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

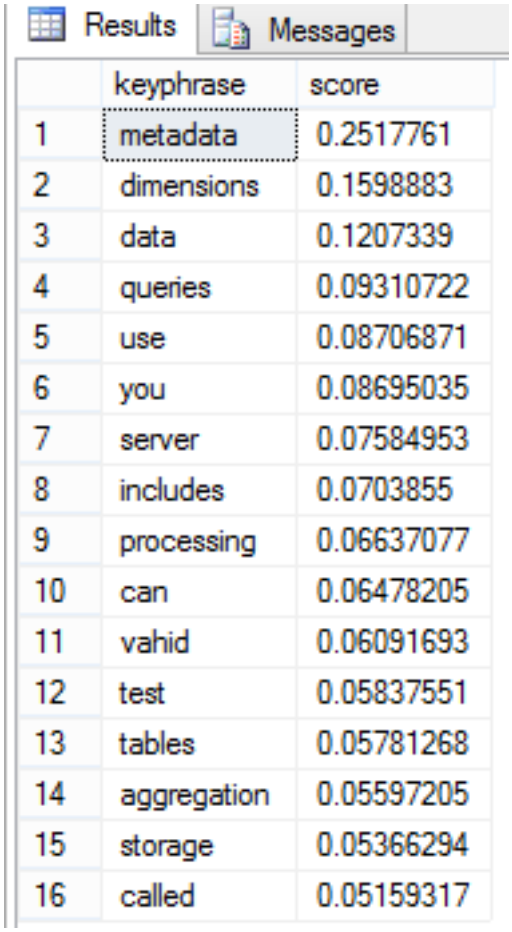
	source_document_title	matched_document_key	matched_document_title	score
1	Columnstore Indices and Batch Processing	4	Additivity of Measures	0.1531117
2	Columnstore Indices and Batch Processing	2	Introduction to Data Mining	0.07079753
3	Columnstore Indices and Batch Processing	3	Why Is Bleeding Edge a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