Лабораторная работа 4.
Создание запросов и фильтров

Цель работы: научиться создавать запросы и фильтры в среде SQL Server Management Studio.

Теоретические сведения

Запросы предназначены для связи одной или нескольких таблиц. Также они могут осуществлять отбор отдельных полей из таблицы и производить фильтрацию данных согласно условию, наложенному на одно или несколько полей. Такие запросы называют *фильтрами*.

Для реализации запросов используют специальный язык запросов SQL (Structured Query Language), в MS SQL Server используется процедурное расширение языка SQL компанией Microsoft, T-SQL (Transact-SQL).

В информационных системах запросы могут храниться как на стороне клиентского приложения, так и на стороне сервера. Если запрос хранится на стороне клиента, то он прописывается внутри объекта связи. В этом случае клиентское приложение не зависит от файла данных. Файл данных содержит только таблицы, поэтому, мы легко можем модифицировать клиентское приложение, не затрагивая файл данных, но в этом случае запрос передается серверу через сеть, что может вызвать проблемы с безопасностью.

Если запрос хранится или выполняется на сервере, то сам запрос выступает в качестве компонента БД, вся передача информации происходит внутри файл данных, т.е. внутри самого сервера, клиентскому приложению только передаются результаты выполнения запроса. В этом случае обеспечивается высокая защита данных, но в случае изменения запроса придется менять сам файл данных.

Все запросы делятся на статические и динамические. Структура *статических запросов* неизменна в ходе работы с программой, а динамические запросы могут меняться в зависимости от ситуации.

Обычно *динамические запросы* могут быть реализованы только при помощи запросов, выполняющихся на стороне клиента. Если необходимо реализовать динамические запросы, которые выполняются на стороне сервера, то в этом случае необходимо использовать *хранимые процедуры*. Подробно хранимые процедуры будут рассмотрены в лабораторной работе №5.

В основном запрос или хранимая процедура либо реализует связь между таблицами, либо осуществляет фильтрацию данных, некоторые SQL запросы также могут производить вычисления.

В случае связей между таблицами одна таблица всегда выступает первичной, а другая – вторичной, связь происходит при помощи полей связи. При связи сопоставляются записи с одинаковыми значениями полей связи. *Первичная таблица* всегда заполняется первой, а ее поле связи заполняется

автоматически (тип данных – счётчик). *Вторичная таблица* всегда заполняется после заполнения первичной таблицы, значения ее поля связи подставляется из значений поля связи первичной таблицы. Поля связи должны иметь одинаковый тип данных.

Существует четыре вида связи между таблицами:

1) одна к одной – одному полю в первичной таблице соответствует одно поле во вторичной таблице;
2) одна ко многим – одному полю в первичной таблице соответствует несколько полей во вторичной таблице;
3) многие к одной – нескольким полям в первичной таблице соответствует одно поле во вторичной таблице;
4) многие ко многим – одному полю в первичной таблице соответствует несколько полей во вторичной таблице и наоборот.

Запросы с первым видом связи называются *простыми*, а с остальными видами связи – *сложными*.

Чтобы создать запрос необходимо сделать активной БД, для которой создается запрос, затем в рабочей области редактора запросов создать запрос с помощью команды `SELECT`, имеющей следующий синтаксис:

```
SELECT [ ALL | DISTINCT ]
    [ TOP (expression) [ PERCENT ] ]
    {
      *
      | { table_name }.*
      | {
          [{ table_name }. ]
               { column_name }
          | expression
          [ [ AS ] column_alias ]
         }
      | column_alias=expression
    }
    [ INTO new_table ]
    [ FROM { table_name } ]
    [ WHERE search_condition ]
    [ GROUP BY column_expression ]
    [ HAVING search_condition ]
[ORDER BY { order_by_expression [ ASC | DESC ] }]
```

Здесь параметры `ALL|DISTINCT` показывают, какие записи окажутся в результирующей таблице: `ALL` – дублирующиеся записи могут оказаться в результате, `DISTINCT` – только уникальные записи окажутся в результате; `TOP expression PERCENT` определяет, сколько записей попадет в результат запроса (в количестве записей или в процентах); `table_name` – имя таблицы; `column_name` – имя столбца; `INTO new_table` – результат запроса может быть помещен в таблицу с именем `new_table`; `search_condition` определяет условие для осуществления поиска данных в таблице (после

WHERE) или для группировки данных в результате запроса (после HAVING); ORDER BY позволяет сгруппировать данные, полученные в результате запроса, по столбцам (order_by_expression) в порядке возрастания или убывания (ASC | DESC).

Замечания:

1) Если выбираются поля из разных таблиц с одинаковыми именами нужно указывать и имя таблицы table_name.column_name.

2) Полям можно присваивать псевдонимы, следующим образом column_name AS column_alias.

3) Если необходимо выбрать все поля из таблицы, то их можно заменить значком «*».

4) Раздел INTO. Если присутствует этот раздел, то на основе результатов запроса создается новая таблица. Параметр INTO это имя новой таблицы.

5) Раздел FROM. Здесь указываются таблицы и запросы, через запятую, которые участвуют в новом запросе. В разделе FROM также можно задавать сложные связи: связь поля одной таблицы с несколькими полями другой таблицы.

6) Раздел WHERE. Данный раздел используют для создания простых запросов, в этом случае в качестве условия указываются связываемые поля, либо этот раздел используют для создания фильтров, и здесь указываются условия отбора. В условиях отбора мы можем использовать стандартные логические операторы NOT, OR, AND.

7) Раздел GROUP BY определяет поле для группировки записей в запросе.

8) Раздел ORDER BY определяет поле для сортировки записей в запросе. Если указан параметр ASC, то будет производиться сортировка по возрастанию, если DESC – по убыванию. По умолчанию используется сортировка по возрастанию.

Кроме связывания таблиц и отбора данных оператор SELECT может использоваться для вычислений. В этом случае он имеет синтаксис:

```
SELECT expression
```

где expression – какое-то математическое выражение или функция.

В SQL Server существуют следующие встроенные агрегатные функции:

AVG	Возвращает среднее значение
CHECKSUM_AGG	Возвращает контрольную сумму значений
COUNT	Возвращает количество значений (результат имеет тип int)
COUNT_BIG	Возвращает количество значений (результат имеет тип bigint)
GROUPING	Определяет, является ли столбец, указанный в списке

	GROUP BY агрегированным. GROUPING возвращает 1 для агрегированных и 0 для не агрегированных столбцов
MAX	Возвращает максимальное значение
MIN	Возвращает минимальное значение
SUM	Возвращает сумму всех значений
STDEV	Возвращает среднеквадратичное отклонение всех значений
STDEVP	Возвращает среднеквадратичное отклонение для множества всех значений
VAR	Возвращает дисперсию всех значений
VARP	Возвращает дисперсию для множества всех значений

Примеры использования агрегатных функций:

`SELECT AVG(возраст) FROM Студенты` – выводит средний возраст студента из таблицы «Студенты».

`SELECT COUNT(ФИО) FROM Студенты` – выводит количество различных ФИО из таблицы «Студенты».

Пример

Рассмотрим пример создания статических запросов для база данных информационной системы по обслуживанию хранилищ, музеев, выставок и аукционов (см. примеры из предыдущих лабораторных работ). На следующем рисунке показана диаграмма этой БД:

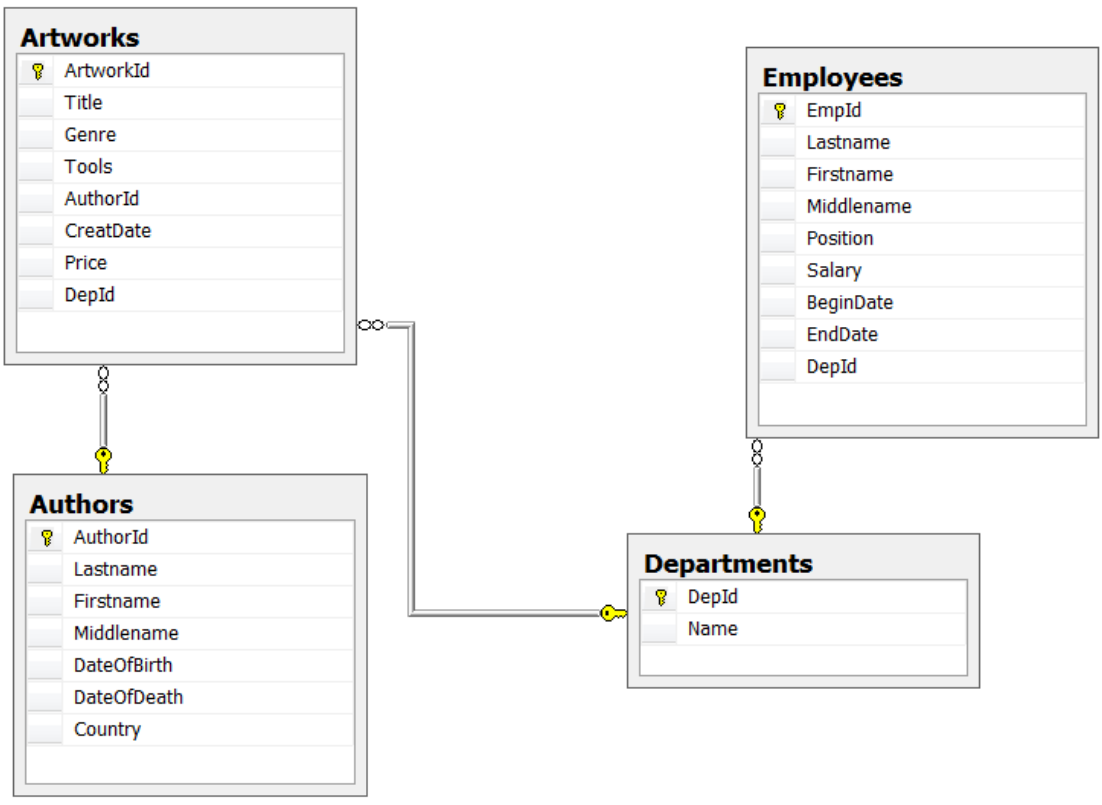


Рис. 1

Создадим запрос «Отдел кадров» (связывает таблицы «Employees» и «Departments» по полю «DepId») и фильтр для отображения сотрудников отдельных отделов (На основе запроса «Отдел кадров»).

Следующий программный код осуществляет внутреннее соединение таблиц «Employees» и «Departments» по внешнему ключу «DepId»:

```
SELECT E.Lastname, E.Firstname, E.Middlename,
       E.Position, D.Name, E.BeginDate
FROM dbo.Employees AS E
     JOIN dbo.Departments AS D
     ON E.DepId = D.DepId;
```

Результатом выполнения этого запроса будет динамическая таблица, содержащая все строки из таблицы «Employees», в которых значение атрибута «DepId» равно значению этого же атрибута из таблицы «Departments»:

Результаты | Сообщения

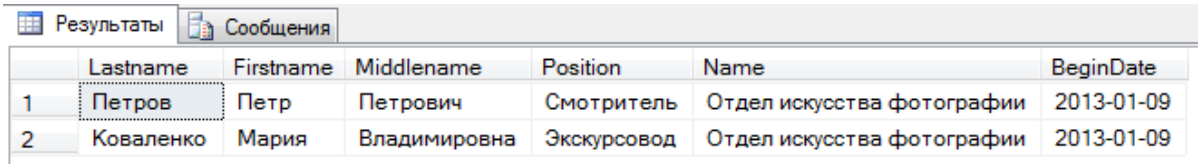
	Lastname	Firstname	Middlename	Position	Name	BeginDate
1	Иванов	Иван	Иванович	Реставратор	Отдел реставрации	2013-01-09
2	Петров	Петр	Петрович	Смотритель	Отдел искусства фотографии	2013-01-09
3	Степанов	Степан	Степанович	Начальник	Отдел гравюры и рисунка	2013-01-09
4	Федорова	Анна	Федоровна	Уборщица	Отдел искусства Античного мира	2013-01-09
5	Потапов	Виктор	Сергеевич	Реставратор	Отдел искусства стран Европы XIX-XX вв.	2013-01-09
6	Сергеев	Николай	Петрович	Экскурсовод	Отдел русского искусства XIX-XX вв.	2013-01-09
7	Коваленко	Мария	Владимировна	Экскурсовод	Отдел искусства фотографии	2013-01-09
8	Верещагин	Вадим	Петрович	Экскурсовод	Отдел гравюры и рисунка	2013-01-09
9	Сидорова	Татьяна	Викторовна	Экскурсовод	Отдел искусства Античного мира	2013-01-09
10	Семенова	Инна	Николаевна	Экскурсовод	Отдел искусства Древнего Востока	2013-01-09

Рис. 2

Для создания фильтра, выводящего данные сотрудников отдельных отделов нам нужно модифицировать предыдущий запрос следующим образом:

```
SELECT E.Lastname, E.Firstname, E.Middlename,
       E.Position, D.Name, E.BeginDate
FROM dbo.Employees AS E
     JOIN dbo.Departments AS D
     ON E.DepId = D.DepId AND
        D.Name = 'Отдел искусства фотографии';
```

Поле «Name» таблицы «Departments» содержит названия отдела, сравнивая его с конкретным именем отдела, мы можем организовать вывод информации о сотрудниках этого отдела:


Результаты Сообщения

	Lastname	Firstname	Middlename	Position	Name	BeginDate
1	Петров	Петр	Петрович	Смотритель	Отдел искусства фотографии	2013-01-09
2	Коваленко	Мария	Владимировна	Экскурсовод	Отдел искусства фотографии	2013-01-09

Рис. 3

Задание

Создать фильтры и запросы в соответствии с заданием для своего варианта.