**Лабораторная работа №5**

**Использование оператора SELECT языка SQL**

**Цель:** изучить основные возможности оператора SELECT языка SQL и получить практические навыки создания и выполнения запросов на этом языке при работе с СУБД Access.

Для получения **теоретических сведений** настоятельно рекомендуется при домашней подготовке изучить материалы по тематике лабораторной работы, представленные в открытых источниках.

Далее следует краткий конспект теоретического материала.

## Теоретические сведения

SQL-запросы, задающие выборку данных, записываются с использованием оператора SELECT, который является наиболее функциональным из всех операторов SQL.

Оператор **SELECT** читает данные из базы данных (БД) и возвращает их в виде таблицы результатов запроса. Оператор SELECT состоит из шести предложений, или инструкций.

Предложения SELECT и FROM являются обязательными. Четыре остальные включаются при необходимости. Функции каждого из предложений следующие:

1) в предложении SELECT указывается список возвращаемых столбцов, которые должны быть включены в таблицу результатов запроса. Возвращаемые столбцы могут содержать значения, считываемые из столбов таблиц базы данных, или значения, вычисляемые во время выполнения запроса;
2) в предложении FROM указывается список таблиц, которые содержат элементы данных, считываемые запросом;
3) WHERE показывает, что в таблицу результатов запроса следует включать только строки, удовлетворяющие условию поиска;
4) GROUP BY позволяет задавать итоговый запрос. Обычный запрос включает в таблицу результатов по одной строке для каждой строки из БД. Итоговый запрос вначале группирует строки БД по определённому признаку, а затем включает в таблицу результатов одну итоговую строку для каждой группы;
5) HAVING показывает, что в таблицу результатов запроса следует включать только некоторые из групп, созданных с помощью предложения GROUP BY. Для отбора включаемых групп используется условие поиска;
6) ORDER BY сортирует таблицу результатов запроса на основании данных, содержащихся в одном или нескольких столбцах.

В предложении SELECT, с которого начинаются все операторы SELECT, список возвращаемых столбцов содержит элементы, представляющие собой либо имя столбца, либо константу, либо выражение.

Имя столбца идентифицирует один из столбцов, содержащихся в таблицах, которые перечислены в предложении FROM. В языке Microsoft Access SQL принято имя столбца или таблицы, содержащее пробелы или другие знаки препинания, обязательно записывать в квадратных скобках. Когда в качестве возвращаемого столбца указывается имя столбца из таблицы БД, то СУБД просто берёт значение этого столбца для каждой из строк таблицы БД и помещает его в соответствующую строку таблицы результатов запроса.

Константа, записанная в списке возвращаемых столбцов, (например, строка символов, которая на языке Microsoft Access SQL заключается в апострофы или кавычки) показывает, что в каждой строке таблицы результатов запроса должно содержаться одно и то же значение.

Выражение показывает, что СУБД должна вычислять значение, помещаемое в таблицу результатов запроса, по формуле, заданной выражением.

Чтобы столбцы, содержащие константы или значения выражений, в таблице результатов имели более информативные надписи, в операторе SELECT предусмотрено в элементе списка возвращаемых столбцов указывать альтернативное имя (псевдоним) столбца после ключевого

слова AS (причем для Access псевдоним столбца, содержащий пробелы или знаки препинания следует заключать в квадратные скобки):

```
SELECT 'Получатель' AS Получатель, АдресПолучателя,
  'оплачивает' AS оплачивает, СтоимостьДоставки,
  'со скидкой' AS [со скидкой],
  СтоимостьДоставки*0.05 AS Скидка
FROM Заказы
```

Если необходимо получить содержимое всех столбцов таблицы, чтобы ознакомиться с её структурой и хранимыми в ней данными, то для этого в операторе SELECT предусмотрено использовать символ звёздочки, который означает, что требуется прочитать все столбцы из таблиц, перечисленных в предложении FROM. Например, содержимое таблицы ЗАКАЗЫ можно получить с помощью оператора:

```
SELECT * FROM Заказы
```

Таблица результатов этого запроса содержит все 14 столбцов таблицы ЗАКАЗЫ, которые расположены в том же порядке, что и в исходной таблице, хранящейся в БД.

По умолчанию в таблицу результатов запроса включаются все строки, в том числе и повторяющиеся. Например, оператор:

```
SELECT ГородПолучателя FROM Заказы
```

возвращает города всех получателей, и среди этих городов будут повторяющиеся.

Чтобы в таблице результатов запроса содержались только неповторяющиеся строки, в операторе SELECT перед списком возвращаемых столбцов записывается ключевое слово DISTINCT. Например, чтобы получить список разных городов, предыдущий оператор нужно дополнить этим ключевым словом:

```
SELECT DISTINCT ГородПолучателя FROM Заказы
```

**Отбор строк из таблиц**

SQL-запросы, считывающие из таблицы все строки, полезны при просмотре базы данных и создании итоговых отчетов. Однако чаще требуется выбрать из таблицы базы данных несколько строк и включить в таблицу результатов запроса только их. Чтобы указать, какие строки требуется отобрать, следует использовать предложение WHERE.

Предложение WHERE содержит условие поиска, определяющее, какие именно строки требуется прочитать. Например, чтобы получить информацию о том, какие заказы поступили из Москвы, нужно выполнить оператор:

```
SELECT КодЗаказа FROM Заказы
WHERE ГородПолучателя = "Москва"
```

Фактически условие поиска служит фильтром для строк таблицы. Строки, удовлетворяющие условию поиска, проходят через фильтр и становятся частью таблицы результатов.

В SQL используются различные условия поиска, позволяющие эффективно и естественным образом создавать различные типы запросов. В частности, возможны следующие условия поиска:

1) сравнение (значение одного выражения сравнивается со значением другого выражения);

2) проверка на принадлежность диапазону значений (проверяется, попадает ли указанное значение в определенный диапазон значений);

3) проверка на принадлежность множеству (проверяется, совпадает ли значение выражения с одним из значений, имеющихся в заданном множестве);

4) проверка на соответствие шаблону (проверяется, соответствует ли строковое значение, содержащееся в столбце, определенному шаблону);

5) проверка на равенство значению NULL (проверяется, содержится ли в столбце значение NULL).

**Сравнение**

Имеется шесть различных способов сравнения значений двух выражений с использованием знаков <, =, >, <=, <>, >=. Например, можно вывести список сотрудников, родившихся после определенной даты:

```
SELECT КодСотрудника, Фамилия, Должность, ДатаРождения
FROM Сотрудники
WHERE ДатаРождения>#31/12/1960#
```

При сравнении значений двух выражений может получиться один из таких результатов:

1) если сравнение истинно, то результат проверки имеет значение TRUE;
2) если сравнение ложно, то результат проверки имеет значение FALSE;
3) если хотя бы одно из двух выражений имеет неопределенное значение NULL, то результатом проверки будет значение NULL.

В таблицу результатов запроса включаются только те строки, для которых условие поиска имеет значение TRUE.

Проверка на принадлежность диапазону значений. В условие поиска этого вида входят три выражения:

1) выражение, которое задает проверяемое значение;
2) выражение, задающее нижний предел проверяемого диапазона;
3) выражение, задающее верхний предел проверяемого диапазона. Например, формирование списка сотрудников, родившихся в 1 -м квартале 1955 года, задается оператором:

```
SELECT КодСотрудника, Фамилия, Должность, ДатаРождения
FROM Сотрудники
WHERE ДатаРождения BETWEEN #01/01/1955# AND #03/31/1955#
```

При проверке на принадлежность диапазону верхний и нижний пределы считаются частью диапазона.

Для проверки выхода значения за пределы диапазона следует использовать ключевое слово NOT. Например, список адресов получателей заказов, для которых скидка меньше 5 рублей или больше 10 рублей, можно получить с помощью оператора:

```
SELECT АдресПолучателя, СтоимостьДоставки*0.05 AS Скидка
FROM Заказы
WHERE СтоимостьДоставки*0.05 NOT BETWEEN 5.00 AND 10.00
```

**Проверка на членство во множестве**

С помощью этого условия поиска можно узнать, соответствует ли значение данных какому-либо значению из заданного списка.

Например, следующий оператор выводит список сотрудников с заданными кодами:

```
SELECT * FROM Сотрудники
WHERE КодСотрудника IN (1,3,5,9)
```

Список сотрудников, родившихся указанные дни, формируется оператором

```
SELECT * FROM Сотрудники
WHERE ДатаРождения IN (#29/05/60#, #27/01/66#)
```

С помощью проверки NOT IN можно обнаружить значения данных, не являющиеся членами заданного множества:

```
SELECT * FROM Заказы
WHERE ГородПолучателя NOT IN ("Москва", "Берлин")
```

Проверяемое выражение может быть любым допустимым выражением, однако обычно оно представляет собой имя столбца. Если результатом проверяемого выражения является значение NULL, то проверка возвращает значение NULL. Все константы в списке проверяемых значений должны быть одного и того же типа, который должен быть совместим с типом проверяемого выражения.

**Проверка на соответствие шаблону**

Для формирования таблицы результатов, состоящей из строк, в которых значение некоторого текстового столбца (поля) совпадает с заданным текстом, можно использовать простое сравнение. Например, следующий оператор создает таблицу результатов из строк, в которых встречаются заданные имя и фамилия представителя поставщика:

```
SELECT * FROM Поставщики
WHERE ОбращатьсяК = "Sven Petersen"
```

Однако если необходимо получить сведения обо всех представителях с фамилией Petersen, то следует воспользоваться проверкой LIKE, которая позволяет определить, соответствует ли значение данных в столбце некоторому шаблону. Шаблон представляет собой строку символов, в которую может входить один или более подстановочных знаков, интерпретируемых особым образом. В стандарте языка SQL в качестве подстановочных знаков используются «%» и «_» (подчеркивание), а в Microsoft Access SQL вместо них предусмотрены соответственно знаки «*» и «?».

Подстановочный знак «*» совпадает с любой последовательностью из нуля или более символов. Например, чтобы получить сведения обо всех представителях с фамилией Petersen, нужно воспользоваться шаблоном:

```
SELECT * FROM Поставщики
WHERE ОбращатьсяК LIKE "*Petersen"
```

Ключевое слово LIKE указывает, что необходимо сравнивать содержимое столбца «ОбращатьсяК» с шаблоном '*Petersen', которому соответствуют все имена, заканчивающиеся словом «Petersen».

Подстановочный знак «?» совпадает с любым отдельным символом. Например, для получения сведений о товарах, поставляемых в таре емкостью от 500 до 599 миллилитров, можно воспользоваться таким шаблоном:

```
SELECT * FROM Товары
WHERE ЕдиницаИзмерения LIKE "* 5?? мл*"
```

Подстановочные знаки можно помещать в любое место шаблона, и в одном шаблоне может содержаться несколько подстановочных знаков.

С помощью конструкции NOT LIKE можно выбирать строки, которые не соответствуют шаблону.

Проверка на соответствие шаблону применима только к столбцам, имеющим строковый тип данных или содержащим дату и время. Если в столбце содержится неопределенное значение NULL, то результатом проверки будет значение NULL.

**Проверка на равенство значению NULL**

В ряде случаев бывает необходимо проверять значения в столбцах на равенство значению NULL и помещать соответствующие строки в таблицу результатов. Для этого в языке SQL предусмотрена специальная проверка на равенство значению NULL. В следующем операторе

проверка на NULL используется для поиска заказанных товаров, для которых не определена скидка (не путать со скидкой, равной нулю):

```
SELECT * FROM Заказано WHERE Скидка IS NULL
```

Инвертированная форма проверки на NULL позволяет отыскать строки, которые не содержат значения NULL. Например, список заказанных товаров, на которые не забыли установить величину скидки, формирует оператор:

```
SELECT * FROM Заказано
WHERE Скидка IS NOT NULL
```

В отличие от условий поиска, рассмотренных ранее, проверка на NULL не может возвратить значение NULL в качестве результата. Она всегда возвращает TRUE или FALSE.

**Составные условия поиска**

Все описанные простые условия поиска можно объединить в более сложные с помощью ключевых слов AND, OR, NOT.

Ключевые слова AND, OR, NOT являются операторами логических операций И, ИЛИ, НЕ. С помощью этих операторов и круглых скобок можно создавать очень сложные условия поиска. Например, проверку на членство во множестве NOM IN (1,3,5,9) можно задать другим (менее удачным) условием поиска:

```
(NOM=1) OR (NOM=3) OR (NOM=5) OR (NOM=9)
```

**Сортировка таблицы результатов запроса**

Строки таблицы результатов, как и строки таблицы в БД, не имеют определенного порядка. Чтобы отсортировать таблицу результатов, в оператор SELECT нужно включить предложение ORDER BY.

Например, таблица результатов следующего запроса отсортирована по двум столбцам «Страна» и «КодКлиента»:

```
SELECT КодКлиента, Название, Страна FROM Клиенты
ORDER BY Страна, КодКлиента
```

В такой таблице легко определить клиентов, проживающих в одной стране, причем список таких клиентов упорядочен по их коду.

Первый столбец («Страна») является главным ключом сортировки, следующие за ним столбцы (в данном примере «КодКлиента»), являются всё более второстепенными ключами. Можно сортировать таблицу результатов запроса по любому элементу списка возвращаемых столбцов.

В предложении ORDER BY можно задать возрастающий или убывающий порядок сортировки. По умолчанию, данные сортируются в порядке возрастания (ASC). Чтобы сортировать данные по убыванию, следует использовать ключевое слово DESC. Например:

```
SELECT * FROM Заказано ORDER BY Цена DESC
```

Если столбец таблицы результатов, используемый для сортировки, является вычисляемым, то у него нет имени. Поэтому в таком случае вместо имени столбца необходимо указать его порядковый номер:

```
SELECT АдресПолучателя, СтоимостьДоставки*0.05 AS Скидка
FROM Заказы ORDER BY 2 ASC
```

**Многотабличные запросы**

Язык SQL позволяет получать ответы на многотабличные запросы, которые соединяют данные из нескольких таблиц.

Например, чтобы вывести список заказов с указанием фамилий сотрудников, которые оформляли заказы, необходимо использовать данные, хранящиеся в двух различных таблицах ЗАКАЗЫ и СОТРУДНИКИ, связанных по полю «КодСотрудника». Строки из этих таблиц требуется поместить в одну таблицу результатов следующим образом:

1) каждая строка таблицы результатов образуется сцеплением пары строк: одна строка находится в таблице ЗАКАЗЫ, а другая - в таблице СОТРУДНИКИ;

2) для нахождения пары сцепляемых строк производится сравнение на равенство содержимого полей «КодСотрудника» в этих таблицах.

Процесс формирования строк путем сравнения содержимого соответствующих полей называется соединением таблиц. Соединение таблиц задается в предложении WHERE, в котором условие поиска имеет вид сравнения полей из разных таблиц, или в предложении FROM с использованием операции INNER JOIN.

Для указания принадлежности полей конкретным таблицам используются полные имена полей (столбцов), образуемые из имени таблицы и имени поля, разделенных точкой. В нашем примере нужная таблица результатов формируется одним из следующих операторов:

```
SELECT КодЗаказа, Фамилия, Имя, Должность
FROM Заказы, Сотрудники
WHERE Заказы.КодСотрудника = Сотрудники.КодСотрудника
```

или

```
SELECT КодЗаказа, Фамилия, Имя, Должность
FROM Заказы INNER JOIN Сотрудники
ON Заказы.КодСотрудника = Сотрудники.КодСотрудника
```

Microsoft Access SQL требует обязательного указания полных имен полей в многотабличных запросах для одноименных полей таблиц, используемых в запросе.

В многотабличном запросе вместо списка возвращаемых столбцов может быть использована звездочка *, которая означает включение в таблицу результатов всех полей из всех таблиц, указанных в предложении FROM. В Microsoft Access SQL звездочка используется как особый вид универсального имени поля, которое распространяется на все поля, и заменяет список полных имен полей. В следующем запросе таблица результатов содержит все столбцы из таблицы СОТРУДНИКИ и один столбец из таблицы ЗАКАЗЫ:

```
SELECT КодЗаказа, Сотрудники.*
FROM Заказы, Сотрудники
WHERE Заказы.КодСотрудника = Сотрудники.КодСотрудника
```

**Итоговые запросы на чтение**

Многие запросы к БД требуют подведения итогов по хранящейся в ней информации. На языке SQL запросы такого типа можно создавать с помощью агрегатных функций и предложений GROUP BY и HAVING, используемых в операторе SELECT.

Для подведения итогов в SQL предусмотрены агрегатные функции. Агрегатная функция принимает в качестве аргумента какой-либо столбец данных целиком, а возвращает одно значение, которое определенным образом подытоживает этот столбец. Аргументом агрегатной функции является простое имя столбца или выражение. Например, можно вычислить общую сумму цен заказанных товаров:

```
SELECT SUM(Цена) FROM Заказано
```

В SQL имеется шесть агрегатных функций, которые позволяют получать различные виды итоговой информации:

- SUM( ) вычисляет сумму всех значений, содержащихся в столбце;
- AVG( ) вычисляет среднее среди значений, содержащихся в столбце;
- MIN( ) находит наименьшее среди всех значений, содержащихся в столбце;
- MAX( ) находит наибольшее среди всех значений, содержащихся в столбце;
- COUNT( ) подсчитывает количество значений, содержащихся в столбце;
- COUNT(*) подсчитывает количество строк в таблице результатов запроса.

Агрегатная функция может быть использована для вычисления итогов как по одной таблице базы данных, так и по таблице результатов, полученной соединением нескольких исходных таблиц. Например, можно вычислить общую сумму заказанных товаров, заказы на которые оформлял Новиков Павел:

```
SELECT SUM(Цена)
FROM Сотрудники, Заказы, Заказано
WHERE Сотрудники.КодСотрудника = Заказы.КодСотрудника
      AND Заказы.КодЗаказа = Заказано.КодЗаказа
      AND Фамилия = 'Новиков'
      AND Имя = 'Павел'
```

**Запросы с группировкой**

Наряду с «интегральными» итоговыми запросами, относящимися ко всей таблице результатов в целом, интерес представляют итоговые запросы, в которых агрегатные функции применяются к определенным группам строк. Эту возможность предоставляет предложение GROUP BY оператора SELECT. Например, определить сумму заказанных товаров, заказы на которые оформлял каждый сотрудник, позволяет такой запрос:

```
SELECT Сотрудники.КодСотрудника, Фамилия, Имя, SUM(Цена)
FROM Сотрудники, Заказы, Заказано
WHERE Сотрудники.КодСотрудника = Заказы.КодСотрудника
      AND Заказы.КодЗаказа = Заказано.КодЗаказа
GROUP BY Сотрудники.КодСотрудника, Фамилия, Имя
```

Запрос, включающий в себя предложение GROUP BY, называется запросом с группировкой. Столбцы, указанные в этом предложении, называются столбцами группировки, поскольку именно они определяют, по какому признаку строки объединяются в группы. Ниже приведен ряд запросов с группировкой:

1) сколько клиентов из каждой страны?

```
SELECT Страна, COUNT(*) FROM Клиенты GROUP BY Страна
```

2) сколько заказов оформил каждый сотрудник?

```
SELECT Фамилия, Имя, "оформил", COUNT(КодЗаказа) , "заказов"
FROM Сотрудники, Заказы
WHERE Сотрудники.КодСотрудника = Заказы.КодСотрудника
GROUP BY Фамилия, Имя
```

Предложение GROUP BY видоизменяет действие агрегатных функций, указывая, что результаты запроса следует разделить на группы, применить агрегатную функцию по отдельности к каждой группе и получить для каждой группы одну строку в таблице результатов.

Столбцы группировки, указанные в предложении GROUP BY, должны быть реальными столбцами таблиц, перечисленных в предложении FROM.

Нельзя группировать строки на основании значения вычисляемого выражения.

Следует учитывать также ограничения на элементы списка возвращаемых столбцов. Все элементы этого списка должны иметь одно значение для каждой группы строк. Это означает, что возвращаемым столбцом может быть:

1) константа;
2) агрегатная функция, возвращающая одно значение для всех строк, входящих в группу;
3) столбец группировки, который, по определению, имеет одно и то же значение во всех строках группы;
4) выражение, включающее в себя перечисленные выше элементы.

Точно так же, как предложение WHERE используется для отбора отдельных строк, участвующих в запросе, предложение HAVING можно применить для отбора групп строк. Например, ранее рассмотренный запрос для определения числа заказов, оформленных каждым сотрудником, можно модифицировать с помощью предложения HAVING, чтобы получить информацию только о сотрудниках, оформивших больше 100 заказов:

```
SELECT Фамилия, Имя, "оформил", COUNT(КодЗаказа) , "заказов"
FROM Сотрудники, Заказы
WHERE Сотрудники.КодСотрудника = Заказы.КодСотрудника
GROUP BY Фамилия, Имя
HAVING COUNT(*)>100
```

Запрос выполняется в такой последовательности. Сначала предложение GROUP BY образует группы строк с заказами, оформленными каждым сотрудником. После этого предложение HAVING исключает все группы, в которых количество строк не больше 100. И наконец, предложение SELECT определяет число заказов для каждой из оставшихся групп и генерирует таблицу результатов запроса.

Условие поиска, используемое в предложении HAVING, применяется не к отдельным строкам, а к группе в целом. Это значит, что в условие поиска в качестве операндов могут входить те же элементы, что и в список возвращаемых столбцов запроса с группировкой. На практике условие поиска в предложении HAVING всегда должно включать в себя как минимум одну агрегатную функцию. Если это не так, то условие поиска можно переместить в предложение WHERE.

**Создание SQL-запроса в Access**

Для создания SQL-запроса в Access необходимо выполнить следующую последовательность действий после выбора вкладки «Создание» на ленте:

1) на вкладке выбрать команду «Конструктор запросов»;
2) закрыть окно «Добавление таблицы»;
3) на контекстной вкладке «Конструктор» выбрать команду «Режим SQL» или в нижнем правом углу окна базы данных нажать кнопку «Режим SQL», чтобы задать этот режим в окне запроса;
4) в окне запроса ввести оператор на языке SQL;
5) на панели инструментов нажать кнопку «Выполнить» (кнопка с восклицательным знаком), чтобы отобразить таблицу результатов в окне запроса;
6) исправить ошибки в операторе, если СУБД их обнаружила и не сформировала таблицу результатов, и повторно нажать кнопку «Выполнить»;
7) сохранить запрос, нажав кнопку «Сохранить» на панели быстрого доступа;
8) перейти в режим SQL с помощью команды или кнопки «Режим SQL».

## Лабораторное задание

Для БД, разработанной в предыдущей лабораторной работе, **при домашней подготовке** к сформулировать **не менее 15 запросов** для получения всесторонней информации о предметной области и **записать их на языке SQL**. Среди запросов должны быть:

- итоговые;
- одно- и многотабличные запросы;
- запросы с сортировкой и группировкой;
- должны быть использованы все 5 видов условий поиска.

Проверить соответствие содержимого таблиц результатов формулировке запросов и отметить обнаруженные несоответствия (**результаты занести в Отчет).**

## Требования к сдаче работы

1. Продемонстрировать **отчет** о выполнении лабораторной работы.
2. Ответить на контрольные вопросы и выполнить дополнительные задания, демонстрирующие полученные знания.