‌ی آن در انتهای بحث تهیه کوئری بر

روی ایندکس‌های Full Text Search در قسمت « 9) جستجو بر روی خواص و متادیتای فایل‌ها» و همچنین در مطلب « استفاده از Adobe iFilter برای جستجوی Full Text در فایل‌های PDF » در قسمت PdfSearchPropertyList مثال زده شده‌است.

نویسنده: رضا ایرانی
تاریخ: ۱۳۹۳/۱۰/۲۵ ۱۴:۱۴

با سلام.
من در جدول documents فیلد docexcerpt را در یک رکورد با Apple Iphone 6 64Gb و در رکوردی دیگر با Apple iphone 6 16Gb مقدار دهی کردم.
دنبال این هستم که چطور میشود کوئری ای ساخت که در صورت که بر روی مقدار Apple Iphone 6 64Gb جستجو صورت گرفت هر دو رکورد را بیاورد ولی با rankها مختلف.
تا الان نتونستم به نتیجه ای برسم. و فقط یکی از رکوردها را برمیگرداند.
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۵ ۱۴:۵۶

برای نمونه « کار با FREETEXT » را مطالعه کنید.

```
1 SELECT id, title, docexcerpt
2 FROM dbo.Documents
3 WHERE FREETEXT(docexcerpt, N'Apple Iphone 6 64Gb');
```

100 %

Results | Messages

	id	title	docexcerpt
1	5	تست یك	Apple Iphone 6 64Gb
2	6	تست دو	Apple iphone 6 16Gb

نویسنده: مونا کریمی
تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۱:۳۷

با سلام
راهی وجود داره که حین استفاده از Containstable تعیین کنیم که کلمات بصورت Case Sensitive سرچ بشن یا خیر؟ علاوه بر اون، میشه عباراتی رو که شامل کلمه مورد جستجو هستند برگردوند؟ (با استفاده از مقداردهی عبارت در حال جستجو به شکل '"name*"' میشه startWith رو پیاده سازی کرد اما '"*name*"' بصورت Contains عمل نمی‌کنه)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۴:۱۰

- کوئری‌های FTS به صورت پیش فرض case sensitive نیستند (^) و به این صورت طراحی شده‌است. البته اگر می‌خواهید این مورد همیشه صادق باشد، بهتر است عبارت جستجو شده را تبدیل به lower case یا uppercase کنید.

```
Full-text queries are not case-sensitive.
For example, searching for "Aluminum" or "aluminum" returns the same results.
```

مرجع

- «شامل کلمه» یا partial words پشتیبانی نمی‌شود . چون tokenizer مربوط به FTS کوچکترین جزئی که دارد یک word است.
این سؤال را هم بررسی کنید.

عنوان: **نگاهی به محتوا و نحوه‌ی تشکیل ایندکس‌های FTS**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۷ ۱۹:۱۳
آدرس: www.dotnettips.info
گروه‌ها: SQL Server

SQL Server به همراه تعدادی تابع سیستمی است که امکان مشاهده‌ی ریز جزئیات تشکیل دهنده‌ی ایندکس‌های FTS را فراهم می‌کنند. در ادامه قصد داریم این موارد را بررسی کنیم.

متد sys.dm_fts_index_keywords

این متد محتوای full-text index یک جدول را باز می‌گرداند. از آن می‌توان برای موارد ذیل استفاده کرد:
- آیا واژه کلیدی خاصی جزو full-text index است؟
- چه تعداد رکورد دارای واژه‌ی کلیدی خاصی هستند؟
- متداولترین واژه‌های کلیدی موجود در ایندکس کدامند؟
- کدام واژه را می‌توان به عنوان stop word تشخیص داد؟ شاید پس از بررسی، تشخیص داده شود که بهتر است متداولترین واژه‌ی کلیدی ایندکس شده، به stop list اضافه شود.

```
SELECT *
FROM sys.dm_fts_index_keywords(DB_ID(DB_NAME()), OBJECT_ID(N'dbo.Documents'));
```

Results | Messages

	keyword	display_term	column_id	document_count
80	0x00610067006700720065006700610074006...	aggregated	5	1
81	0x00610067006700720065006700610074006...	aggregating	4	1
82	0x00610067006700720065006700610074006...	aggregating	5	1
83	0x00610067006700720065006700610074006...	aggregation	5	2
84	0x00610067006700720065006700610074006...	aggregations	5	1
85	0x00610067006F	ago	5	1
86	0x0061006C0067006F007200690074006800 6D	algorithm	5	2
87	0x0061006C00690067006E00650064	aligned	5	1
88	0x0061006C006C	all	4	2

متد sys.dm_fts_index_keywords_by_document

این متد اطلاعاتی را در سطح اسناد باز می‌گرداند. کاربردهای آن می‌توانند شامل موارد زیر باشند:
- یافتن جمع تعداد واژه‌های کلیدی که یک full-text index دارا است.
- آیا واژه‌ی کلیدی مورد نظر، در ردیف در حال بررسی وجود دارد؟
- یک واژه‌ی کلیدی چندبار در کل ایندکس ظاهر شده‌است؟
- یک واژه‌ی کلیدی در یک ردیف یا سند مشخص، چندبار تکرار شده‌است؟
- یک ردیف یا سند، از چند واژه‌ی کلیدی تشکیل شده‌است؟

```
SELECT
 I.document_id,
 D.title,
 I.display_term,
 I.occurrence_count
FROM sys.dm_fts_index_keywords_by_document(DB_ID(DB_NAME()), OBJECT_ID(N'dbo.Documents')) AS I
INNER JOIN dbo.Documents D
ON D.id = I.document_id;
```

Results | Messages

	document_id	title	display_term	occurrence_count
1	4	Additivity of Measures	$100.00	1
2	4	Additivity of Measures	$130.00	1
3	4	Additivity of Measures	$150.00	1
4	4	Additivity of Measures	$2,000.00	2
5	4	Additivity of Measures	$200.00	1
6	4	Additivity of Measures	$230.00	1
7	4	Additivity of Measures	$3,000.00	1
8	4	Additivity of Measures	$350.00	1

متد sys.dm_fts_index_keywords_by_property

در قسمت‌های قبل، خواص و متادیتای اسناد آفیس را نیز ایندکس کردیم. این متد، اطلاعات مرتبط با خواص اسناد موجود در full-text index را باز می‌گرداند.

کاربردهای آن:

- چه محتوایی، در خاصیتی مشخص از سندی معلوم، ذخیره شده‌است؟
- خاصیت مورد نظر چه اندازه بکار رفته و تکرار شده‌است؟
- چه اسنادی دارای خاصیتی مشخص هستند؟

```
SELECT
 I.document_id,
 D.title,
 I.display_term,
 I.property_id
FROM sys.dm_fts_index_keywords_by_property(DB_ID(DB_NAME()), OBJECT_ID(N'dbo.Documents')) AS I
INNER JOIN dbo.Documents D
ON D.id = I.document_id;
```

Results Messages

	document_id	title	display_term	property_id
1	1	Columnstore Indices and Batch Processing	test	1
2	2	Introduction to Data Mining	test	1
3	3	Why Is Bleeding Edge a Different Conference	test	1
4	4	Additivity of Measures	test	1

متد sys.dm_fts_parser

متدهای قبلی که بررسی کردیم، نیاز به یک جدول و وجود full-text index بر روی آن دارند؛ اما متد dm_fts_parser خیر. این متد یک ورودی را گرفته و سپس تمام مراحل تهیه‌ی یک full-text index را به صورت پویا انجام می‌دهد.

کاربردهای آن:

- درک اینکه موتور FTS با یک ورودی رشته‌ای چگونه رفتار می‌کند.
- استخراج ایندکس‌های یک متن و ذخیره‌ی دستی آن در یک جدول.
- استخراج واژه‌های کلیدی یک رشته.
- آنالیز پویای INFLECTIONAL (مانند مثال زیر)

```
SELECT
 display_term,
 keyword
FROM sys.dm_fts_parser(N'"Mycustom string"', 1033, NULL, 0);
```

Results Messages

	display_term	keyword
1	mycustom	0x006D007900630075007300740006F006D
2	string	0x007300740072006900E0067

```
SELECT *
FROM sys.dm_fts_parser('FORMSOF(INFLECTIONAL,'+ 'term' + ')', 1033, NULL, 0);
```

Results | Messages

	keyword	group_id	phrase_id	occurrence	special_term	display_term	expansion_type	source_term
1	0x00740065007200650064	1	0	1	Exact Match	term's	2	term
2	0x00740065007200650064	1	0	1	Exact Match	termed	2	term
3	0x00740065007200650069006E0067	1	0	1	Exact Match	terming	2	term
4	0x00740065007200650073	1	0	1	Exact Match	terms	2	term
5	0x00740065007200650073	1	0	1	Exact Match	terms'	2	term
6	0x00740065007200650064	1	0	1	Exact Match	term	0	term

در اینجا پارامتر دوم آن شماره زبان مورد استفاده است. پارامتر سوم مشخص کننده‌ی stop list می‌تواند باشد و پارامتر سوم حساسیت به لهجه را مشخص می‌کند.

نظرات خوانندگان

نویسنده: امیرحسین ابراهیمیان
تاریخ: ۱۳۹۴/۰۴/۱۳ ۱۹:۴۳

ممنون؛ میشه semantic search را با entityframework شبیه سازی کرد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۱۳ ۱۹:۵۹

« استفاده از Full text search توسط Entity Framework »

عنوان: **استفاده از Adobe iFilter برای جستجوی Full Text در فایل‌های PDF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۹ ۱۰:۵۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server

در قسمت‌های قبل، نحوه‌ی کار با فیلترهای FTS آفیس را بررسی کردیم. شرکت Adobe نیز برای جستجوی Full-Text بر روی فایل‌های PDF، یک iFilter خاص را طراحی کرده‌است که نسخه‌ی آخر آن‌را از آدرس ذیل می‌توانید دریافت کنید:

http://www.adobe.com/support/downloads/product.jsp?product=1&platform=Windows

یک تجربه‌ی مهم : نگارش 11 آن را با SQL Server X64 تست کردم کار نکرد. اما **نگارش 9** کار می‌کند.

مستندات کامل

پس از نصب ابتدایی آن، مراحل ذیل را برای فعال سازی آن باید طی کرد:

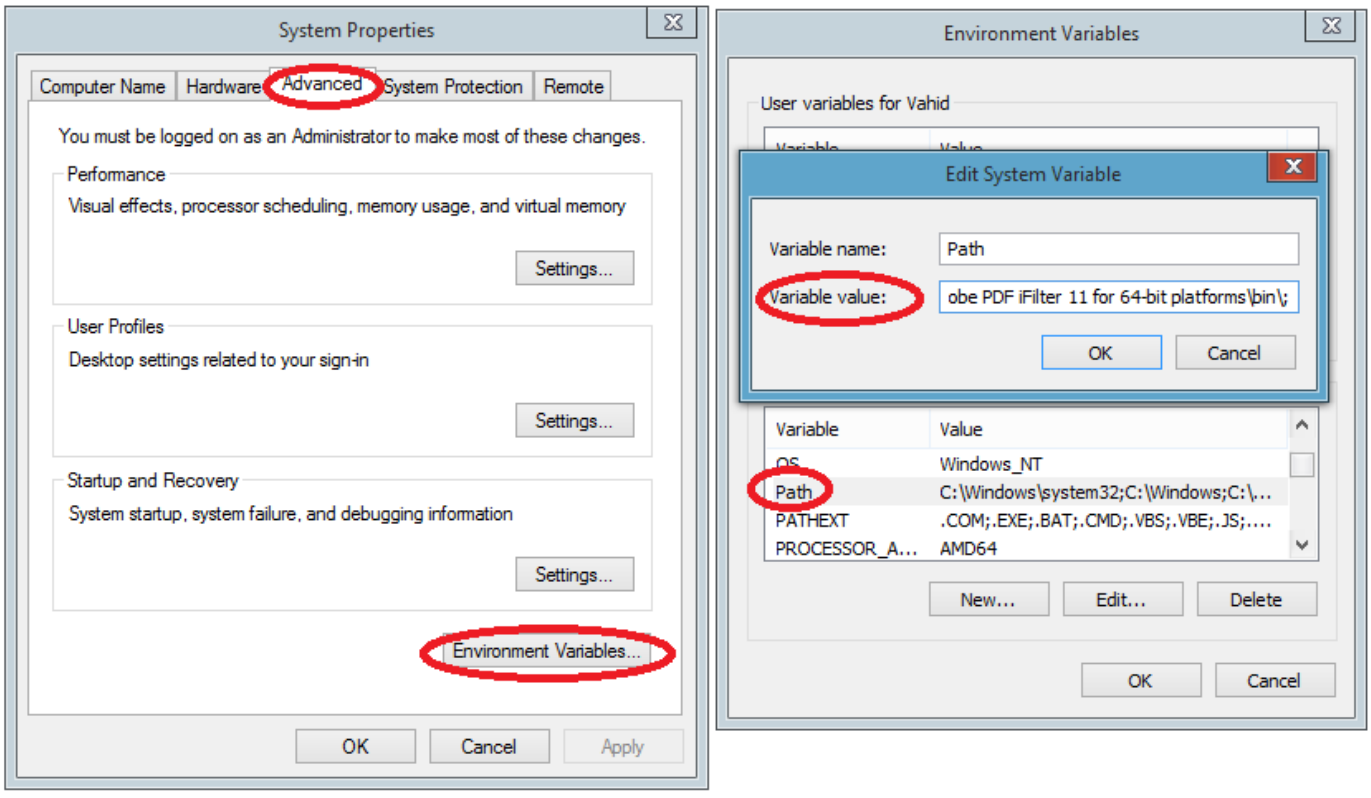
1) تنظیم مسیر پوشه bin نصب فیلتر (مهم!)

```
Start > Control Panel > System > Advanced
Environment Variables -> System Variables -> find PATH
```

مسیر فوق را در تنظیمات ویندوز یافته و سپس به انتهای Path، آدرس پوشه bin فیلتر نصب شده را اضافه کنید:

```
C:\Program Files\Adobe\Adobe PDF iFilter 9 for 64-bit platforms\bin\
```

دقت داشته باشید که این مسیر باید به \ ختم شود.
سپس کل سیستم را ری استارت کنید.



2) ثبت آن در وهله‌ی جاری SQL Server

برای این منظور ابتدا دستورات ذیل را جرا کنید:

```
exec sys.sp_fulltext_service 'load_os_resources', 1;
EXEC sp_fulltext_service 'verify_signature', 0
EXEC sp_fulltext_service 'update_languages'; -- update language list
EXEC sp_fulltext_service 'restart_all_fdhosts'; -- restart daemon
reconfigure with override
```

گزینه‌ی verify_signature مربوط به فایل‌های iFilter ایی است که امضای دیجیتال ندارند.

سپس در management studio یکبار بر روی وهله‌ی جاری کلیک راست کرده و گزینه‌ی Restart را انتخاب کنید (مهم).

3) پس از ری‌استارت SQL Server، **اطمینان حاصل کنید** که این فیلتر جدید نصب شده‌است:

```
exec sys.sp_help_fulltext_system_components 'filter';
```

Results Messages

	componenttype	componentname	clsid	fullpath	version
94	filter	.pdf	E8978DA6-047F-4E3D-9C78-CDBE46041603	C:\Program Files\Adobe\Adobe PDF iFilter 11 for 64-bit ...	11.0.1.36

و یا کوئری ذیل نیز برای این منظور مفید است:

```
SELECT document_type, path from sys.fulltext_document_types where document_type = '.pdf'
```

4) در ادامه برای استفاده از آن **آزمایش ذیل** را ترتیب خواهیم داد.

ایجاد یک جدول جدید که فایل‌های باینری PDF را در خود ذخیره می‌کند:

```
CREATE TABLE PdfDocuments
(
id INT IDENTITY(1,1) NOT NULL,
doctype NCHAR(4) NOT NULL,
doccontent VARBINARY(MAX) NOT NULL,
CONSTRAINT PK_PdfDocuments
PRIMARY KEY CLUSTERED(id)
);
```

ستون‌های doctype معرف نوع سند و doccontent ذخیره کننده‌ی محتوای کامل فایل‌های PDF خواهند بود.

سپس چند رکورد را در آن ثبت می‌کنیم. برای نمونه دو مقاله‌ی خروجی PDF سایت جاری را در این جدول ثبت خواهیم کرد:

```
INSERT INTO PdfDocuments(doctype, doccontent)
SELECT
N'PDF',
bulkcolumn FROM OPENROWSET (BULK 'C:\Users\Vahid\Downloads\1732-DotNetTips.pdf', SINGLE_BLOB) AS doc;

INSERT INTO PdfDocuments(doctype, doccontent)
SELECT
N'PDF',
bulkcolumn FROM OPENROWSET (BULK 'C:\Users\Vahid\Downloads\1733-DotNetTips.pdf', SINGLE_BLOB) AS doc;
```

در ادامه علاقمندیم تا بر روی خواص و متادیتای فایل‌های PDF نیز بتوانیم جستجوی FTS انجام دهیم. به همین منظور search propery list متناظری را نیز تعریف خواهیم کرد. همانطور که در قسمت‌های قبل عنوان شد، نیاز است GUID هر خاصیت را برای

تعریف از سازنده‌ی iFilter دریافت کرد. این اطلاعات در سند ذیل مستند شده‌اند:

http://www.ifiltershop.com/downloads/pdfplusfilter/readme.html

http://msdn.microsoft.com/en-us/library/ms692560%28v=vs.85%29.aspx

```
-- Search property list
CREATE SEARCH PROPERTY LIST PdfSearchPropertyList;
GO
ALTER SEARCH PROPERTY LIST PdfSearchPropertyList
ADD 'Author'
WITH (PROPERTY_SET_GUID = 'F29F85E0-4FF9-1068-AB91-08002B27B3D9',
PROPERTY_INT_ID = 4,
PROPERTY_DESCRIPTION = 'Author - author of a given item.');
GO
```

در اینجا اگر علاقمند بودید، stop list معرفی شده در قسمت‌های قبل را نیز می‌توان افزود.

```
CREATE FULLTEXT STOPLIST SQLStopList;
GO
-- Add a stopwords
ALTER FULLTEXT STOPLIST SQLStopList ADD N'به' LANGUAGE 'English';
ALTER FULLTEXT STOPLIST SQLStopList ADD N'با' LANGUAGE 'English';
--.....
```

سپس یک کاتالوگ FTS و ایندکس Full-Text ایی را بر روی این جدول ایجاد می‌کنیم:

```
-- Full-text catalog
CREATE FULLTEXT CATALOG PdfDocumentsFtCatalog;
GO

-- Full-text index
CREATE FULLTEXT INDEX ON PdfDocuments
(
  [doccontent] TYPE COLUMN [doctype]
  Language 1033
  STATISTICAL_SEMANTICS
)
KEY INDEX PK_PdfDocuments
ON PdfDocumentsFtCatalog
WITH STOPLIST = SQLStopList,
  SEARCH PROPERTY LIST = PdfSearchPropertyList,
  CHANGE_TRACKING AUTO;
GO
```

آیا کار می‌کند ؟ چیزی ایندکس شده‌است؟

```
SELECT
 I.document_id,
 I.display_term,
 I.occurrence_count
FROM sys.dm_fts_index_keywords_by_document(DB_ID(DB_NAME()), OBJECT_ID(N'dbo.PdfDocuments')) AS I
INNER JOIN dbo.PdfDocuments D
ON D.id = I.document_id;
```

Results Messages

	document_id	display_term	occurrence_count
1...	2	عمل	2
1...	2	عملياة	2
1...	1	عنوان	4
1...	2	عنوان	1
1...	2	فايل	1

انجام دو کوئری **بر روی آن** . یکی برای یافتن متنی ساده و دیگری برای یافتن خواص

```
SELECT *
FROM PdfDocuments
WHERE CONTAINS(doccontent, N'است')

SELECT
 I.document_id,
 I.display_term,
 I.property_id
FROM sys.dm_fts_index_keywords_by_property(DB_ID(DB_NAME()), OBJECT_ID(N'dbo.PdfDocuments')) AS I
INNER JOIN dbo.PdfDocuments D
ON D.id = I.document_id;
```

Results Messages

	id	doctype	doccontent
1	1	.pdf	0x255044462D312E340A25E2E3CFD30A31352030206F626A...
2	2	.pdf	0x255044462D312E340A25E2E3CFD30A31382030206F626A...

	document_id	display_term	property_id
1	1	مسعود	1
2	2	مسعود	1
3	1	پاکدل	1
4	2	پاکدل	1