Практическая работа №6

Создание SQL запросов на выборку. Параметрические запросы.

Цель занятия: Научиться создавать SQL запросы на выборку, с различными условиями отбора данных. Научиться создавать запросы с параметрами.

Теоретическая часть

Синтаксис оператора SELECT

Назначение оператора ***SELECT*** состоит в выборке и отображении данных одной или нескольких таблиц БД. Этот исключительно мощный, наиболее часто используемый оператор реализует все операции реляционной алгебры. Возможности данного оператора широки и разнообразны. Один и тот же запрос может быть реализован несколькими способами, которые могут существенно отличаться по времени исполнения.

Синтаксис оператора SELECT:

SELECT [DISTINCT| ALL] {*|[<список полей>]} FROM <список таблиц>
[WHERE <предикат-условие выборки или соединения]
[GROUP BY <список полей>]
[HAVING <предикат-условие для группы >]
[ORDER BY <список полей, по которым требуется упорядочить вывод>]

Указанный порядок следования фраз в операторе select не может быть изменен, но не все его части являются обязательными. К обязательным предложениям относятся только фразы select и From. Все остальные части оператора могут быть использованы по усмотрению программиста.

Оператор Select

1. Обычно ***SELECT*** является первой командой SQL-запроса.
2. Между именами полей следует ставить запятые. Порядок имен полей в списке соответствует порядку их обработки и отображения в результирующем наборе данных.
3. Символ "*" определяет очень часто встречаемую ситуацию, когда в результирующий набор включаются все столбцы из исходной таблицы запроса.
4. Имя поля, содержащее пробел или разделитель, необходимо заключать в квадратные скобки.
5. При создании запроса по нескольким таблицам надлежит приводить полную спецификацию поля: Имя_таблицы.Имя_поля.
6. Наличие ключевого слова all (по умолчанию) означает, что в результирующую таблицу включаются все строки, удовлетворяющие условиям запроса, что может привести к появлению в результирующей таблице одинаковых строк.
7. Ключевое слово distinct предназначено для приведения таблицы в соответствие с принципами теории отношений, где предполагается отсутствие дубликатов строк.

Параметр FROM - задается перечень исходных таблиц запроса.

Пример 1. Вывести фамилию, имя, отчество и адрес из таблицы Студенты.
SELECT Фамилия, Имя, Отчество, Адрес FROM Студенты

Пример 2. Вывести ФИО в одном столбце и адрес из таблицы Студенты.
SELECT Фамилия+ Имя+ Отчество as ФИО, Адрес FROM Студенты

Пример 3. Вывести уникальный список Имен из таблицы Студенты.
SELECT DISTINCT Имя FROM Студенты

Параметр WHERE – определяет условия отбора строк.

Параметр WHERE не обязателен, но если он присутствует в инструкции, то должен следовать за параметром FROM. Если параметр WHERE не задан, SQL-запрос выберет все записи.

Параметр WHERE позволяет определить, какие записи таблиц, указанных в списке FROM, появятся в результирующем наборе данных запроса.

ПРИМЕЧАНИЕ: условия отбора строятся так же, как и в расширенном фильтре, смотри практическую работу №4.

Пример 4. Вывести список студентов родившихся в сентябре.
SELECT * FROM Студенты WHERE Month(ДатаРождения) = 9

Параметр GROUP BY

При использовании параметра ***GROUP BY*** все записи, содержащие в заданном поле идентичные значения, объединяются в один элемент выходного набора. Обычно используются с агрегатными функциями. Все имена полей, приведенные в списке ***SELECT***, должны присутствовать и во фразе ***GROUP BY*** - за исключением случаев, когда имя столбца используется в итоговой функции.

Параметр HAVING

В результате объединения записей с помощью параметра GROUP BY и применения параметра HAVING отображаются записи, соответствующие условиям, заданным в параметре HAVING. Это дополнительная возможность фильтрации выходного набора. Используя параметр HAVING, принимайте во внимание следующее:

1. HAVING — необязательный параметр, но если он задан, то должен следовать за параметром GROUP BY.
2. У параметра HAVING те же функции, что и у параметра WHERE, но область их действия ограничена тем, какие записи, сгруппированные посредством параметра GROUP BY, должны отображаться на экране.

Параметр ORDER BY

Посредством параметра ORDER BY выполняется сортировка данных выходного набора в заданной последовательности. Сортировка может осуществляться по нескольким полям, которые в этом случае перечисляются через запятую после ключевого слова ORDER BY. Способ сортировки определяется ключевым словом, которое указывается в рамках параметра ORDER BY и следует за названием поля, по которому сортируются данные.

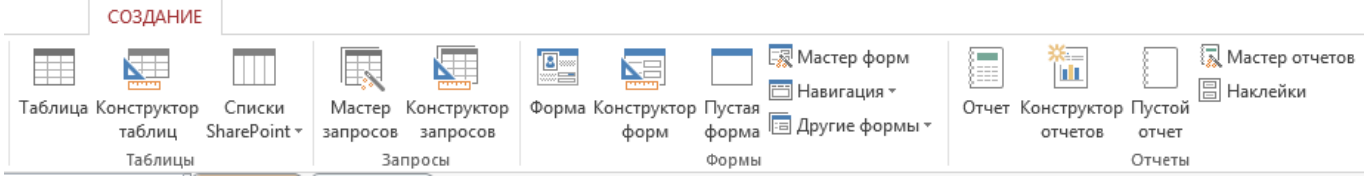
Используя параметр ORDER BY, принимайте во внимание следующее:

1. Параметр ORDER BY не является обязательным параметром; если он не задан, данные не сортируются и приводятся в том порядке, в котором они извлечены из входного набора.
2. По умолчанию выполняется сортировка по возрастанию. Явно она задается ключевым словом ASC.
3. Для выполнения сортировки в обратном порядке (от Я до А) или сортировки по убыванию необходимо после имени поля, по которому сортируются данные, ввести ключевое слово DESC.
4. Параметр ORDER BY обычно является последним элементом SQL- инструкции.

Пример 5. Вывести фамилию, имя, отчество и адрес из таблицы Студенты. Отсортировать по фамилии.
SELECT Фамилия, Имя, Отчество, Адрес FROM Студенты ORDER BY Фамилия

Создание запроса

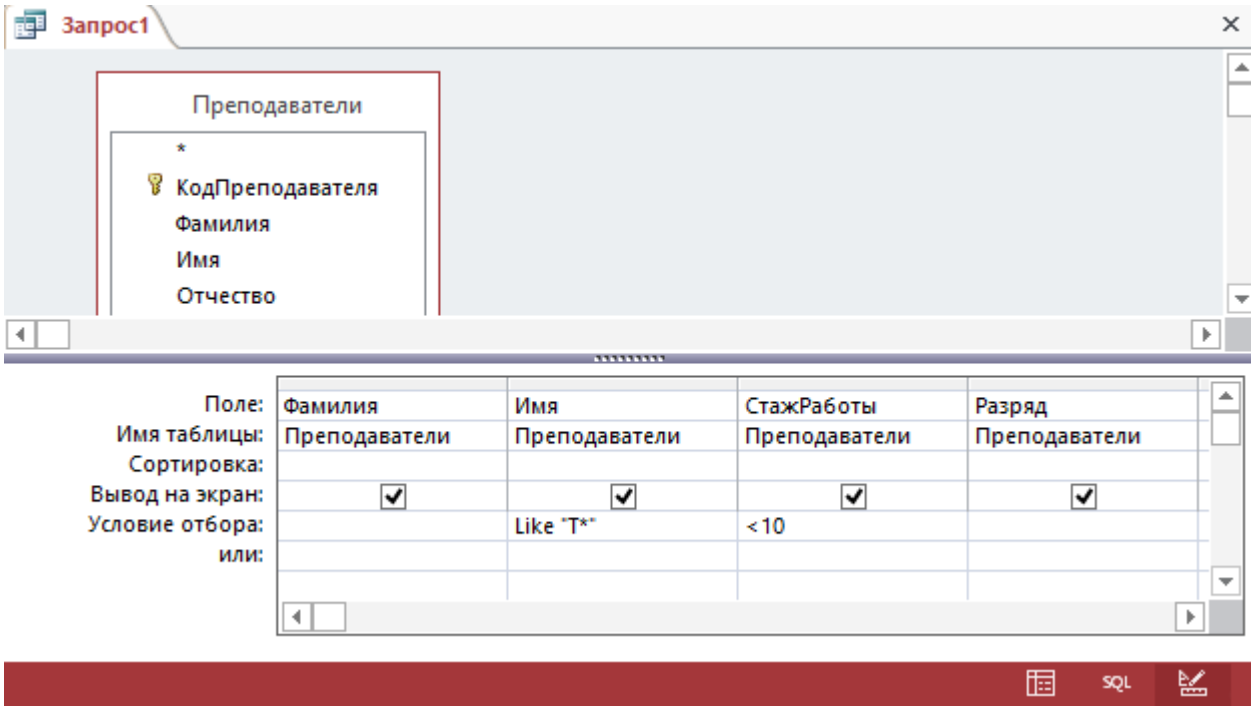
Для создания запроса вы можете использовать — конструктор запросов, который располагается на ленте ***Создание***.



Создать запрос можно следующим образом:

1. На ленте ***Создание***, в разделе ***Запросы*** нажмите кнопку **Конструктор запросов**. Отобразится конструктор запроса с настройки параметров полей.

2. Добавьте таблицы, которые будут использоваться в запросе.
3. Двойным щелчком мыши выберите из таблицы поля, которые будут отображаться при выполнении запроса. Эти поля отобразятся в нижней части конструктора.
4. В строке ***Условия отбора*** укажите условие для нужного поля. Отображаются только записи, которые соответствуют всем условиям в строке **Условия отбора**. Чтобы указать альтернативные условия для отдельного поля, введите первое условие в строке **Условия отбора**, второе условие в строке **или** и т. д.
5. Сохраните полученный запрос и дайте ему имя.



Обратите внимание:

1. Параметр ***Вывод на экран*** обеспечивает вывод или запрет вывода поля при запросе.
2. В нижнем правом углу присутствуют три кнопки удобные для управления создания запроса.

Первая – режим просмотра результатов запроса. Вторая – запрос на языке SQL. Третья – режим конструктора.

Запросы с параметрами

Запросы с параметрами позволяют пользователю задавать условия отбора при каждом запуске. Этот тип запроса не является обособленным, т.е. параметр можно добавить к запросу любого типа. При выполнении такого запроса выводится диалоговое окно, введите значение параметра, в котором пользователь может ввести конкретное значение и затем получить нужный результат. Для определения параметра запроса в строку условие отбора для какого-то столбца вместо конкретного значения вводится слово или фраза заключенные в квадратные скобки, например шаблон параметра названия страны в которую едет клиент имеет вид [введите страну]. Эта фраза будет выводиться в виде приглашения в диалоговом окне каждый раз при выполнении запроса.

Для проверки данных, вводимых в качестве параметра запроса, можно указать тип данных для этого параметра. Его можно установить, выбрав команду ***Параметры*** с ленты ***Конструктор*** при создании запроса. В столбец ***Параметр*** вводится значение параметра точно так, как он определен в бланке запроса, только можно не вводить квадратные скобки, а в столбце ***Тип данных*** выбирается из раскрывающегося списка нужный тип данных.

Допускается создание запросов с несколькими параметрами, для этого необходимо ввести несколько шаблонов в поле условие отбора для разных полей. При выполнении такого запроса для каждого из параметров будут поочередно выводиться диалоговые окна Введите значение параметра в том порядке, в котором они перечислены в бланке запроса.

Пример 6. Вывести список студентов заданного года рождения. Отсортировать по фамилии.
PARAMETERS [Год рождения] Long;
SELECT * FROM Студенты WHERE Year([ДатаРождения])=[Год рождения]
ORDER BY Студенты.Фамилия;

Практическое задание

1. Создайте копию файла БД предыдущей работы и дайте ему имя Работа№6.
2. Удалите элементы фильтра из БД.
3. Выведите список студентов родившихся в 1999 году. Сохраните запрос и полученный список в текстовый файл. Имя запроса – студенты1999.
4. Выведите список студентов, у которых мобильный телефон с кодом 909. Сохраните запрос и полученный список в текстовый файл. Имя запроса – телефон909.
5. Выведите список предметов специальности «Программирование в компьютерных системах». Отсортируйте в обратном алфавитном порядке. Сохраните запрос и полученный список в текстовый файл. Имя запроса – ПредметыПрогр.
6. Выведите список бюджетных групп 3 курса. Сохраните запрос и полученный список в текстовый файл. Имя запроса – БюджетГруппы.
7. Выведите список сданных зачетов заданного студента. Сохраните запрос и полученный список в текстовый файл. Имя запроса – ЗачетыСтудента.
8. Выведите список преподавателей высшей категории с заданным стажем. Отсортируйте по алфавиту. Сохраните запрос и полученный список в текстовый файл. Имя запроса – ВысшКатегория.
9. Выведите список студентов родившихся в мае и сентябре. Сохраните запрос и полученный список в текстовый файл. Имя запроса – РождМайСентябрь.
10. Выведите уникальный список мест работы родителей. Сохраните запрос и полученный список в текстовый файл. Имя запроса – МестаРабот.
11. Выведите список экзаменов проведенных в заданный день. Сохраните запрос и полученный список в текстовый файл. Имя запроса – ЭзаменыДень.
12. Оформите отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите запрос на выборку