Полнотекстовый поиск и индексы.

(Full-Text Search)

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX
index_name
ON { *table* | *view* } (*column* [ASC | DESC] [,...*n*])
[WITH < index_option > [,...*n*]]
[ON *filegroup*]

< index_option > :: =
{ PAD_INDEX |
FILLFACTOR = *fillfactor* |
IGNORE_DUP_KEY |
DROP_EXISTING |
STATISTICS_NORECOMPUTE |
SORT_IN_TEMPDB
}

unique – значение индексируемой группы должно быть уникально

clustered – данные упорядочиваются физически (по умолчанию это используется для всех primary keys)

nonclustered – используется вектор индексов, данные физически не упорядочиваются

```
CREATE TABLE t1 (a int, b int, c AS a/b)
GO
CREATE UNIQUE CLUSTERED INDEX Idx1 ON t1.c
GO
INSERT INTO t1 VALUES ('1', '0')
GO
```


Средства **SQL** для проверки соответствия текста заданному шаблону

- LIKE – возвращает true, если строковое поле содержит шаблон

match_expression [NOT] LIKE *pattern* [ESCAPE *escape_character*]

- PATINDEX или CHARINDEX – возвращает позицию заданной подстроки в строке:

1. PATINDEX **(** '*%pattern%*' **,** *expression* **)**
2. CHARINDEX **(** *expression1* **,** *expression2* [**,** *start_location*] **)**

Пример: Like

- USE pubs
- GO
- CREATE PROCEDURE find_books2 @au_lname varchar(20)

AS SELECT t.title_id, t.title

FROM authors a, titleauthor ta, titles t

WHERE a.au_id = ta.au_id AND ta.title_id = t.title_id AND a.au_lname LIKE @au_lname + '%'

- EXEC find_books2 'ring'

Результат работы процедуры:

title_ id title --------

MC3021 The Gourmet Microwave

PS2091 Is Anger the Enemy?

PS2091 Is Anger the Enemy?

PS2106 Life Without Fear (4 row(s) affected)

Пример: PATINDEX

- USE Northwind
- GO
- SELECT CategoryID, PATINDEX('%candies%', Description)AS POSITION FROM Categories WHERE PATINDEX('%candies%', Description) <> 0
- Результатом будут строки, где в поле Description будет найдена подстрока 'candies' с произвольным началом и концом

Пример: CHARINDEX

- USE pubs
- SELECT CHARINDEX('wonderful', notes)

FROM titles

WHERE title_id = 'TC3218'

Результат- позиция подстроки 'wonderful‘ в поле notes таблицы titles :

46

(1 row(s) affected)

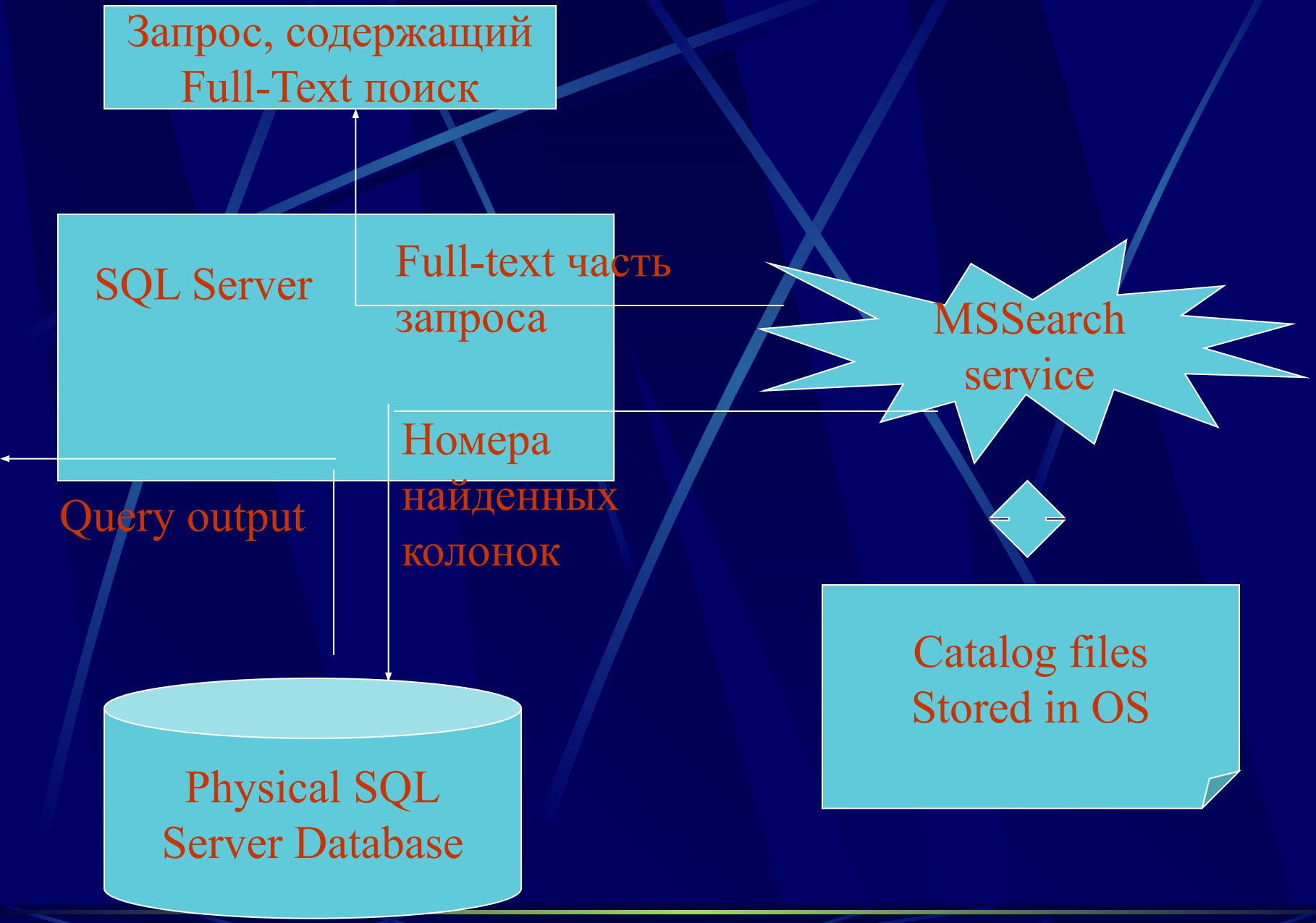
- Оба эти метода ограничены образцом
- Full-Text Search позволяет отслеживать как точное соответствие слову в шаблоне, так и словоформы (Например шаблон drink, а найти нужно не только drink, но и drunk)
- Помимо поиска по plain text поддерживается поиск по документам перечисленных ниже типов (содержимое этих документов должно содержаться в image поле)

Поддерживаемые типы документов

- Word (*.doc)
- Excel (*.xls)
- PowerPoint (*.ppt)
- Text (*.txt)
- HTML (*.htm или *.html)

Кроме того многие производители создают собственные фильтры для поддержки полнотекстового поиска по своим документам (например Adobe для Adobe Acrobat)

- Full-Text Search взаимодействует с MS Search service, который способен индексировать слова, содержащиеся в обработанных файлах для организации в дальнейшем поиска по этим словам (это используется например в Microsoft Index Server)
- Для таблиц MS SQL сервера строятся полнотекстовые индексы, которые хранятся отдельно от базы данных (используется file-based storage архитектура)


Запрос, содержащий Full-Text поиск
SQL Server
Full-text часть запроса
MSSearch service
Номера найденных колонок
Query output
Catalog files Stored in OS
Physical SQL Server Database

Для того, чтобы **Full-Text Search** начал работать необходимо:

- Включить поддержку Full-Text для базы
- Создать Full-Text Catalog
- Создать Full-Text Index
- Заполнить Full-Text Index. В процессе заполнения (population), добавленные в Full-Text Index текстовые поля таблиц просматриваются и составляется «словарь» слов, содержащихся в каждом поле каждой записи, который в дальнейшем используется для поиска

Включение поддержки **Full-Text** для базы

EXEC sp_fulltext_database [@action =]
‘{enable|disable}’

USE Northwind
EXEC sp_fulltext_database @action = ‘Enable’

Создание **Full-Text Catalog**

Для управления Full-Text каталогом используется процедура sp_fulltext_catalog

EXEC sp_fulltext_catalog [@ftcat =] ‘<name of catalog>’,
[@action =]
‘{create|drop|start_incremental|start_full|stop|rebuild}’
[, [@path =] ‘<root directory>’]

USE Northwind
EXEC sp_fulltext_catalog @ftcat = ‘NorthWindFullText’,
@action = ‘CREATE’

Включение поддержки **Full-Text Search** для таблиц

EXEC sp_fulltext_table [@tabname =] '[<owner>.]<table>',
[@action=]
'{create|drop|activate|deactivate|start_change_tracking
|stop_change_tracking|start_background_update_index
|stop_background_update_index|update_index|start_full
|start_incremental|stop}'
[,[@ftcat =] '<fulltext catalog>', [@keyname =] 'index name']

USE Northwind
EXEC sp_fulltext_table @tabname = 'Employees',
@action = 'create',
@ftcat = 'NorthwindFullText',
@keyname = 'PK_Employees'

Добавление колонки в **Full-Text** индекс

sp_fulltext_column [@tabname =] ‘[<owner>.]<table>’,
[@colname =] ‘<column name>’,
[@action =] ‘{add|drop}’
[,[@language =] ‘<language>’]
[, [@type_colname =] ‘<type column name>’]

```
USE Northwind
EXEC sp_fulltext_column @tabname = ‘Employees’,
        @colname = ‘Notes’
        @action = ‘add’
```

После всех этих действий необходимо запустить заполнение (population) индекса

EXEC sp_fulltext_table @tabname = ‘Employees’, @action = ‘start_full’,

Full-Text Query Syntax

Существует 2 условных выражения:

- CONTAINS – строгое соответствие шаблону
- FREETEXT – нестрогое соответствие (словоформы)

и 2 эквивалентных выражения, возвращающие таблицы:

- CONTAINSTABLE
- FREETEXTTABLE

CONTAINS

CONTAINS ({<column>|*} , ‘<search condition>’)

SELECT EmployeeID, LastName, FirstName
FROM Employees
WHERE CONTAINS(*, ‘Course’)

FREETEXT

FREETEXT({<column>|*} , ‘<search condition>’)

SELECT EmployeeID, LastName, FirstName
FROM Employees
WHERE **FREETEXT**(*, ‘Course’)

CONTAINSTABLE

CONTAINSTABLE (<table>, {<column>|*} , ‘<contains search condition>’ [, <top ‘n’>])

SELECT *
FROM CONTAINSTABLE(Employees, *, ‘Course’)

Возвращает таблицу с 2мя полями – key и rank.

- Key – ключевое поле исходной таблицы (Employees), соответствующее найденным записям
- Rank – уровень соответствия найденной записи заданному шаблону (число от 0 до 1000)

```
SELECT Rank, EmployeeID, LastName,
   FirstName, Notes
FROM CONTAINSTABLE(Employees, *,
   'Course') AS ct
JOIN Employees AS e
 ON ct.[KEY] = e.EmployeeID
```


```
SELECT Rank, EmployeeID, LastName,
   FirstName, Notes
FROM FREETEXTTABLE(Employees, *,
   'Course') AS ct
JOIN Employees AS e
 ON ct.[KEY] = e.EmployeeID
```


Использование фраз в шаблонах

SELECT EmployeeID, LastName, FirstName, Notes

FROM Employees

WHERE CONTAINS(*, ‘ ”University of California” ’)

Этот запрос найдет все записи, в поле Notes которых есть слова ‘University’, ‘of’ и ‘California’

Можно использовать and, or и not

SELECT EmployeeID, LastName, FirstName, Notes

FROM Employees

WHERE FREETEXT(*, ‘ ”University of California” ’)

Этот запрос найдет все записи, в поле Notes которых есть слова ‘University’, ‘of’ ИЛИ ‘California’. ‘Of’ будет проигнорировано как ‘noise word’. Список таких слов содержится в обычном текстовом файле

Использование **NEAR**

NEAR – между словами шаблона имеется не более 8-10 других слов:

```
SELECT Rank, EmployeeID, LastName, FirstName, Notes
FROM CONTAINSTABLE(Employees, *, ‘completed near sales’)
    AS ct
JOIN Employees AS e
 ON ct.[KEY] = e.EmployeeID
```

Пример: NEAR

- USE Northwind
- GO
- SELECT FT_TBL.Description, FT_TBL.CategoryName, KEY_TBL.RANK FROM Categories AS FT_TBL
  INNER JOIN CONTAINSTABLE (Categories, Description, '("sweet and savory" NEAR sauces) OR ("sweet and savory" NEAR candies)') AS KEY_TBL ON FT_TBL.CategoryID = KEY_TBL.[KEY] WHERE KEY_TBL.RANK > 2 AND FT_TBL.CategoryName <> 'Seafood'
  ORDER BY KEY_TBL.RANK DESC

Использование префиксов

SELECT LastName, FirstName, Notes

FROM Employees

WHERE CONTAINS(*, ‘ ”grad*” ’)

Будет искать все слова начинающиеся с grad

CONTAINS(*, ‘grad*’) будет искать подстроку grad* (важно ставить кавычки)

Задание весов для частей шаблона

```
SELECT Rank, EmployeeID, LastName, FirstName, Notes
FROM CONTAINSTABLE(Employees, Notes,
'ISABOUT (BA WEIGHT (.2), BS WEIGHT (.4), MA WEIGHT (.8))') AS ct
JOIN Employees AS e
 ON ct.[KEY] = e.EmployeeID
ORDER BY Rank DESC
```

То есть найти ‘MA’ в 2 раза важнее, чем ‘BS’, а ‘BS’ в 2 раза важнее, чем ‘BA’. Записи, в которых содержится ‘MA’ будут иметь более высокий RANK. Сначала будут выведены, магистры (MA), затем бакалавры (BS), затем бакалавры искусств (BA) – производится ранжирование по уровню образования.

Поиск словоформ с использованием **CONTAINS**

- По умолчанию CONTAINS ищет точное соответствие шаблону

SELECT LastName, FirstName, Notes

FROM Employees

WHERE CONTAINS(Notes, ‘FORMSOF (INFLECTIONAL, graduate, degree)’)

Будет искать все слова однокоренные graduate или degree (FREETEXT работает так по умолчанию)