Практическая работа 49
Создание и использование запросов. Группировка и агрегирование данных. Создание в запросах вычислимых полей. Использование условий.

Цель работы: Получить практический опыт создания и использование запросов.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)

Краткие теоретические сведения

Команда SELECT

Команда SELECT - наиболее часто употребляемая команда из всех восьми. Она используется для выборки данных из базы данных. Её синтаксис:

SELECT [Предикат] Поля FROM Таблицы [IN БазаДанных] [WHERE ...] [GROUP BY ...] [HAVING ...] [ORDER BY ...];

Необязательные аргументы заключены в [].

› Предикат - одно из четырёх слов ALL, DISTINCT, DISTINCTROW, TOP. Если предикат не указан, то устанавливается ALL. Предикат ALL позволяет отобрать все записи. При использовании предиката DISTINCT, записи, которые содержат повторяющиеся значения в выбранных в запросе полях, исключаются. Предикат DISTINCTROW исключает из выборки записи, если повторяется вся запись, а не одно из полей. Предикат TOP позволяет отобрать определённое количество записей.

- Поля - имена одного или нескольких полей, выборка которых производится. Для выборки всех полей вместо имен полей можно поставить звёздочку [*].
- Таблицы - имена одной или нескольких таблиц, из которых производится выборка.
- БазаДанных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.

Также возможно использование предложения WHERE вместе с операторами BETWEEN, IN и LIKE.

Оператор BETWEEN позволяет отобрать записи, значение определённого поля которых находится в заданном диапазоне. Например:

SELECT * FROM Orders WHERE ID BETWEEN 10 AND 20;

Здесь выбираются все записи, значение поля ID которых находится между 10 и 20.

Оператор IN позволяет отобрать записи, значение поля которых соответствует одному из значений, указанных в скобках.

SELECT * FROM Orders WHERE ID IN (10, 12, 30, 45);

Здесь отбираются все записи, значение поля ID которых соответствует одному из значений 10, 12, 30, 45.

Используя предложение WHERE совместно с оператором LIKE, возможен отбор записей, значение одного из полей которых совпадает с маской. Оператор LIKE применим только к текстовым полям. В маске можно использовать следующие символы:

Символ Описание

Подчёркивание [_] Замещает один любой символ.

Процент [%] Замещает последовательность любого числа символов.

Например:

SELECT * FROM Orders WHERE Name LIKE 'Ba_я%'

Здесь выбираются все записи, поле Name которых соответствует маске Ba_я%. Обратите внимание, что значения текстового типа в SQL-запросах указываются в кавычках.

Предложение GROUP BY позволяет объединять поля в запросе.

Предложение ORDER BY позволяет упорядочивать выбираемые записи. При использовании совместно с предложением ключевого слова ASC можно определить возрастающий порядок, а используя DESC, определяется убывающий порядок.

SELECT * FROM Orders ORDER BY NameASC;

Также можно упорядочивать записи по нескольким полям. Сначала записи упорядочиваются по первому полю, если в нём есть записи, имеющие одинаковые значения, то они упорядочиваются по следующему указанному в предложении ORDER BY полю и т.д. Имена полей пишутся через запятую [,].

SELECT * FROM Orders ORDER BY NameASC, EmailASC;

Вставка записей

Синтаксис:

INSERT INTO <имя_таблицы>[(<имя_столбца>,<имя_столбца>,...)]

VALUES (<значение>,<значение>,..)

Список столбцов в данной команде не является обязательным параметром. В этом случае должны быть указаны значения для всех полей таблицы в том порядке, как эти столбцы были перечислены в команде CREATE TABLE

Команда INSERT INTO предназначена для добавления одной или нескольких записей в конец таблицы. Возможны 2 варианта использования

этой команды. Первый вариант добавляет одну запись в таблицу, а второй вариант добавляет записи из одной таблицы в другую.

Синтаксис первого варианта:

INSERT INTO Таблица Назначения [(Поля)] VALUES (Значения);

Синтаксис второго варианта:

INSERT INTO Таблица Назначения [(Поля)] [IN База Данных] SELECT [Таблица.] Поля FROM Таблица;

- Таблица Назначения - таблица, в которую добавляются записи.
- Поля - названия полей.
- Таблица - имя таблицы, источника данных.
- База Данных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.
- Значения - значения полей добавляемой записи.

Все поля записи и соответствующие им значения должны быть определены, иначе им будут присвоены значения Null.

Если таблица, в которую добавляются записи, имеет ключевое поле, то в него должны добавляться уникальные, непустые значения. Иначе запись не будет добавлена.

Пример:

INSERT INTO Orders (ID, Name, Email, Order) VALUES (12, 'Вася Пупкин', ' vasya@pupkin.ru', 'PentiumII 450 MHz');

Добавляется новая запись, в которой полям ID, Name, Email, Order соответствуют значения 12, 'Вася Пупкин', ' vasya@pupkin.ru', 'Pentium II 450 MHz'.

INSERT INTO Orders2001 (ID, Name, Email, Order) SELECT ID, Name, Email, Order FROM Orders2000;

Этот запрос добавляет все записи из таблицы Orders2000 в таблицу Orders2001.

Модификация записей
Синтаксис:

UPDATE <имя_таблицы> SET <имя_столбца>=<значение>,...
[WHERE <условие>]

Если задано ключевое слово WHERE и условие, то команда UPDATE применяется только к тем записям, для которых оно выполняется. Если условие не задано, UPDATE применяется ко всем записям.

Команду UPDATE удобно использовать, если изменяется сразу большое число записей или если изменяемые записи находятся в разных таблицах. Новые значения указываются через запятую для каждого поля. Использование предложения WHERE аналогично его использованию в команде SELECT.

Пример:

UPDATE Buyers SET Order='Ничего' WHERE ID=7;

Устанавливаем значение поля покупки 'Ничего' у покупателя, номер которого равен 7.

UPDATE Заказы SET СуммаЗаказа = СуммаЗаказа * 1.2, СтоимостьДоставки = СтоимостьДоставки * 1.1 WHEREСтрана='США';

Этот запрос немного сложнее. Он повышает сумму заказа на 20% и стоимость доставки на 10% для покупателей из США.

Удаление записей
Синтаксис:

DELETE FROM<имя_таблицы>[WHERE<условие>]

Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отсутствуют, из таблицы удаляются все записи.

Аргумент команды DELETE можно не указывать, поскольку он фактически дублируется в предложении FROM.

Пример:

DELETE FROM Buyers WHERE ID=8;

Этот запрос удаляет из таблицы Buyers запись, в которой ID равно 8.

Для удаления не всей записи, а только ее поля, следует воспользоваться запросом на изменение записи (команда UPDATE) и поменять значения нужных полей на Null.

Перекрестный запрос

Перекрестный запрос это соединение нескольких таблиц в одной. Одна из таблиц используется для заголовков строк, вторая – для заголовков столбцов, а данные для создаваемой таблицы могут храниться в третьей. Значит, необходимо команда соединения таблиц JOIN. Эта команда соединяет две таблицы, а если необходимо соединить три, воспользуемся вложенными запросами. Кроме того, ключевым словом SQL-оператора перекрестного запроса, задающим его тип, является слово **TRANSFORM** (преобразовать). Это подразумевает, что значения одного из столбцов (полей) выборки, будут преобразованы в названия столбцов итоговой выборки.

Результаты перекрестного запроса группируются по двум наборам данных, один из которых расположен в левом столбце (столбцах) таблицы, а второй — в верхней строке. В остальном пространстве таблицы отображаются результаты статистических расчетов (Sum, Count и т.д.), выполненных над данными трансформированного поля. И наконец, **PIVOT** – это оператор, который поворачивает результирующий набор данных, т.е. происходит транспонирование таблицы, при этом используются агрегатные функции, и данные соответственно группируются. Другими словами, значения, которые расположены по вертикали, мы выстраиваем по горизонтали. Но оператор PIVOT можно использовать, только начиная с 2005 sql сервера.

У данного оператора очень специфический, непривычный и некоторые даже скажут сложный синтаксис, как для написания, так и для простого понимания.

Синтаксис оператора PIVOT

SELECT *столбец для группировки, [значения по горизонтали], …*
FROM *таблица или подзапрос*
PIVOT(*агрегатная функция*
FOR *столбец, содержащий значения, которые станут именами столбцов*
IN (*[значения по горизонтали], …*)
)AS *псевдоним таблицы (обязательно)*
в случае необходимости ORDER BY;

Инструкция TRANSFORM используется для создания перекрестного запроса. Данные, представленные с помощью перекрестного запроса, изображаются в более компактном виде, чем с помощью запроса-выборки.

Синтаксис:

TRANSFORM Функция
SELECT …;
PIVOT поле;

Функция - групповая функция SQL, обрабатывающая данные ячейки таблицы

Поле - поле или выражение, значения из которого становятся заголовками столбцов.

Запрос в режиме таблицы имеет столько столбцов, сколько различных значений принимает поле. Например, если поле выдает названия месяцев, то получится до 12 столбцов, заголовки которых упорядочены по возрастанию (Август, Апрель…Январь). После аргумента поле можно поместить предложение IN(список_значений). Фиксированные значения в списке_значений разделяются запятыми. При наличии предложения IN каждое значение поля сравнивается со значениями в списке_значений. При совпадении в соответствующем столбце выводится результат вычисления функции. Фиксированные заголовки, которым не соответствуют реальные данные, можно использовать для создания дополнительных столбцов.

Использование предложения PIVOT эквивалентно определению свойства “Заголовки столбцов” в бланке свойств конструктора запросов.

Пример

```
TRANSFORM Count([Оценка])
SELECT Группа, Count([Предмет]) AS Всего FROM Экзамены
WHERE [Предмет]="1" OR [Предмет]="2"
GROUP BY Группа
ORDER BY Группа
PIVOT Оценка IN(5,4,3,2);
```

Результат выполнения может иметь, например, такой вид:

Группа	Всего	5	4	3	2
1	4	1		2	1
2	2		1	1	

Столбцы “Группа” и “Всего” сформированы инструкцией SELECT, включающей в себя предложения WHERE, FROM, GROUP BY и ORDER BY. Заголовки остальных столбцов определены предложением PIVOT, а значения в ячейках этих столбцов формирует функция Count из предложения TRANSFORM.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания организовать выборки данных с помощью запросов:

- на добавление данных с помощью запроса и из другой таблицы;
- на обновление таблиц;

– перекрестный;
– на удаление.

Порядок выполнения работы

1 Запрос на добавление данных

Ввод данных в таблицу производится с помощью команды INSERT INTO непосредственным вводом данных (в Access для каждой строки таблицы отдельный запрос):

INSERT INTO Преподаватели ([Фамилия преп],[Имя преп],[Отчество преп], [Код преп])
VALUES (‘Хлестаков’, ‘Иван’, ‘Александрович’,4),

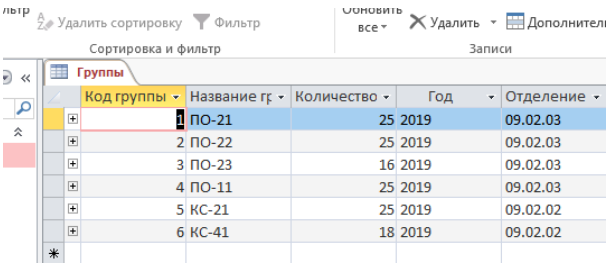
(‘Уховертов’, ‘Степан’, ‘Ильич’,5);

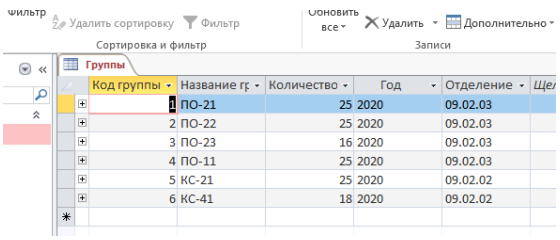
Или из другой таблицы (все записи из таблицы Преподаватели1 будут перенесены в таблицу Преподаватели):

INSERT INTO Преподаватели ([Фамилия преп],[Имя преп],[Отчество преп], [Код преп])
SELECT * FROM Преподаватели1;

2 Запрос на обновление выполняется с использованием команды UPDATE. Пример замены данных в одном поле:

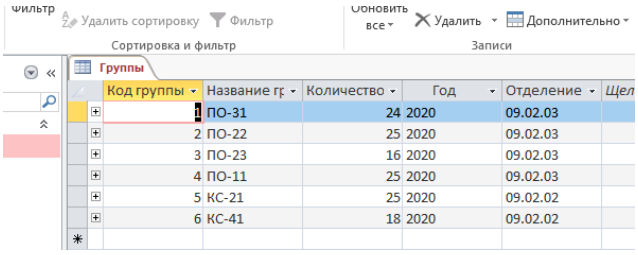
UPDATE Группы SET Год=2020;





Пример обновления нескольких полей с условием:

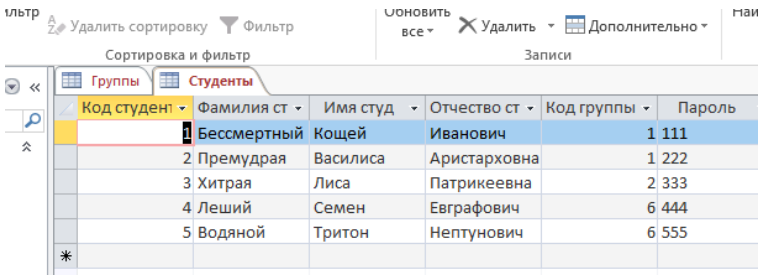
UPDATE Группы SET [Количество человек]=24, [Название группы]=”ПО-31”
WHERE [Название группы]=”ПО-21”



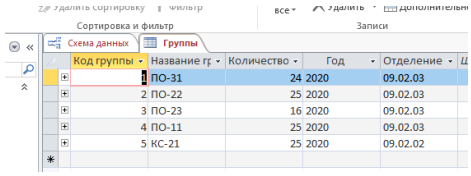
3 Запрос на удаление выполняется с использованием команды DELETE FROM. Пример удаления данных о группе КС-41 в 2019 году. В 2020 эта группа уже окончила обучение, поэтому ее надо удалить:

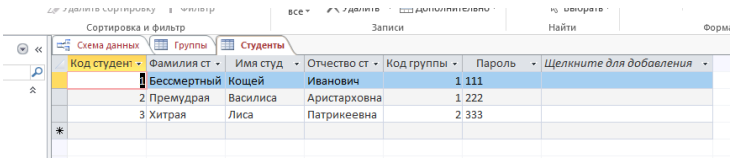
DELETE FROM Группы WHERE [Название группы]=”КС-41”;

Но таблица Группы связана с таблицей Студенты, в которой имеются записи студентов группы КС-41:



Если при организации связи таблиц не установлено каскадное обновление полей и каскадное удаление записей, то будет выдано сообщение об ошибке, иначе будут удалены записи в двух таблицах:





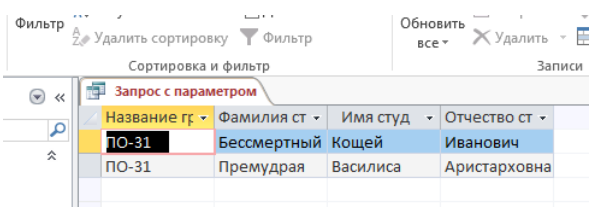
4 Запрос с параметром выполняется с использованием команды SELECT. В примере выборка фамилий студентов из заданной с помощью параметра группы. Так как название группы в таблице Группы, а ФИО студентов в таблице Студенты, то используем условие соответствия кода группы в двух таблицах вместе с условием проверки параметра:

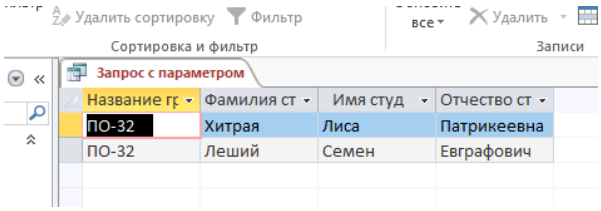
SELECT Группы.[Название группы], [Фамилия студ], [Имя студ], [Отчество студ]
FROM Группы, Студенты
WHERE Группы.[Название группы] = [Задайте группу]
AND Группы.[Код группы]=Студенты.[Код группы];

Или

```
SELECT Группы.[Название группы], [Фамилия студ], [Имя студ], [Отчество студ]
FROM Группы
JOIN Студенты
ON Группы.[Код группы]=Студенты.[Код группы]
WHERE Группы.[Название группы] = [Задайте группу];
```

В результате:

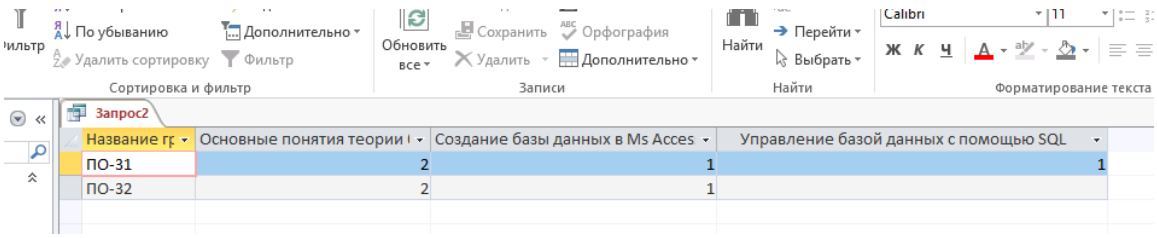




5 Перекрестный запрос, вычисляющий количество тестов в базе данных тестовой оболочки, начнем со слова **TRANSFORM**, далее следует агрегатная функция **Count**, вычисляющая количество тестов, запланированных по каждой теме для каждой группы, далее объединим три таблицы в операторе **SELECT** с помощью команды **INNER JOIN** с вложенным запросом. И наконец, в **PIVOT** укажем поле, которое станет заголовком столбцов таблицы перекрестного запроса.

```
TRANSFORM Count([Проведение теста].[Код проведения]) AS Всего
SELECT Группы.[Название группы]
FROM Темы INNER JOIN (Группы INNER JOIN [Проведение теста] ON Группы.[Код группы] = [Проведение теста].[Код группы]) ON Темы.[Код темы] = [Проведение теста].[Код темы]
GROUP BY Группы.[Название группы]
PIVOT Темы.[Название темы];
```

И в результате:



Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения

4 Порядок выполнения работы
5 Ответы на контрольные вопросы
6 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 39

Контрольные вопросы:

1. Для чего используется язык манипулирования данными?
2. Основные команды языка DML?
3. К каким объектам применяются команды языка DML?
4. Какие действия с данными выполняет команда SELECT?
5. Для чего служит команда DELETE FROM?
6. Что произойдет с данными после выполнения команды UPDATE?
7. Какие действия с данными выполняет команда INSERT INTO?

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020

– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018

– https://infopedia.su/16x10105.html
– https://studfile.net/preview/5917121/page:2
– https://codetown.ru/category/sql
– https://oracleplsql.ru/joins-sql.html
– https://life-prog.ru/2_45010_TRANSFORM.html