МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА

Кафедра "Информационные технологии"

Ю. П. ЛЫЧ

ТЕХНОЛОГИИ ОРГАНИЗАЦИИ, ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ

Часть 2: Основы программирования в СУБД ACCESS

Пособие для самостоятельной подготовки к занятиям

Рекомендовано советом гуманитарно-экономического факультета
Белорусского государственного университета транспорта
в качестве пособия для студентов
экономических специальностей вузов

Гомель 2003

УДК
Л889

Рецензент – **В. С. Серегина**, кандидат физико-математических наук, доцент, заведующая кафедрой "Прикладная математика" БелГУТа.

Лыч Ю. П.

Л889 Технологии организации, хранения и обработки данных. Часть 2. Основы программирования в СУБД Access: Пособие для самостоятельной подготовки к занятиям. – Гомель: БелГУТ, 2003. – 70 с.

Вторая часть пособия посвящена структурированному языку запросов SQL (назначение, основные понятия, необходимые для изучения SQL, применение его на практике) и основам программирования на языке Visual Basic for Applications (VBA).

Пособие адресовано студентам экономических специальностей высших учебных заведений, а также студентам других специальностей, изучающих основы программирование в среде Microsoft Access.

УДК

ISBN 985-6550-24-6

1 ВВЕДЕНИЕ В ЯЗЫК SQL

1.1 Назначение языка SQL

Программирование в условиях современных информационных технологий, в частности в среде MS Office, получило адекватное развитие. В классическом смысле продуктом программирования была программа, которую составляли для решения задачи, получения соответствующего документа. Сегодня в среде MS Office программирование есть процесс создания документов. Первичным становится документ, а программа — это его часть. Цель программиста — не создание программы, а формирование конкретного выходного документа с широким спектром функций. Пользователь работает не с программой, а с документом.

Изменился смысл термина *«документ»*. Документ рассматривается теперь как объект в объектно-ориентированном программировании (единство данных различного типа и программ, которые их обрабатывают).

Процесс создания документа в среде MS Office называют «офисным программированием» (программированием без программирования). Документ становится «живым», т. е. с данными, которые в нем зафиксированы, можно работать, их можно сменять, анализировать и т. д.

Документ и его программные компоненты объединяются в единое целое — *проект*, который является частью документа и не существует отдельно от него.

Язык SQL — Structured Query Language (структурированный язык запросов) разработан фирмой IBM в начале 70-х гг. Это современное средство для работы с базами данных, которое применяется в среде реляционных баз данных (создание, поиск, изменения, обновления, передачи данных).

Язык SQL утвержден Американским национальным институтом стандартов (ANSI) и Международной организацией стандартов (ISO) в качестве официального стандарта для реляционных баз данных и не зависит от специфики компьютера.

1.2 Структура команд языка SQL

Язык SQL состоит из команд (инструкций). Команды передаются программе, которая управляет работой базы данных, для выполнения конкретных действий. Объектом действия команды SQL может быть как одна таблица, так и группа таблиц. MS Access автоматически создает эквивалентные команды SQL во время создания запроса в режиме Конструктора.

Команды SQL (инструкции, операторы) делят на две группы:

- *язык определения данных* (Data Definition Language — DDL):
  Create Table — создает новую таблицу;
  Create Index — создает новый индекс;
  Alter Table — дополняет новое поле или индекс в существующую таблицу;
- *язык манипулирования данными* (Data Manipulation Language — DML):
  Select — запрос к базе данных; запрос возвращает данные в виде набора записей;
  Select Into — запрос к базе данных; запрос возвращает данные в виде новой таблицы;
  Update — запрос на объединение; запрос изменяет значения полей таблицы;
  Insert Into — запрос на добавление в таблицу одной или нескольких записей;
  Delete — запрос на удаление записей с одной или нескольких таблиц.

Существуют две формы языка SQL: интерактивная и встроенная.

Интерактивный SQL применяется непосредственно в базе данных для выполнения определенных действий над данными. Вводятся определенные команды, после выполнения которых тут же выводятся выходные данные (результат).

Встроенный SQL — это включение команд языка SQL в программы, которые написаны на другом языке программирования, например Pascal.

В данной книге описывается интерактивный SQL, наиболее удобный и целесообразный для экономистов-непрограммистов.

Каждая команда включает в себя соответствующий перечень параметров (средств) для выбора, группировки и упорядочения записей из одной или нескольких таблиц. Любая команда имеет свой формат (синтаксис).

Основные термины

Ключевое слово — это инструкция; слово, которое имеет специальное смысловое значение в SQL. В тексте книги ключевые слова выделены прописными буквами.

Команда — это инструкция, которая дается базе данных SQL. Команда состоит из одной или нескольких логически различных частей, называемых

предложениями. Предложения начинаются с ключевого слова и кроме него содержат аргументы, например:

WHERE fio= ‘Астрожский’,

где fio = ‘Астрожский’ — аргумент; WHERE — ключевое слово.

Объекты — это структуры в базе данных (таблицы, запросы, формы и др.), которые именуются и хранятся в памяти.

Соглашения по синтаксису команд:

[ ] — квадратные скобки — часть команды, которую при желании можно опустить;

() — круглые скобки — предшествующее им можно повторить любое количество раз;

< > — слова, заключенные в угловые скобки, — специальные термины, которые объясняются по мере вывода.

Ядром (главной командой) языка SQL является команда SELECT. Эта команда находит таблицу или несколько таблиц в базе данных, которая указана в ее параметрах, выбирает заданные столбцы, выделяет строки в соответствии с условиями отбора, сортирует и группирует строки результата в заданном порядке.

Выполнение команды SELECT не изменяет данные в базе данных. Выбираемые столбцы с таблицы не удаляются, из них только извлекаются данные.

Синтаксис (формат) команды SELECT:

SELECT [предикат] {* [таблица.* | [таблица.] поле_1
[AS псевдоним_1] [,[таблица.] поле_2 [AS псевдоним_2] [,...]]}
FROM выражение [,.,.] [IN Внешняя_База_данных]
[WHERE.. условие_Отбора]
[GROUP BY.. список_полей_группировки]
[HAVING условие группировки]
[ORDER BY.. вполе J [ASC | DESC]. [, вполе_2 [ASC | DESC]]...]

«Предикат» — задает ограничения на возвращаемые записи: ALL — по умолчанию, т. е. без ограничений; DISTINCT — все записи без их дублирования;

«Таблица» - имя таблицы-источника, из которой берутся записи;

«поле_1», «поле_2»... — имена полей, из которых берутся записи;

«псевдоним_1», «псевдоним_2»... — имена, которые станут заголовками столбцов вместо исходных названий столбцов в таблице;

«выражение» — имена одной или нескольких таблиц, из которых берутся данные;

«Внешняя_База_Данных» — имя базы данных с таблицами, которые указаны с помощью аргумента «выражение», если они не находятся в текущей базе данных;

«условие_Отбора» — выражение с условием, которому должны соответствовать записи, включенные в результат исполняемого запроса;

«список_полей_группировки» — имена полей (до 10), которые применяются для группирования записей;

«условие_Группировки» — выражение, которое определяет, какие сгруппированные записи надлежит отображать;

«вполе_1, вполе_2» — поля, по значениям которых сортируются записи в результате запроса;

ASC— по возрастанию, *DESC*— по убыванию;

FROM— обязательный параметр.

Пример. Предположим, из базы данных (таблица Spisok) надо составить список всех студентов. Список должен иметь следующий вид:

Фамилия	Курс	Группа

SELECT fio, kurs, grupa
FROM Spisok;

По этой команде выводятся заголовки столбцов (фамилия, курс, группа) и все данные (значения полей fio, kurs, grupa) будут выведены из таблицы Spisok.

1.3 Описание таблиц

Объекты SQL создаются с помощью языка определения данных (Data Definition Language — DDL), который применяется для описания атрибутов базы данных, таблиц, полей, индексов и способов сохранения данных.

Для создания таблицы, описания ее структуры предназначена команда **CREATE TABLE.** По этой команде:

- создается пустая таблица;
- присваивается имя пустой таблице;
- присваивается имя столбцам (полям) и определяется порядок их следования;
- устанавливается тип и размер каждого поля.

Минимальное число столбцов в таблице — один.

Формат команды **CREATE TABLE:**

CREATE TABLE <имя таблицы >
(<имя столбца> <тип данных> [(<размер>)],
<имя столбца> <тип данных> [(<размер >)]...);

Для данных символьного типа **«размер»** указывать обязательно, так как по умолчанию это 1 (один символ).

Пример команды CREATE TABLE для создания таблицы Spisok следующей структуры:

Факультет	Курс	Группа	ФИО	Стипендия (сумма)	Удержания

Перед созданием таблицы базы данных необходимо:

1) создать структуру таблицы, т. е. определить необходимый перечень полей и отношений между ними;
2) присвоить имена выбранным полям;
3) определить тип каждого поля (символьное, числовое, логическое и т. д.);
4) задать размер полей.

После этого приступаем к созданию таблицы. Для каждого поля структуры файла следует указать имя поля, его тип, длину, а для числовых данных — и количество цифр после десятичной точки, если это необходимо:

CREATE TABLE Spisok
fakultet char (4),
kurs char(l),
grupa char (5),
fio char (15),
stip decimal (7,2),
uderzano decimal (6,2);

Поля **Stip и Uderzano** описаны как десятичные, соответственно длина поля (field width) 7 и 6 десятичных цифр и точность (количество знаков после запятой в цифровых полях) — 2.

Эту команду можно записать и в строку:

CREATE TABLE Spisok
(fakultet char (4), **kurs** char (1), **grupa** char (4), fio char (15), **slip** decimal (7,2), **uderzano** decimal (6,2);

Контрольный пример таблицы Spisok приведен на рисунке 1.1.

Spisok : таблица

fakultet	kurs	grupa	fio	stip	uderzana
ГЭФ	1	ГК-11	Романова	25000	250
ГЭФ	1	ГК-11	Уткина	26000	260
ГЭФ	1	ГК-11	Пшеничко	25000	250
ГЭФ	1	ГК-12	Титов	25000	250
ГЭФ	1	ГК-13	Жичко	25000	250
	0			0	0

Запись: 6 из 6

Рисунок 1.1 – Контрольный пример таблицы Spisok

Индексирование

Записи в файлах базы данных размещены по ключам упорядочения. Возможны два вида упорядочения записей в таблице: а) логическое (CREATE INDEX — индексирование); б) физическое (SORT — сортирование).

Ключи сортирования должны быть сравнимыми, т. е. любые два ключа сортирования К1 и К2 должны удовлетворять одному из трех отношений:

К1 < К2, К1 = К2, К1 >К2.

Ключ, на который имеется ссылка в другой таблице, называется *внешним*.

Для манипулирования со значением строк таблицы предназначены *индексы*. Индексирование — это упорядочение записей по ключу (алфавиту, хронологии, в порядке возрастания или убывания). Для индексного поля создается упорядоченный *список значений* для этого поля. В таблице данных строки не упорядочены. Для поиска строки с заданным значением поля-ключа программа последовательно просматривает все записи таблицы, строка за строкой, пока не встретит строку с заданным значением поля. Это долгий путь. Индекс же сразу находит запись по значению поля-ключа.

Индекс (индексный файл) создается по команде:

CREATE INDEX <имя индекса> ON <имя таблицы>
(<имя столбца>[,<имя столбца>]...);

Пример. Создать индекс по полю *fio* таблицы Spisok:

CREATE INDEX fio ON Spisok (fio);

Таблица индексов (индексный файл), созданная командой CREATE INDEX, для пользователя невидима. SQL сам автоматически обращается к таблице индексов по мере надобности.

Корректировка таблицы (добавление столбцов в таблицу, удаление столбцов, изменение их размера и др.) выполняется командой **ALTER TABLE** следующего формата:

ALTER TABLE <имя таблицы> ADD <имя столбца>
<тип данных>,<размер>;

Новый столбец по этой команде в таблице становится последним; в него заносятся NULL-значения. Пользоваться этой командой следует осмотрительно, чтобы не повредить базу данных.

Удаление таблицы. Удалить можно только пустую таблицу, поэтому предварительно следует удалить ее данные. Формат команды:

DROP TABLE <имя таблицы>;

1.4 Манипулирование данными

Для управления данными в таблице и их манипуляцией предназначены команды DML (Data Manipulation Language): **INSERT** (вставка), **UPDATE** (обновление), **DELETE** (удалить). С помощью этих команд данные заносятся в поля и исключаются из них.

Ввод в таблицу значений полей

Команда добавления данных INSERT используется для вставки содержимого одной или нескольких новых строк в указанную таблицу или запрос. Упрощенный формат команды:

INSERT INTO <имя таблицы >
VALUES (<значение>, <значение>...);

Имя таблицы в команде **INSERT** должно быть определено до выполнения команды INSERT в команде **CREATE TABLE**. Значения в списке значений (<значение>...) должны иметь тип данных, соответствующий типу данных столбцов таблицы. Значения вводятся в таблицу в порядке следования столбцов. Программа сообщает пользователю о добавлении записи.

Пример. Ввести запись в таблицу Spisok:

INSERT INTO Spisok
VALUES ('ГЭФ', 1, 'ГК-11', 'Петькин', 25000, 400);

Если значение какого-либо поля неизвестно, то в списке значений можно вставлять NULL-значение. Предположим, неизвестен размер стипендии:

VALUES 'ГЭФ', 1, 'ГК-11', 'Петькин', NULL, 400);

Вставка результатов запроса (команда **INSERT).**

С помощью команды INSERT можно по запросу извлечь значения из одной таблицы и разместить их в другой; Для этого в команде INSERT предложение VALUES заменяется на SELECT. Столбцы таблиц должны быть одного типа данных.

Пример. Все строки таблицы Spisok 1-го курса расположить в таблице Spisok1;

INSERT INTO Spisok1
FROM Spisok
WHERE kurs = 1;

Изменение значения полей (команда **UPDATE**).

Командой **UPDATE** можно изменить в строке некоторые или все значения. В команде указываются имя таблицы и изменения. Предположим, приказом ректора всем студентам, которые получают стипендию, устанавливается новый (одинаковый для всех) размер стипендии в размере 50 000 руб. Для этого подается команда:

UPDATE Spisok
SET stip = 50 000;

В команде **UPDATE** в предложении **WHERE** можно задать обновление только определенных строк.

Пример.

UPDATE Spisok
SET stip = 50 000
WHERE kurs=I;

В предложении SET команды **UPDATE** можно через запятую указать любое количество значений для столбцов.

Одной командой **UPDATE** нельзя обновить несколько таблиц.

В команде **UPDATE** в предложении SET можно применять скалярные выражения для изменения значения поля. Предположим, всем студентам размер стипендии увеличен на 25 %; требуется внести изменения:

UPDATE Spisok
SET stip = (stip *25)/100;

Итак, команда **UPDATE** предназначена для замены значений в строках (записях) таблицы.

Удаление строк из таблицы (команда **DELETE).**

Командой **DELETE** удаляются не отдельные значения полей строки, а целые строки. После выполнения команды **DELETE** для всей таблицы она становится пустой, например:

DELETE FROM Spisok;

Для удаления конкретно указанных строк можно использовать предикат

DELETE FROM Spisok
WHERE stip = 0;

или указать значение первичного ключа для удаления одной записи

DELETE FROM Spisok
WHERE fio = ‘Петькин’;

В командах **INSERT, DELETE, UPDATE** можно применять подзапросы.

1.5 Формирование запросов (команда SELECT)

Язык SQL есть структурированный язык запросов. Запрос в этой среде — это команда, с помощью которой пользователь формулирует задачу для СУБД. После выполнения команды СУБД должна представить указанную в запросе информацию для пользователя.

Все запросы в SQL формулируются с помощью одной команды **SELECT**, после которой в базе данных начинается поиск определенной (нужной) информации в таблице.

Пример. На основании таблицы Spisok получить таблицу со всеми записями следующего вида:

Фамилия	Стипендия

Подается команда:

SELECT fio, stip **FROM** Spisok;

Результат выполнения команды приведен на рисунке 1.2.

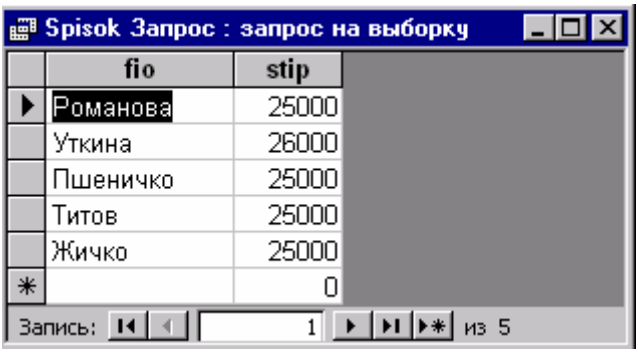

Рисунок 1.2 – Команда SELECT

Поясним команду в приведенном примере:

SELECT— ключевое слово, которое «сообщает» СУБД о том, что команда является запросом;

fio, stip — список имен полей (столбцов), по которым должна выбираться информация и формироваться новая таблица;

FROM Spisok; FROM— ключевое слово, должно быть в каждом запросе; Spisok — имя таблицы-источника данных для запроса;

символ точка с запятой (;) — признак окончания команды и готовности к ее выполнению.

После ключевого слова **SELECT** следует пробел. Далее через запятую перечисляются имена полей (столбцов) выборки.

Для вывода всех столбцов таблицы базы данных список полей можно не перечислять, заменив его символом «звездочка» (*). Столбцы выводятся в соответствии со структурой таблицы-источника

SELECT * FROM Spisok;

Командой SELECT можно выводить столбцы в любой последовательности, отличной от упорядоченной по определению структуры таблицы-источника. Эта последовательность задается перечнем имен столбцов в команде SELECT. Пример переупорядоченных столбцов в выходной таблице

SELECT kurs, grupa, fio **FROM** Spisok;

Выбор по критерию

С помощью предложения **WHERE** в команде **SELECT** задается условие выбора записей из таблицы; предикат может принимать значение «истина» или «ложь».

Пример. Команда **SELECT** для выбора из таблицы Spisok студентов группы ГК-11:

SELECT grupa, fio FROM Spisok WHERE grupa = ‘ГК-11’;

Программа просмотрит все записи таблицы Spisok, проверяя каждую из них, на истинность предиката grupa = ‘ГК-11’. Результат выполнения этого запроса приведен на рисунке 1.3.

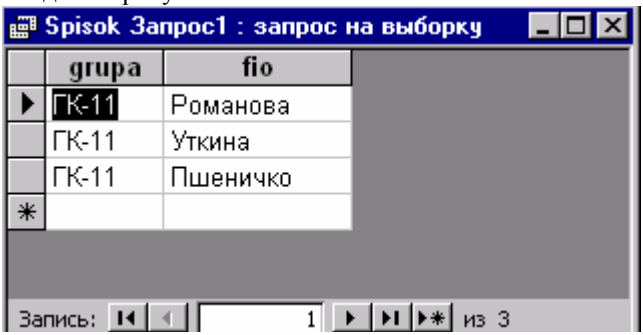

Рисунок 1.3 – Команда SELECT с предложением WHERE

Замечание. Столбец, используемый в предложении WHERE (в примере grupa) необязателен в выходных данных.

Исключение дублирующих значений

Предикат DISTINCT в команде SELECT исключает повторяющиеся записи, содержащие повторяющиеся значения в выбранных полях. Значения в каждом поле должны быть уникальными.

Предположим, требуется составить список футбольной команды на основании таблицы Sport. В команду включается по одному студенту из каждой группы (принцип исключения повторяющихся значений):

SELECT fio, grupa **DISTINCT** futbol **FROM** Sport;

Предикат DISTINCT проверяет, какие значения появились в выходном списке данных, и исключает из него дублирующие значения. . Таким образом, команда SELECT позволяет на основании таблицы-источника данных получить необходимую информацию в желаемом виде.

Операторы в сравнениях

Операторы сравнения

В SQL применяются операторы сравнения для задания типов сравнения между двумя значениями: = *(равно),* > *(больше, чем),* < *(меньше),* >= *(больше или равно),* <= *(меньше или равно),* <>*(неравно).*

Например, надо выбрать фамилии студентов, которые получают стипендию, превышающую 20 000 руб.

```
SELECT *
   FROM Spisok
        WHERE stip > 20 000;
```

Булевы операторы

Оператор **AND** сравнивает два выражения — *A AND B — в* качестве аргументов и в результате выдает истину только в случае истинности обоих.

Оператор OR сравнивает два выражения — *A OR B* — в качестве аргументов и оценивает результат как истину, если хотя бы один из них истинен.

Оператор NOT анализирует единственное булево выражение *NOT A* в качестве аргумента и изменяет его значение на противоположное (с истинного на противоположное или сложного на истинное).

Пример. Выбрать из таблицы Spisok всех студентов группы ГК-11, которые получают стипендию менее 20 000 руб.:

```
SELECT *
   FROM Spisok
        WHERE grupa =’ГК-11’ AND stip<20000;
```

Специальные операторы в условиях

Для формирования запросов применяются специальные операторы **IN, BETWEEN, LIKE, IS NULL**.

Оператор **IN** позволяет выбрать из множества те элементы, значения которых точно принадлежат именам полей, перечисленных в круглых скобках через запятую.

Пример. Выбрать из таблицы Sport всех студентов, которые получают именные стипендии в сумме 45000 и 50000 руб.:

```
SELECT *
   FROM Sport
        WHERE stip IN (45000, 50000);
```

Оператор **BETWEEN** по своим функциям похож на оператор **IN**. В нем задается граница для начального и конечного значений множества, а между ними расположено ключевое слово **AND.**

Пример. Выбрать из таблицы Spisok всех студентов, фамилии которых начинаются от буквы Е до буквы М:

```
SELECT*
   FROM Spisok
        WHERE fio BETWEEN ‘E’ AND ‘M’;
```

Оператор **BETWEEN** чувствителен к порядку записей в таблице (они должны быть рассортированными).

Оператор **LIKE** применяется только для выборки значений из полей типа CHAR или VARCHAR с поиском подстрок в указанном поле.

Применяются два типа шаблонов:

• символ *«подчеркивание» (_)* — заменяет в строке один любой символ;

• *символ «процент» (%)* — заменяет последовательность символов произвольной длины, включая и нулевую.

Оператор **IS NULL** возвращает в запросе записи с пустыми значениями.

Подведение итогов с помощью функций агрегирования

В запросах можно выполнять вычисления, результаты которых не запоминаются в базовой таблице. Каждый раз при выполнении запроса вычисления выполняются над текущим содержимым базы данных — для этого применяются базовые (агрегатные) функции (таблица 1.1).

Таблица 1.1 - **Агрегатные функции SQL**

Функция	Действие функции
COUNT ()	Подсчет числа выбранных строк, исключая NULL-значение
SUM()	Суммирование всех выбранных значений данного поля
AVG()	Подсчет среднего значения для всех выбранных значений
MAX()	Вычисляет наибольшее из всех выбранных значений
MIN()	Вычисляет наименьшее из всех выбранных значений

Функции используются в командах языка SQL.

Синтаксис функций в команде **SELECT:**

SELECT *функция (столбец_1 или *),..., функция (столбец_n)*

FROM *имя_таблицы;*

В функциях AVERAGE, MIN, MAX, SUM значение NULL недопустимо для подсчетов.

Синтаксис функции COUNT:

SELECT COUNT (имя_столбца)

FROM имя_таблицы;

Подсчитывается количество значений столбцов или записей, которые соответствуют выражению выбора (цифровые и символьные поля). Применение маски (*) в качестве аргумента вместо имени столбца определяет все строки, удовлетворяющие критерию команды **SELECT**. Например, для подсчета количества всех строк в таблице Spisok подается команда:

SELECT COUNT(*)

FROM Spisok;

Для подсчета количества строк по критерию отбора в команде **SELECT** используется предложение **WHERE**. Так, для возврата количества студентов 5-го курса подается команда:

SELECTCOUNT(*) FROM Spisok

WHERE kurs = 5;

Функция AVG возвращает среднее арифметическое значение для выделенных столбцов (в примере для столбца stip всех строк таблицы Spisok):

SELECT AVG (stip)

FROM Spisok;

Функция **SUM** вычисляет и возвращает арифметическую сумму значений столбцов (только для цифрового поля).

Синтаксис функции SUM:

SELECT SUM ([DISTINCT] выражение)

FROM имя_таблицы;

Пример. Найти общую сумму стипендии по группе ГБ-41 4-го курса:

SELECT SUM (stip) **FROM** Spisok

WHERE grup='ГБ-41' **AND** kurs = 4;

Для подсчета итогов по каждой группе из серии групп применяется предложение **GROUP BY**. Например, для подсчета суммы стипендии по каждой группе надо подать команду:

SELECT stip, **SUM** (grup) **FROM** Spisok **GROUP BY** stip;

Внутри одного выражения можно использовать несколько функций:

SELECT MAX (stip), **MIN** (stip), **AVG** (stip) **FROM** Spisok;

Формирование результатов запросов

Для пользователя важно не просто получить нужную информацию, но и получить ее в определенном виде. Для этого SQL имеет средства для вставки текста и констант в выбранные поля, упорядочения выходных полей (ORDER BY), вставки комментария в выходные данные.

Упорядочение выходных полей

Для вывода результатов в запросе в требуемой последовательности применяется команда **ORDER BY**. Сортировку можно задавать по значению одного или нескольких выбранных полей. Последовательность сортировки для каждого из столбцов задается ASC (по возрастанию) или DESC (по убыванию). Столбец, по значениям которого упорядочиваются возвращаемые строки, можно указывать именем столбца или его относительным порядковым номером.

Пример. Рассортировать все записи в таблице Spisok по алфавиту студентов:

SELECT *

FROM Spisok

ORDER BY fio **ASC;**

Выходные столбцы формируются в запросе, а не извлекаются непосредственно из базовой таблицы (их нет в таблице базы данных); они не имеют имен. Для ссылки на выходные столбцы в предложении **ORDER BY** *используется порядковый номер выходного столбца* из предложения **SELECT**. В этом случае в SELECT используются не имена столбцов для указания полей, а номера в выходных данных (это не номера в таблице базы данных).

Пример. Подсчитать количество отличников (по размеру стипендии — 30374 руб.) на каждом курсе и вывести результат в порядке возрастания курса:

SELECT kurs, COUNT (DISTINCT 30374)

FROM Spisok

GROUP BY kurs

ORDER BY 2 **ASC;**

Соединение таблиц

Запрос может быть сформулирован (получен) на основе данных из нескольких таблиц. Между элементами таблиц устанавливаются связи; имена соединяемых таблиц перечисляются через запятую в запросе в предложении FROM. Предикат запроса может ссылаться на любой столбец любой соединяемой таблицы. Полное имя столбца соединяемых таблиц составляют:

<имя_таблицы>.<нмя_столбца>

Примеры записи столбцов.

Spisok.fio
Sport,gimnast
Kultura.pevec

Пример соединения таблиц. Установить связь между таблицами Sport и Kultura для выбора студентов, которые играют в футбол и занимаются сольным пением:

SELECT Sport.fio, Kultura.pevec
FROM Sport, Kultura
WHERE Sport.fio = Kultura.fio;

Команда **SELECT** в данном примере берет первую запись из таблицы Sport, в которой перечислены студенты, увлекающиеся футболом, и ищет в таблице Kultura, в которой перечислены студенты, занимающиеся сольным пением, фамилию этого же студента.

В операциях соединения таблиц кроме равенства (эквисоединения) можно использовать любые операции сравнения в предложении **WHERE**.

1.6 Объединение таблицы по принципу »Сама с собой»

Смысл этой операции в том, что любую одну строку (запись) таблицы можно комбинировать (соединять) с любой другой строкой этой же таблицы и с копией этой строки. В одной таблице устанавливаются связи между различными записями, например поиск и выбор пар строк с общим значением поля.

Для указания имен столбцов в команде **SELECT** в предложении FROM применяются временные имена полей — алиясы (псевдонимы), которые позволяют рассматривать одну таблицу в команде SELECT как две независимые таблицы: первая (first) и вторая (second). Алиясы определяются в предложении FROM. Записываются имя таблицы, пробел и имя алияса для данной таблицы, например FROM Spisok first, Spisok second.

Соединение таблицы по принципу «сама с собой» покажем на примере соединения всех пар одинаковых фамилий в разных группах:

SELECT first.fio, second.fio, first.grup
FROM Spisok.first, Spisok.second;

Вложение запросов

Один запрос может быть вложен в другой запрос. Запрос, который вкладывается, — это *подзапрос*, а в который вкладывается — *основной* (внешний). Подзапрос записывается в предложении **WHERE**, выполняется первым по отношению к внешнему, и используется для определения истинности или ложности предиката.

Пример. Уточнить размер стипендии студента Курдесова:

```
SELECT *
   FROM stip =
         (SELECT stip
               FROM Spisok
                     WHERE fio =’Курдесов’);
```

Выполнение запроса начинается с подзапроса: просматриваются все записи таблицы Spisok и выбираются все записи, для которых значение поля равно ‘Курдесов’; для этих записей выбирается значение для поля stip, выбранное значение подставляется в предикат основного запроса в предложение WHERE вместо самого подзапроса (предложение будет иметь вид: WHERE stip = 9005). Подзапрос возвращает одно и только одно значение. Необходимо быть уверенным, что подзапрос выдает только одну строку. Далее выполняется основной запрос.

Для гарантии того, что результатом подзапроса является единственное значение (строка), рекомендуется использовать аргумент DISTINCT:

(SELECT DISTINCT stip...**)**

Использование агрегатных функций в подзапросе

Агрегатные функции (COUNT, SUM, AVG, MAX, MIN) автоматически выдают единственное значение для любого количества строк, которое используется в предикате.

Пример. Выбрать фамилии студентов, стипендия которых меньше средней на курсе:

```
SELECT*
   FROM Spisok
         WHERE stip <
               (SELECT AVG(stip)
                     FROM Spisok
                           WHERE kurs = 1);
```

В подзапросах можно использовать оператор **IN** в предложении **WHERE** или **HAWING** внешнего запроса.

Подзапросы могут быть вложены в предложения **SELECT, UPDATE, DELETE, INSERT**.

Оператор EXISTS проверяет только наличие в таблице результатов вложенного запроса хотя бы одной строки. Он используется для образования предиката, который фиксирует, будет ли подзапрос генерировать выходные данные. Оператор EXISTS генерирует значение «истина» или

«ложь». Его можно применять в комбинации с операторами **AND, OR, NOT**. В операторе EXISTS подзапрос используется в качестве аргумента; подзапрос не может принимать значение «неизвестно».

Пример. Вывести с таблицы Spisok список академических групп, если ни один студент не получает повышенной стипендии (>30374).

SELECT kurs, grupa FROM Spisok

WHERE EXISTS (SELECT* FROM Spisok

WHERE stip > 200 000);

Оператор EXISTS фиксирует наличие выходных данных подзапроса.

Главный запрос последовательно просматривает все строки таблицы Spisok, и для каждой академической группы выполняется вложенный запрос. Результатом вложенного запроса является столбец данных, содержащий названия академических групп и студентов, получающих стипендию, превышающую 200 000 руб.

Операторы ANY и ALL. Для многократного сравнения в SQL имеются операторы ANY и ALL. В проверке этих операторов используется один из шести операторов (=, <>, <, <=, >, >=). При сравнении строк в Access регистр не учитывается.

В **операторе IN** проверяется, не равно ли некоторое значение одному из множества значений в столбце результатов вложенного запроса.

Объединение результатов нескольких запросов

Результатная информация находится в структурной и смысловой взаимосвязи. Результаты двух запросов и более можно объединить в одну таблицу командой UNION следующего формата (общий вид):

<запрос> {UNION [ALL] <запрос>}...;

Синтаксис оператора UNION в развернутом виде:

Оператор_SELECT
UNION [All]
Оператор_SELECT
UNION [All]
Оператор_SELECT
[, ...n]

Команда UNION объединяет результаты нескольких запросов. Каждый из них должен содержать одинаковое число элементов в выражении, содержащем значение> в предложении SELECT, а элементы, стоящие в одном и том же месте в <выражениях, содержащих значения>, должны быть совместимыми по типу данных и размеру.

В объединении подзапросы не могут управлять друг другом. Подзапросы объединения выполняются независимо, а затем их выходные данные объединяются. Команда **UNION** объединяет выходные данные двух запросов и более в единый выходной документ (единое множество строк и столб-

цов). По умолчанию оператор UNION автоматически устраняет повторяющиеся строки из результата объединения.

Пример. Вывести информацию о всех студентах (таблица Spisok), играющих в футбол (таблица Sport), в виде (запрос):

Фамилия	Группа

```
SELECT fio, grupa
   FROM Spisok
     WHERE futbol = 1
UNION
SELECT fio, grupa
  FROM Sport
    WHERE futbol=l;
```

Функция команды UNION приведена на рисунке 1.4.

Результат объединения выглядит так, как будто он получен одним запросом. Столбцы в выходных данных не поименованы. Обратите внимание: только другой запрос заканчивается точкой с запятой — после первого запроса данный знак отсутствует.

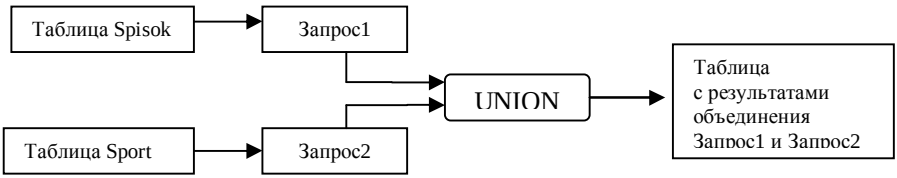

Рисунок 1.4 – Оператор UNION для объединения результатов запросов

1.7 Представления

Основу базы данных составляют *базовые таблицы*, т. е. таблицы с реальными данными. На основе данных базовых таблиц по запросам результаты представляются в таблицах, которые не содержат собственных данных. *Представление (\\t-w)* — это альтернативный способ просмотра данных из одной или нескольких таблиц; это виртуальная таблица, которая не содержит собственных данных; ее содержимое берется или выводится из других таблиц. Представления:

1) позволяют пользователям видеть данные базы данных по-разному, с позиций своих интересов;

2) упрощают доступ к базе данных: каждый пользователь видит данные определенной структуры, хранимые в базе данных;

3) являются одним из средств ограничения доступа к данным (пользователь видит не все столбцы и строки). Создается представление командой CREATE VIEW:

CREATE VIEW <имя_представлення>
AS SELECT предикат (WITH CHECK OPTIONS];

На протяжении выполнения команды CREATE VIEW выполняется запрос, по результатам которого формируется содержимое таблицы-представления. Если указано **WITH CHECK OPTIONS**, то эти обновления должны удовлетворять условию запроса.

Пример. Создать представление с именем SportGimnast, отображающее пловцов и гимнастов 1-го курса:

```
CREATE VIEW SportGimnast
  AS SELECT plovec, gymnast
    FROM Sport
      WHERE kurs=l;
```

Групповое представление содержит предложение **GROUP BY** или базируется на других групповых представлениях. Например, деканаты ежедневно отслеживают пропуски занятий студентами. Можно

получить соответствующую информацию по запросу, а можно создать представление и получать нужную информацию:

```
CREATE VIEW Propuski
  AS SELECT date, COUNT (propuski), SUM ()
   FROM Spisok
     GROUP BY date;
```

После этого можно получить необходимую информацию с помощью запроса:

```
SELECT*
    FROM Propuski;
```

1.8 Определение прав доступа к данным

Информация базы данных должна находиться в состоянии полной готовности без права на отказы и сбои и круглые сутки быть доступной для пользователей. В этих условиях база банных должна быть надежно защищена. Для охраны данных от порчи и потери используются соответствующие комплексы программных и технических средств. Защита информации сводится к обеспечению высокой доступности к ней.

В SQL Server применяются следующие ***методы защиты*** *данных:*

- резервное копирование данных;
- сервер «теплого» резервирования — первоначально создается полная резервная копия базы данных, а к ней с заданной периодичностью добавляются резервные копии журнала транзакций;

- сервер «горячего» резерва в составе отказоустойчивого кластера MSCS. В этом случае клиент обращается к имени виртуального сервиса на кластере независимо от того, каким конкретно узлом обрабатывается его запрос в данный момент;
- тиражирование данных.

Защита от несанкционированного доступа предусматривает два этапа:

1 аутентификация пользователя — убеждение в том, что пользователь именно тот, кем он представляется;

2 управление правом доступа пользователя:

- пользователь имеет право оперировать только разрешенными объектами;
- пользователь совершает над объектами только дозволенные ему действия.

К данным базы могут обращаться много пользователей одновременно. Одним пользователям разрешается, а другим запрещается использовать какие-то объекты. За безопасность данных отвечает

СУБД. Основой системы безопасности СУБД является язык SQL, в котором применяются три принципа защиты данных:

1 *пользователи.* Любые операции с базой данных СУБД выполняет от имени пользователя. СУБД может выполнить требование пользователя, а может и отказаться выполнять его с учетом прав конкретного пользователя и требований безопасности баз данных;

2 *объекты базы данных* (таблицы, формы и др.) защищаются языком SQL;

3 *привилегии* предоставляют пользователю право выполнять те или иные действия над определенными объектами базы данных.

Система безопасности в SQL рассмотрена на рисунок 1.5.

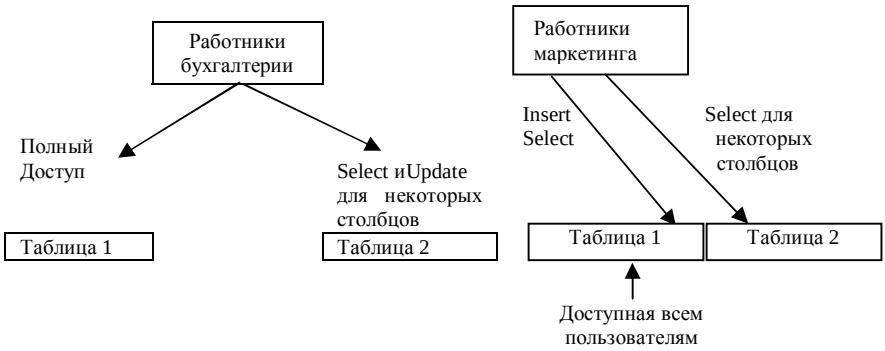

Рисунок 1.5 – Система безопасности в SQL

Замечание. Полный доступ ко всем данным (таблицы 1–4) имеют руководители отделов и ведущие специалисты.

Для доступа к информации в базе данных SQL выполняет функцию распознавания и дифференциации пользователей. Администратор базы данных может назначить пользователям привилегии.

Привилегия — способ определения того, может или нет конкретный пользователь выполнить данную команду. Минимальная привилегия — вход в систему. Привилегии назначаются командой **GRANT** и отменяются командой **REVOKE.**

Пользователь. В среде SQL каждый пользователь имеет идентификатор в виде имени или числа. Команды, которые подаются базе данных, связаны с идентификатором (имя) Пользователя. Для ссылки на идентификатор пользователь SQL может применять ключевое слово **USER.** Привилегии дают идентификатору пользователя право выполнять действия на различных объектах базы данных.

Доступ к компьютерной системе. Для доступа в многопользовательских системах используется специальная процедура входа, которая определяет, какой идентификатор пользователя (ID) связан с текущим пользователем. Одному пользователю (человеку) с множеством функций может быть назначено множество ID. Один идентификатор также может применяться несколькими пользователями. Во всех случаях SQL связывает пользователя с его ID.

Передача привилегий. Привилегии непостоянны: добавляются новые, отменяются старые. Часть привилегий определена в ANSI SQL. Они называются объектными привилегиями: команда выполняется только для определенного объекта базы данных.

Пользователь, создавший таблицу, является ее владельцем. Он имеет все привилегии на таблицу и может назначить привилегии для работы с ней другим пользователям. Объектные привилегии связаны с пользователями и таблицами.

Стандартные привилегии, которые может назначить пользователь командой **GRANT,** приведены в таблице 1.2.

Таблица 1.2 – **Стандартные привилегии**

Привилегия	Что может пользователь?
INSERT	Может выполнять команду INSERT для таблицы
SELECT	Может выполнять запросы для таблицы
UPDATE	Может выполнять команду UPDATE для таблицы
DELETE REFERENCES	Может выполнять команду DELETE для таблицы Может определить внешний ключ

Пример. Пользователь Kolas является владельцем таблицы Kultura и желает передать пользователю Genijz право на формирование запроса к этой таблице. Пользователь Kolas должен ввести такую команду:

GRANT SELECT ON Kultura TO Genijz;

Сам пользователь Genijz не может задать выполнение этой команды.

Группы привилегий и группы пользователей

Командой **GRANT** передаются не только единичные привилегии единственному пользователю (см. предыдущий пример), а и их списки, разделенные запятыми.

Пример. Передача права на выполнение команд **SELECT, INSERT:**

GRANT SELECT, INSERT ON Kultura TO Genijz;

передача двум пользователям:

GRANT SELECT, INSERT ON Kultura TO Genijz, Furs;

Можно установить привилегии для изменения одного или всех столбцов таблицы. Например, командой **GRANT UPDATE ON Kultura TO Algerd;** пользователю Algerd разрешено изменять столбцы таблицы Kultura.

Командой **GRANT UPDATE (stip) ON Spisok TO Algerd;** разрешается пользователю Algerd изменять только значения столбца (поля) stip.

Команда **GRANT** поддерживает два аргумента **ALL и PUBLIC.**

Аргумент ALL используется для передачи всех привилегий для таблицы, например **GRANT ALL ON** Spisok TO Mikola;

Аргумент PUBLIC (общедоступный) относится к пользователям, а не к привилегиям. Все пользователи получают привилегии автоматически.

Пример. **GRANT SELECT ON** Spisok **TO PUBLIC;**

Создатель таблицы может передать другим пользователям этой таблицы право в свою очередь *передать привилегии на эту таблицу третьим пользователям.* Это реализуется в команде **GRANT** в предложении **WITH GRANT OPTION.**

Пример. **GRANT SELECT ON** Spisok TO Lubomir **WITH GRANT OPTION;**

Пользователь Lubomir получил право передать третьим лицам привилегии выполнения команды **SELECT.**

Замечание. Таблицы всегда принадлежат их создателям. Перед именем таблицы следует указывать идентификатор автора таблицы:

GRANT SELECT ON Mikola.Spisok TO Zmagar **WITH GRANT OPTION;**

Лишение привилегий

Выполняется командой **REVOKE,** которая имеет формат, аналогичный формату команды **GRANT,** но с противоположным значением.

Пример. Лишим пользователя Zmagar привилегия на выполнение команды INSERT для таблицы Spisok:

REVOKE INSERT ON Spisok **FROM** Zmagar;

В командах **GRANT и REVOKE** списки привилегий и пользователей доступны.

Пример. **REVOKE DELETE, INSERT ON** Spisok **FROM** Janka, Janina;

Пользователь, который назначил привилегии, отменяет их последовательно по отношению ко всем пользователям (каскадно)

Работа SQL с множеством пользователей

К базе данных обращается одновременно n-е число пользователей, поэтому возможны накладки между различными действиями. СУБД обеспечивает контроль и управление одновременно выполняющимися транзакциями.

Транзакция (transaction) — это механизм, обеспечивающий выполнение целого ряда действий в качестве единого неделимого задания. Каждая из команд, которые выполняются одновременно, выполняется так, как она выполнялась бы при отсутствии других команд от самого начала до момента получения результата. Тем самым не разрешается воздействие на таблицу свыше одной транзакции в один момент времени. Вместе с тем база данных постоянно доступна для многих пользователей. Управление их работой (одновременное выполнение транзакций) называется *параллелизмом.*

В СУБД есть механизм управления совпадающими операциями (блокировка): обновление таблицы должно выполняться без взаимного влияния пользователей; воздействие одного действия на частичный результат другого действия; одновременное выполнение действий, которые оказывают взаимное влияние. Для разрешения подобных ситуаций система управления не позволяет единовременное воздействие на таблицу более чем одной транзакции, ограничивает определенные операции и устанавливает очередность их выполнения.

Многопользовательская система снабжена схемой блокировки для управления совпадающими операциями по умолчанию. Эта схема может быть определена для всей базы данных либо индивидуально. Системы управления предоставляют детектор тупиковых ситуаций — две операции выполняют взаимные блокировки: выполнение одной из команд откладывается, и ее блокировка отменяется.

Блокировки бывают разделительными и исключительными.

Разделительные блокировки обеспечивают в единицу времени выполнение команд более чем одним пользователем: множество пользователей, не изменяя данные, получают к ним доступ. Этот вид блокировки применяется для запросов.

Исключительные блокировки запрещают всякий доступ к данным со стороны множества пользователей и применяются для команд, которые изменяют структуру базы данных и ее содержимое. Исключительная блокировка сохраняется до конца транзакции.

Какую часть таблицы охватывает блокировка, определяет уровень изоляции, который имеет три уровня:

- уровень изоляции «*повторное чтение*» предоставляет гарантию неизменности всех значений, участвующих в запросе, внутри данной транзакции;
- уровень изоляции *столько для чтения*» делает моментальный снимок данных. Его нельзя применять вместе с командами обновления;
- уровень *стабильности курсора* защищает каждую запись в процессе ее изменения при чтении.

В некоторых СУБД блокировки выполняются на уровне страницы, а не строки. Страница—это единица объема памяти (обычно 1024 байт).

Системный каталог

База данных SQL состоит из множества объектов: таблицы, индексы, представления, синонимы, пользователи, привилегии и др. Все объекты имеют собственную организацию. Внутренняя структура базы данных представлена соответствующей информацией в таблицах, которая хранится в *системном каталоге*, состоящем из системных таблиц. Владельцем системного каталога является сама СУБД — суперпользователь (системный администратор). При выполнении операторов SQL СУБД постоянно обращается к данным системного каталога и проверяет, например, имеет ли пользователь право на доступ, существует ли запрашиваемая таблица базы данных и другая информация.

Таблицы системного каталога создаются автоматически при создании базы данных и принадлежат ей. Они объединяются под системными именами SYSTEM, MASTER, SYSIBM или DBA. Каждая таблица системного каталога содержит информацию об элементе базы данных. К основным относятся пять системных таблиц: *таблицы*, *пользователи*, *столбцы*, *представления*, *привилегии*. Таблицы системного каталога нельзя обновлять, но можно формулировать запросы.

1.9 Использование SQL с другими языками программирования

Язык SQL используется для написания программ доступа к базам данных в качестве подъязыка других процедурных языков программирования (Pascal, PL/1, Fortran, COBOL, C, Ada и др.). Язык SQL не процедурный, а декларативный, локальный. Его можно встраивать* в программы, написанные на процедурных языках. В языке SQL отсутствуют базовые операторы проверки условий IF, операторы FOR, DO и WHILE и др. Язык SQL предназначен исключительно для управления базами данных.

Совместное применение этих языков позволяет программировать сложные процедурные программы, например программировать их на

Pascal, а через SQL обращаться к базе данных. Все встроенные команды SQL включаются в основной текст программы на языке высокого уровня, начинаются фразой EXEX SQL и заканчиваются знаком завершения соответствующего языка (в Pascal;).

Программа с встроенным SQL перед собственной компиляцией проходит предкомпиляцию, где команды SQL преобразуются (транслируются) предкомпилятором в форму языка высокого уровня; после этого происходит компиляция всей основной программы.

Команды SQL, включаемые в другую программу, рассматриваются основной программой как SQL-процедуры. В процессе выполнения основная программа связывается с базой данных, как и пользователь в среде SQL.

Контрольное задание

С использованием языка SQL выполнить запросы, приведенные в контрольном задании из первой части данного пособия, глава 7.

2 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРИКЛАДНЫХ ПРОГРАММ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

2.1 Средства автоматизации обработки данных в системе управления базой данных

Современные СУБД содержат инструментальные средства разработки приложений: развитый язык программирования и инструменты отладки. Для автоматизации операций в простых задачах применяются макросы, для разработки более сложных приложений используется язык Visual Basic. Microsoft Access 2000 имеет средства для доступа к данным в интрасети организации, для создания приложений Access, непосредственно связанных с SQL Server.

Понятие макроса и модуля

Любая процедура, связанная с обработкой информации, подразумевает определенную последовательность действий. Бели такая последовательность выполняется многократно, то ее объединяют в единый объект, имеющий имя. Это позволяет автоматически выполнять макрокоманды процедуры путем одноразового нажатия кнопки (или другим способом). *Макрокоманда* — это любая команда, выполняемая Access. Созданный макрос можно применять в различных документах ACCESS.

Макрос—это не программа, а набор из одной или нескольких макрокоманд, которые обеспечивают последовательность операций и применяются для автоматизации их выполнения, например открытие форм, распечатка отчетов.

Макросы разделяются на *простые* (состоящие из последовательности макрокоманд) и *групповые* (набор логически связанных макросов). Групповые макросы сохраняются под общим именем. По способу выполнения последовательности макрокоманд макросы делятся *на линейные* и *макросы с условием* (условие задается логическим выражением).

Для запуска макроса нужно обратиться к его имени: в меню выбрать пункт **«Вид» →«Имена макросов».** Для запуска макроса из группы нужно указать имя группы и после точки — имя макроса, например **«Печать.Приход»** или **«Печать.Расход»**. Некоторые задачи нельзя решить путем создания документов в среде Office 2000, включая макросы — в таком случае необходимо программировать задачи на языке Visual Basic for Application (VBA). На этом языке создаются объекты базы данных, которые называются модулями.

Модуль — это объект, который включает программы на языке Visual Basic. Модуль состоит из процедур — совокупностей команд языка VB.

База данных может содержать два вида модулей:

1. стандартные модули, которые являются объектами базы данных;
2. модули форм или отчетов, которые являются частями объектов форм или отчетов.

Стандартные модули открытой базы данных можно увидеть в окне базы данных путем вызова вкладки **«Модули».** Они предназначены для создания и сохранения процедур, которые выполняются (вызываются) из запросов, отчетов.

Новый модуль создается в окне базы данных кнопкой **«Создать».** Для просмотра или изменения существующего модуля нужно нажать кнопку **«Конструктор»** →[окно редактора VB].

В Access форма или отчет могут быть связанными с соответствующими модулями для обработки событий.

Событие (event) — это какое-либо действие или ситуация, создаваемые пользователем, которые могут потребовать соответствующей реакции со стороны программы. Классические примеры событий — нажатие кнопок и клавиш интерфейса, перемещение мыши, открытие формы. Программа постоянно отслеживает событие и сразу реагирует на него. Событие — это любое изменение состояния объекта Access (открыть форму, закрыть форму, ввод новой строки в форму и др.), действие, которое распознается объектом управления.

Для обработки событий можно создать макрос или процедуру VB.

Объект, например форма, после получения сообщения о событии приступает к обработке события (реакция объекта на событие). Каждый объект может реагировать только на предварительно зафиксированные события. Перечень событий в Access приведен в таблице 2.1.

Автомакросы (autoMacros) — это макросы со специально фиксированными именами. Они вызываются в ответ на возникшее событие. Имени макроса соответствует определенная последовательность команд, которая заменяет имя макроса в любом месте программы, где встречается имя макроса. Автомакрос — способ обработки событий.

Таблица 2.1 – **События в СУБД ACCESS**

Название событий	Название событий
Включение (Activate)	Открытие (Open)
Возврат (Retreat)	Отсутствие в списке (NotInList)
Вход (Enter)	Отсутствие данных (NoData)
Выгрузка (Upload)	Ошибка (Error)
Выход (Exit)	Перемещение указателя (MouseMove)
Двойное нажатие кнопки (DblClick)	Печать (Print)
До встречи (BeforeInsert)	Получение фокуса-курсора (GetFocus)
До подтверждения (BefbreDelConftrm)	После вставки (AfterInsert)
Добавление элемента (ItemAdded)	После обновления (AfterUpdate)
Завершение (Terminate)	После подтверждения (AfterDelConfirm)
Загрузка (Load)	Потеря фокуса-курсора (LostFocus)
Закрытие (Close)	При обновлении (Updated)
Изменение (Change)	Применение фильтра (ApplyFilter)
Изменение размера (Resize)	Страница (Page)
Клавиша вверх (KeyUp)	Таймер (Timer)
Клавиша вниз (Key Down)	Текущая запись (Current)
Кнопка вверх (MouseUp)	Удаление (Delete)
Кнопка вниз (MouseDown)	Удаление элемента (ItemRemoved)
Нажатие клавиши (Keypress)	Фильтрация (Filter)
Нажатие кнопки (Click)	Форматирование (Format)
Отключение (Deactivate)	

2.2 Создание макроса

Последовательность действий:

1) в окне базы данных выбрать вкладку **«Макросы»**;
2) нажать кнопку **«Создать»** →[Макрос] (рисунок 2.1);

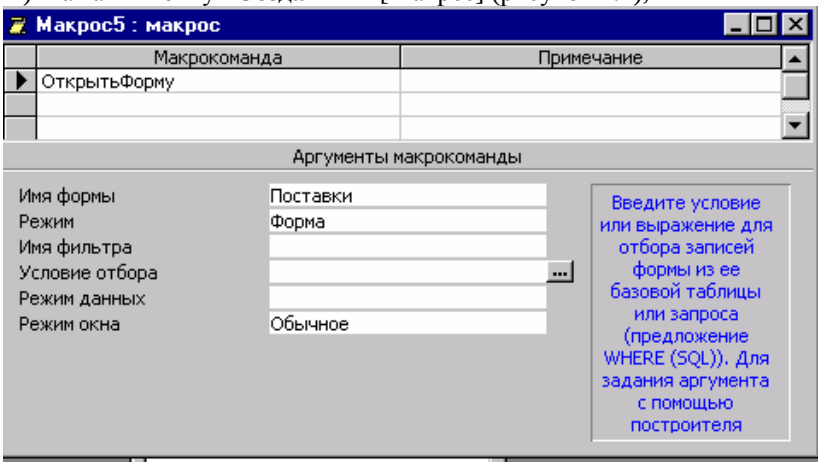

Рисунок 2.1 – Окно диалога «Макрос»

3) в окне диалога «Макрос» в ячейке столбика «Макрокоманда» раскрыть список макрокоманд, которые могут быть использованы в макросе (в списке имена типовых процедур, например «Выделить объект», «Вывод на экран» и др.);

4) выбрать имя макрокоманды;

5) по желанию ввести текст комментирования к макрокоманде;

6) в нижней части окна макроса указать аргументы (константа, переменная или выражение, которые являются источником данных для макрокоманды, процедуры или метода);

7) для включения в макрос других макрокоманд перейти на другую строку и повторить действия, указанные в пп. 3–6.

Быстрый способ создания макроса

Выполнить в определенной последовательности необходимые действия над конкретными объектами базы данных (таблица, запрос, форма, отчет), которые войдут в макрос: в окне базы данных выбрать объект и мышью перетянуть его в окно макроса в ячейку макрокоманды. Окна базы данных и макроса разместить рядом **(«Окно» →«Сверху вниз»** или «Слева направо»).

Применение условий в макросе

Условие определяет порядок передачи управления при выполнении макроса между макрокомандами. Условие задается логическим выражением.

Примеры выражений в макросах:

IsNull ([Фамилия]) — в форме, из которой запускается макрос, встроенная функция проверяет поле «Фамилия» на равенство значению Null (содержится ли в нем пустое значение);

Forms!Список!Stip < 200 000 — в форме «Список» значение поля «Stip» менее 200 000;

[Фамилия] = «Астрожский» And Forms![Культура]![Спонсор] > *5* **млн.**— одновременно выполняются два условия: в форме, из которой запускается макрос поля «Фамилия», сохраняется значение «Астрожский», а в форме «Культура» — значение поля «Спонсор» более *5* млн. руб.

Для ввода условий команд в макрос нужно находиться в окне макроса в главном меню и выбрать пункт **«Вид» →«Условие»**; в результате в окне макроса выводится третий столбик **«Условие»** для записи условий. Следующая макрокоманда будет выполняться только при выполнении заданного условия. Если условие «истина», то выполняется макрокоманда в строке записи этой макрокоманды.

Работа с макросами

Сохранение макроса: на панели инструментов нажать кнопку **«Сохранить»** → [«Сохранение»] → при первом сохранении макроса в поле **«Имя макроса»** ввести имя макроса и нажать кнопку «ОК».

Запуск макроса. После запуска макроса макрокоманды будут выполняться, начиная с первой строчки и до конца макроса (или до начала следующего макроса, если макрос входит в группу макросов).

Варианты запуска макроса пользователями:

• *из окна макроса* кнопкой **«Запуск»** (!) на панели инструментов;

• *из окна базы данных* выбрать вкладку **«Макросы»** - выбрать имя макроса и дважды щелкнуть по нему;

• *в режиме конструктора формы или в режиме конструктора отчета* выбрать в меню пункт **«Сервис»→ «Макрос»→ «Запуск макроса...»** →[Запуск макроса]→ в поле **«Имя макроса»** ввести или выбрать имя макроса →кнопка **«ОК»**;

• *из любого окна ACCESS:* пункт **«Сервис» → «Запустить макрос»** →выбрать нужный макрос в поле **«Имя макроса».**

Для запуска макроса, который входит в группу макросов, нужно руководствоваться общим синтаксисом.

Пример.

«Имя Группы макросов. Имя макроса»

Печать. Отчет

Группа макросов

Имя макроса в группе макросов

Создание **кнопки для запуска макроса без Мастера:**

1) открыть форму в **режиме** Конструктора;

2) для отключения Мастера на панели элементов предназначена кнопка **«Мастер»;** щелкнуть по ней, если она уже нажата;

3) на панели элементов нажать кнопку **«Кнопка»;**

4) в форме мышью выбрать место, в левой части которого размещается верхний левый угол кнопки, и щелкнуть — появится элемент управления под названием «Кнопка» с маркерами;

5) уточнить, что кнопка выбрана (выбранный элемент имеет маркеры перемещения и изменения размеров) и нажать кнопку **«Свойства»** на панели инструментов; открывается окно **«Кнопка»** с вкладками;

6) ввести в ячейку свойств **«Имя»** имя макроса (кнопки);

7) для размещения на кнопке подписи перейти на вкладку **«Макет»** и ввести текст подписи в ячейку свойств **«Подпись».** Вместо подписи можно поместить рисунок: открыть форму в режиме Конструктора, выбрать кнопку и нажать кнопку **«Свойства»** на панели инструментов, ввести в ячейку **«Рисунок»** путь и имя файла .bmp, .ico, .dib, .wmp.

Назначение сочетания клавиш для запуска макроса:

1) в окне базы данных выбрать вкладку **«Макросы»;**

2) нажать кнопку **«Создать»;**

3) на панели инструментов нажать кнопку **«Имена макросов»;**

4) указать в ячейке столбика **«Имя макроса»** клавишу или сочетание клавиш, с которыми связывается макрос;

5) ввести в макрос макрокоманды, которые должны выполняться при нажатии назначенных клавиш;

6) сохранить группу макросов под именем «Autokrys»

Отладка макроса в пошаговом режиме (по одной команде):

1) открыть макрос (в окне базы данных вкладка **«Макрос»** →выбрать макрос; нажать кнопку **«Конструктор»;** внести нужные изменения в макрос);

2) на панели инструментов нажать кнопку **«По шагам»;**

3) нажать кнопку **«Запуск»** → [Пошаговое исполнение макроса];

4) в окне диалога нажать кнопку **«Шаг»** для выполнения макрокоманды, имя которой выведено в окне диалога;

5) нажать кнопку **«Прервать»** для остановки выполнения макроса и закрытия окна диалога:

6) нажать кнопку **«Продолжить»** для отключения режима пошагового выполнения и выполнить часть оставшегося макроса.

Корректировка макроса

Добавление макрокоманды в макрос:

1) в окне макроса выбрать первую пустую ячейку в столбике **«Макрокоманда»**. Для вставки макрокоманды между существующими командами выделить строку макрокоманды, перед которой вставляется новая макрокоманда, и нажать на панели инструментов кнопку **«Добавить строки»;**

2) раскрыть список макрокоманд кнопкой в ячейке столбика **«Макрокоманда»;**

3) выбрать имя макрокоманды;

4) при необходимости ввести текст комментирования;

5) при необходимости в нижней части окна указать аргументы макрокоманды.

Перемещение/Копирование макрокоманды:

1) выбрать элемент, который нужно переместить/копировать;

2) для перемещения нажать кнопку **«Вырезать»,** а для копирования – **«Копировать»;**

3) выбрать место, куда будет помещен объект;

4) нажать кнопку **«Вставить»** на панели инструментов.

Классификация макрокоманд. Типы макросов

Макрокоманды, которые можно включить в макросы, по функциональному признаку делятся следующим образом:

- открытие и закрытие объектов Access, например:

Close — закрывает активное или указанное окно таблицы, запроса, формы, отчета;

OpenForm - открывает форму в режиме таблицы, Конструктора формы;

- выполнение запроса, например:

OpenQuery - открывает запрос на выборку в режиме Конструктора или запускает запрос и выводит набор записей в режиме таблицы;

- печать данных, например:

OpenReport — печатает отчет или открывает отчет в режиме предварительного просмотра;

- проверка истинности условий и управление выполнением макрокоманд;
- вставка значений, например:

Requery (обновление) — обновляет данные в элементе управления, который связан с запросом;

- поиск данных, например:

FindRecord (НайтиЗапись) — ищет запись, которая соответствует условию поиска;

- построение специального меню и выполнение команд меню, например:

AddMenu (ДобавитьМеню) — добавляет раскрывающееся меню в специальную строку меню или в специальное контекстное меню для формы или отчета;

- управление выводом на экран и фокусом, например:

Maximize (Развернуть) – увеличивает активное окно до размера рабочей области Access;

GoToPage (НаСтраницу) – передает фокус первому элементу управления, который размещен на указанной странице активной формы;

- сообщение пользователю о выполняемых действиях:

Beep (Сигнал) – выдает звуковой сигнал;

- переименование, копирование, удаление, сохранение, импорт и экспорт объектов:

CopyObject (КопироватьОбьект) – копирует объект текущей базы данных в другую базу данных Access или в ту же самую под новым именем;

- запуск других приложений:

RunApp (ЗапускПриложения) — запускает приложение MS DOS или Windows.

Типы макросов:

а) клавишные;

б) условные — в виде программного модуля на языке VB.

Сферы применения макросов. Макросы незаменимы при закреплении последовательности действий за комбинацией клавиш (нажатие) и управлении запуском приложений при открытой базе данных. Их целесообразно использовать при небольшом количестве форм и отчетов в приложении, для автоматизации выполнения задач и др.

Однако макросы имеют ограниченный диапазон применения. Для программирования приложений MS Office применяется язык Visual Basic.

2.3 Основы программирования на языкеVBA. Основные типы данных и их описание

В программировании базовыми понятиями являются «переменная» и «значение переменной». *Переменная* (variable) – это величина (объект), значение которой меняется в ходе выполнения программы. Переменная в компьютере представляется по присвоенному ей имени (идентификатору), и ее значение определяется оператором присвоения.

Для каждого идентификатора переменной выделяется место в памяти, где хранится ее значение. Запомним и уясним, на первый взгляд, элементарное понятие: любая программа использует переменные и их значения. Для каждого встречаемого в программе нового имени переменной VBA автоматически определяет ее (выделяет для нее место в памяти). Это место остается пустым до присвоения переменной какого-либо значения. Переменные сравнивают с контейнерами для хранения данных любых типов.

Данные описывают объект числами, буквами и другими способами. Над числами выполняются арифметические, над буквами – логические операции. Уже это свидетельствует о разных типах данных, для каждого из которых приняты свои способы их внутримашинного представления.

Итак, каждая переменная обладает собственным типом. Типы данных делятся на простые (или скалярные) и сложные. У простых типов значения данных единые и неделимые. Простые типы данных подразделяются на арифметические, строковые и логические.

В Visual Basic Application применяются 11 типов данных и один пользовательский тип данных; их характеристики приведены в таблице 2.2.

При написании программ используются константы и переменные, которые могут применяться:

- в одной процедуре;
- во всех процедурах модуля;
- во всех процедурах базы данных.

Таблица 2.2 – **Типы данных и значений**

Типы данных	Префикс	Символ описания	Размер	Значение (может содержать)
Integer	Int	%	2 байта	Короткое целое число: от –32788 до +32767
Long	Lng	&	4 байта	Длинное целое число: от -2 147 483 648 до + 2 147 483 647
Single	Sng	!	4 байта	Число с плавающей точкой одинарной точности: от -3,4E38 до +3.4E38

Типы данных	Префикс	Символ описания	Размер	Значение (может содержать)
Double	Dbl	#	8 байт	Число с плавающей точкой двойной точности: от –1.79E308 до 1.79E308
Byte	Byt	(нет)	1байт	Байт: от 0 до 255
Currency	Cur	@	8 байт	Число с фиксированной точкой: от –922 337 203 685 477,5808 до +922 337 203 685 477,5807
String	Str	$	10 байт +2 байта на символ	Строка: от 0 до 65535 символов
Boolean	Bin	(нет)	2 байта	Булевское число: True или False
Date	Dat	(нет)	8 байт	Дата и время
Variant	Var	(нет)	< 16 байт	Любые данные (универсального типа, определяемые пользователем)
Object	Obj	(нет)		Любая ссылка (указатель) на объект

Объявление переменных и массивов

Переменные в программе объявляются, т. е. определяется тип переменной и область действия (видимости). Объявлять переменные можно на двух уровнях — уровне процедуры и уровне модуля.

Объявление *на уровне модулей* производится операторами **Public** и Private; *на уровнях модулей и процедуры* - оператором **Dim;** *только на уровне процедуры (локальные)* — оператором **Static.**

Инициализация объявленных переменных происходит во время компиляции: числовой переменной присваивается значение нуль (0); строковой переменной — пустая строка (нулевой длины); переменной типа Variant — значение Emtry (отсутствие значения).

Константы, их объявление. Как и переменные, константы имеют имя, но не изменяют своего значения во время выполнения программы. Для определения констант в Visual Basic используется оператор **Const** такого формата:

[Public | Private] Const {имя константы [AS <тип данных>] = <константное выражение>}…

Поясним значение ключевых слов.

Public — объявленная константа доступна для любой процедуры из всех модулей базы данных;

Private — объявленная константа доступна для процедур только внутри своего модуля;

AS <тип данных> - объявление типа данных константы.

Примеры объявления констант.

```
Const Max = 100000
Public Const strVolume = «Maximum»
Private Const Max AS Integer = 100 000
Private Const PI AS Double = 3.14159
```

Значение переменной изменяется в процессе выполнения программы. Если переменная доступна многим программам, то какая-то программа может изменить значение переменной; чтобы избежать этого, ограничивают доступ к переменным. Для этого пользователь-программист определяет, какие процедуры будут иметь доступ к переменной, — определяет область действия (область видимости) переменной.

Массив— это n-мерная совокупность индексированных переменных одного типа. Массиву присваивается имя – имя переменной, за которым в круглых скобках указано одно (для одномерных) или два (для двумерных) натуральных положительных числа. Массив состоит из элементов (переменных), которые в памяти размещаются последовательно. Обращение в программе ко всему массиву производится по его имени. Для обращения к конкретному элементу массива необходимо указать его имя и в круглых скобках — индекс (номер) массива, который указывает на относительную позицию (номер) элемента в массиве. Все элементы массива имеют одно имя, но различные номера (положения) в массиве. Число элементов в массиве задается при его объявлении в операторе Dim и во время выполнения программы не изменяется.

Объявление массива — это задание имени массиву, определение типа его элементов и количества элементов. Элемент массива является переменной и может быть использован, как любая классическая переменная. Массивы могут быть числовыми или символьными.

Для объявления переменной или массива используется оператор **Dim.** Его упрощенный формат для объявления переменной:

Dim <имя переменной> AS <имя типа >

для объявления массива:

Dim <имя массива> (<список размерности>) AS <имя типа >)

Примеры

```
Dim intVector(40) AS Integer          одномерный массив
Dim intMatrica (4 TO 6) AS Integer    двумерный массив
```

Соглашение по идентификации

Для ссылки на объекты в программе применяются идентификаторы (имена). Имя переменной в программе является именем поля памяти, где хранится информация. Идентификатор можно сравнить с этикеткой ящика для хранения данных. Желательно, чтобы имя переменной указывало на содержательный смысл величин.

При выполнении программы значения переменных по их именам выбираются из конкретных адресов оперативной памяти. Результаты обработки также заносятся по адресам памяти, адекватным именам переменных.

Правила присвоения переменным имен:

- длина имени не может превышать 255 символов;
- первым символом в имени переменной должна быть буква;
- в имени можно использовать прописные или строчные буквы, числа и знак подчеркивания;
- запрещены ключевые слова (имена функций и операторов VBA) и символы @, $, #, &, %, !.

Имя объекта может начинаться с *префикса типа данных* (три латинские буквы) объявляемого объекта. Например: intСтипендия (int — целое число), strProzvisca (str — символьное число), bitПроверка (bit — логический тип).

2.4 Математические и логические функции и операторы

Математические операторы:

+ - сложение;

- - вычитание;

/ - деление;

*- умножение

Математические функции:

Функция	**Описание**
SQR(x)	Возвращает квадратный корень числа
X^n	Возвращает значение числа в степени n
Sin(x)	Возвращает синус угла в радианах
Cos (x)	Возвращает косинус угла в радианах
Abs(x)	Возвращает модуль числа
Tan(x)	Возвращает тангенс угла в радианах
Atn(x)	Возвращает арктангенс угла в радианах
Exp(x)	Возвращает значение числа e в степени x
Log(x)	Возвращает натуральный логарифм числа

Для целых чисел используют оператор MOD. Возвращает остаток при целом делении двух чисел (значение по модулю).

Синтаксис:

Результат = число1 Mod число2

Логические операторы:

Стандартные логические операции (**>,<,>=,<=,<>**)

1 Оператор **OR**. Выполняет операцию логического ИЛИ (сложения) для двух выражений. Синтаксис: результат=выражение1 **Or** выражение2

2 Оператор **AND**. Возвращает результат конъюнкции (логического И)

для двух выражений. Синтаксис: результат=выражение1 **And** выражение2

3 Оператор **Not.** Выполняет над выражением операцию логического отрицания. Синтаксис: результат=**Not** выражение

2.5 Ввод и вывод данных

Для ввода данных используется оператор присваивания (например: a=200) или функция InputBox.

Функция InputBox расширяет возможности организации диалога с пользователем, позволяя пользователю ввести данные. Функция имеет следующий синтаксис:

InputBox(сообщение[,заголовок]),

где сообщение – это подсказывающее сообщение, которое нужно вывести в диалоговом окне; заголовок – это текст, который будет помещен в строку заголовка диалогового окна. Если этот параметр будет опущен, строка заголовка будет пустой.

Например: переменной a будет присвоено значение, введенное пользователем.

a=InputBox(«Введите число a»,»Ввод данных»)

Для вывода данных используется функция **MsgBox**. Эта функция выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем, а затем возвращает значение типа Integer, указывающее, какая кнопка была нажата.

Формат функции:

MsgBox(prompt[,buttons][,title][,helpfile,context])

Prompt – обязательный аргумент. Строковое выражение, отображаемое как сообщение в диалоговом окне. Максимальная длина строки prompt составляет приблизительно 1024 символов и зависит от ширины используемых символов. Строковое значение prompt может содержать несколько физических строк. Для разделения строк допускается использование символа возврата каретки (Chr(13)), символа перевода строки (Chr(10)) или комбинации этих символов (Chr(13)) & (Chr(10)).

Buttons – необязательный. Числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку. Значение по умолчанию этого аргумента равняется 0- отображается только кнопка ОК.

2.6 Понятие процедуры. Типы процедур

На языке VBA *процедура* — это самостоятельная замкнутая программная единица, включающая операторов описания локальных данных процедуры и операторов, которые выполняются в ней.

В одной логической строке желательно записывать один оператор; при записи нескольких операторов в одной строке надо отделять их двоеточием (:).

Процедура выполняется автоматически в ответ на событие. События возникают в результате действий пользователя или выполнения программы; их может генерировать система.

Процедуры начинаются ключевым словом Sub. *Ключевое слово* — это слово или символ, которые распознаются как элемент языка VB.

Синтаксис процедур в VBA:

[Private | Public][Static] **Sub** имя ([список_аргументов])
[AS < тип данных>]
<тело_процедуры >
[Exit Sub]
<тело_процедуры >
End Sub

Поясним значение параметров:

Sub – основное ключевое слово (глагол) процедуры; означает, что идущие за ним строки есть тело процедуры (подпрограммы);

[Private] – процедура, доступная для других процедур только того модуля, в котором она написана;

[Public] – объявление процедуры общедоступной, т.е. объявленную процедуру могут вызывать из окон других процедур во всех модулях;

[Static] – локальные (объявленные в теле процедуры) переменные сохраняются в промежутках времени между вызовами этой процедуры;

«имя» – имя процедуры;

[список_аргументов] – перечисленные через запятую переменные, которыми задаются передаваемые процедуре параметры при ее вызове;

<тело_процедуры > – операторы программы выполнения процедуры;

End Sub — конец описания процедуры.

Ключевое слово Sub определяет начало процедуры, а End Sub – конец процедуры.

Типы процедур. В Visual Basic применяются процедуры двух типов:

- *процедуры-подпрограммы* Sub; общий формат:

Sub
<тело процедуры >
End Sub

- *процедуры-функции* Function; общий формат:

Function
<операторы>
End Function

Процедура-функция возвращает единственное значение (полученное в результате расчета, возврат текущей даты и др.). Процедура-подпрограмма Sub выполняет действия, но не возвращает значение. Процедуры обоих типов могут иметь аргументы — переменные, значения которых определяют работу процедуры при конкретном вызове.

Создание процедуры

Процедуры создаются в окне модуля, к которому добавляется процедура. Процедуры состоят из блоков команд и выполняют действия программы. Существуют два типа процедур:

- процедуры обработки события (event procedure) — обработка событий, связанных с элементом управления в формах или отчетах;
- процедуры преобразований.

Алгоритм создания процедуры обработки события

Рассмотрим это на примере создания процедуры обработки события: при каждом открытии формы «Spisok» курсор должен автоматически устанавливаться в форме в поле «Grupa».

Последовательность действий:

1 в окне базы данных выбрать вкладку **«Формы»** или **«Отчет»**. Перейти в режим **«Конструктор»**. Вызвать командой **«Вид» →«Свойства»** диалоговое окно **«Форма»** (рисунок 2.2). В нем выбрать вкладку **«События»**;

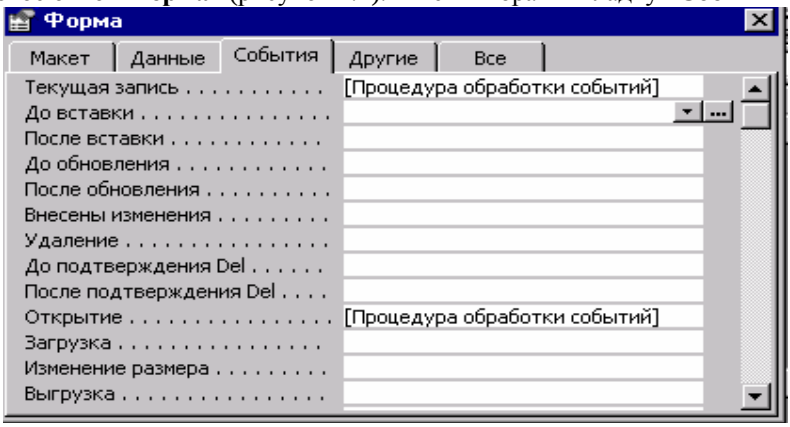

Рисунок 2.2. – **Окно «Форма» с процедурами обработки событий**

2 установить курсор в поле события (на рисунке 2.2 они перечислены), с которым связывается процедура модуля;

3 из раскрывающегося списка выбрать **«Процедура обработки события»;**

4 для перехода в окно модуля нажать кнопку **с тремя точками**. Откроется окно **[Построитель].** Выбрать нужный построитель из трех предложенных (выражение, макросы, программы). Выбрать **«Программы»** и нажать кнопку **«ОК».** Откроется окно модуля (рисунок 2.3).

Вверху окна модуля расположены два раскрывающихся списка. В левом списке содержатся элементы управления объектом (формой или отчетом), во втором списке перечислены процедуры обработки применительно к тому объекту, который выбран в списке объектов (в первом списке).

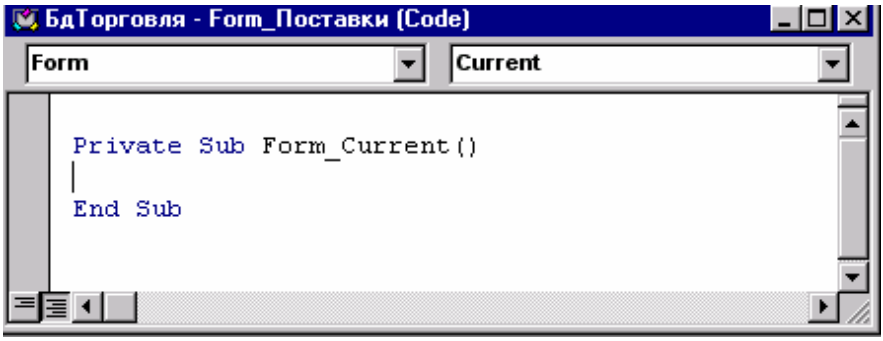

Рисунок 2.3 – Окно модуля для формирования текста процедуры

В списке «Объект» выбрать имя элемента управления, например «ЗаголовокОтчета», а в списке «Процедура» — имя события, которое активизирует эту процедуру, например «Print». В результате в окно модуля выводятся операторы процедуры;

1 в окне модуля автоматически будут установлены две строки еще не созданной процедуры:

- строка заголовка процедуры (операторы начала – Private Sub...);
- строка с закрывающим оператором (конца процедуры – End Sub).

Между ними ввести команды создаваемой процедуры. В нашем примере это

Forms!Spisok!Gruppa.SetFocus

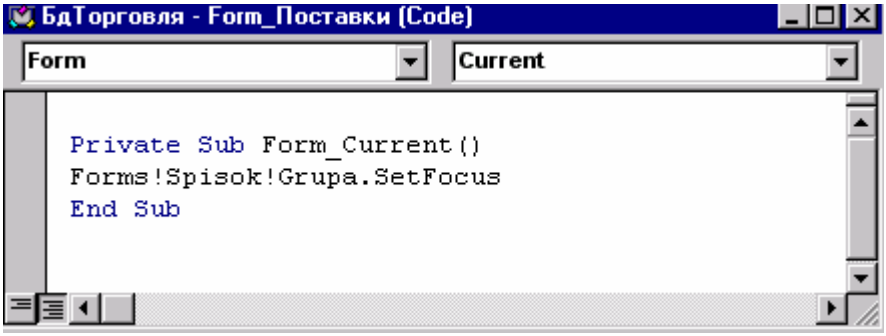

Рисунок 2.4 –Вид процедуры обработки события

2 по команде **«Отладка» →Компилировать загрузочные модули»** выполнить отладку процедуры;

3 командой **«Файл»→ «Сохранить»** сохранить созданную процедуру как модуль.

Процедура обработки события размещается в модуле формы или отчета и служит для выполнения определенного программного кода при наступлении определенного события. Событие связано с действием со стороны другой программы или пользователя. При выполнении прикладной программы автоматически вызываются события из процедуры их обработки.

Пояснение. Команда Set присваивает значение объектной переменной и переводит фокус в поле «Grupa».

Алгоритм создания процедуры преобразования:

1 в окне базы данных перейти на вкладку **«Модули»**;

2 далее возможны два варианта действий:

- создаваемую процедуру поместить в новый модуль – нажать кнопку **«Создать» →Модуль**;
- создаваемую процедуру дополнить в существующий модуль – нажать кнопку **«Конструктор»**;

3 дать команду **«Вставка»→ «Процедура» →Вставка процедуры**;

4 в диалоговом окне установить нужные значения:

- в поле **«Имя»** ввести название процедуры;
- в области **«Тип»** выбрать тип создаваемой процедуры (подпрограмма, функция, свойство);
- в зоне **«Область определения»** выбрать один из параметров Public или Private;

5 нажать кнопку **«ОК»**;

6 ввести текст созданной процедуры;

7 сохранить созданную процедуру командой **«Файл»→ «Сохранять»→ Сохранение→ ввести имя модуля → «ОК».**

Пример процедуры распечатки текстового файла. Процедура должна выполняться до тех пор, пока будет справедливо условие (может ни разу не выполниться), т. е. пока не будет достигнут конец записей в файле. После прочтения последней записи при попытке считывать данные за последним символом файла функция EOF возвращает значение True.

```
Public Sub КонецФайла()
    ‘объявление строковой переменной
Dim strInputData$
    ‘Открытие файла на чтение
Open “Spisok” for Input As #1
    ‘Заголовок цикла: пока не достигнут конец файла
Do While Not Eof(1)
    ‘Считывание строки из файла в переменную
Line Input #1, strInputData
    ‘Вывод в переменную
```

```
Debug.Print strInputData
Loop
    'Закрытие файла, открытого оператором Open
End Sub
```

Пояснение. Оператор **Open** открывает файл для ввода | вывода; выделяет буфер для операций ввода | вывода в файл и определяет режим использования данного буфера.

Оператор **Input** возвращает символы из файла, открытого для последовательного доступа.

Оператор **Line Input** считывает посимвольно строку из файла в переменную до тех пор, пока значение функции EOF не получит значение True.

Оператор **Not** – логическое отрицание.

Отладка процедуры выполняется в окне модуля с помощью команд:

«Отладка»→ «Компилировать загруженные модули»

«Отладка»→ «Все модули»

«Отладка» → «Компилировать и сохранить все модули»

Отладка выполняется традиционным способом. Строки с ошибками выделяются иным цветом.

Редактирование процедуры

Для внесения изменений в тело процедуры следует открыть окно соответствующего модуля, загрузить процедуру и выполнить редактирование.

Редактирование процедуры преобразования:

1) в окне базы данных перейти на вкладку **«Модули»;**
2) выбрать имя модуля, где хранится процедура;

3) кнопкой **«Конструктор»** перейти в режим редактирования. *Редактирование процедуры обработки событий* выполняется по алгоритму (пп. 1–4) создания этой процедуры.

2.7 Операторы языка VBA

Язык VBA относится к операторным языкам. Программа состоит из последовательности операторов, которых в VBA много. Термин «оператор» в программировании означает знак или другой символ, указывающий на операцию, которая выполняется над одним элементом или более. Оператор — самый малый исполняемый элемент в языках программирования.

Операторы в одной строке отделяются друг от друга символом «двоеточие». Однако для удобства отладки и модификации программ рекомендуется в одной строке писать один оператор. Если он не помещается в одной строке, то для его переноса (продолжения) на следующую строку ставятся два символа: «пробел» и «подчеркивание». Оператор может быть помечен меткой.

Простейшие операторы

Оператор присвоения Let обозначается знаком «равно» (=), но это не знак математического равенства. Предназначен для присвоения переменной значения выражения. Это основной оператор, выполняющий вычисления заданного выражения. Формат оператора:

[Let] переменная = выражение

Переменная» — это имя переменной или имя функции. Знак «=» читается «присвоить». «Переменная» получает значение *выражения*, записанного справа от знака «=». Значением «выражение» может быть:

- имя переменной, например S = A; типы обеих переменных должны быть одного типа;
- значение константы, например S = 25.4;
- арифметическое выражение, например S = A + B.

В операторе присваивания типы обеих переменных (слева и справа от знака « = ») должны быть одного типа.

Оператор комментария предназначен для комментирования отдельных операторов, участков программы, модулей. Идентифицируется ключевым словом REM или апострофом (‘) и может включать любой текст в строке справа от оператора, который комментируется. От предыдущего оператора отделяется двоеточием.

Примеры.

```
S = 3.14 R                    ‘Площадь круга
Stoim = Kol * Cena            ‘ Стоимость материалов
```

Операторы управления

Оператор условного перехода IF передает управление ходом выполнения программы в зависимости от истинности некоторого условия. Формат оператора:

```
If условие Then
        [операторы]
[ElseIf условие-n Then
        [операторы-n]...
[Else
        [ИначеОператоры]]
End If
```

Параметр *«условие»* — обязательная компонента, любое числовое или строковое выражение со значением «истина» или «ложь». *Операторы* — один или несколько разделенных двоеточием операторов.

Если условие истинно, выполняются операторы за ключевым словом Then; если нет, то отыскивается первое истинное *условие-n* и выполняются *операторы-n;* в противном случае (все условия ложны) выполняются *ИначеОператоры*. После выполнения одной последовательности «операторы» управление передается на оператор, записанный за End If. Общая схема работы оператора If приведена на рисунке 2.6.

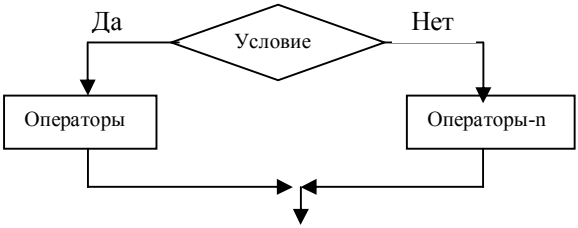

Рисунок.2.6 – **Схема работы оператора If**

Пример 1.

```
If stip< 10000
   THEN stip = (stip * 30) / 100
        Else stip = (stip * 20) / 100
End If
```

Оператор выбора одной из нескольких альтернатив Select Case (оператор вычисляемого перехода) передает управление в зависимости от результата сравнения значения некоторого выражения со списком или диапазоном значений (рисунок 2.7).

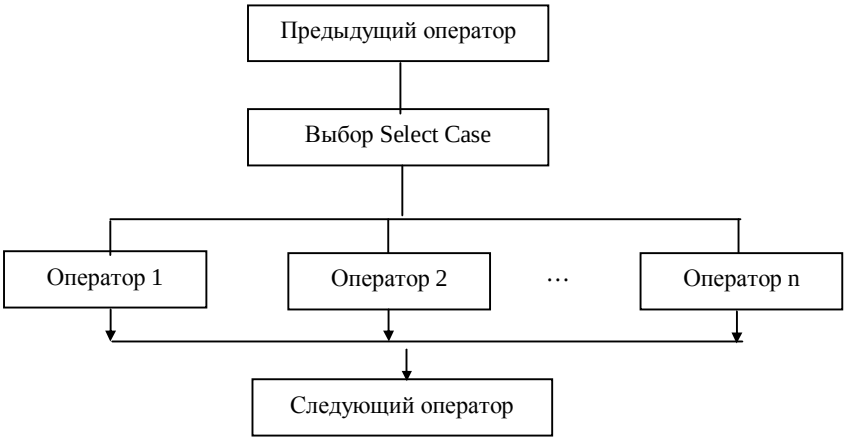

Рисунок 2.7 - Схема работы оператора Select Case

Формат оператора:

Select Case <проверяемое-выражение >
[Case <список-сравнение-1>
[<операторы-1>]]
…
[Case Else
[<ИначеОператоры>]]
End Select

Оператор Select Case выбирает и исполняет одну из последовательностей операторов Case. Параметр *проверяемое выражение* должен присутствовать обязательно, его значение может быть числовым или строковым.

Параметр *список-сравнение-1* также должен присутствовать в строке Case. Список сравнений, компоненты которого перечисляются через запятую, может иметь одну из трех форм:

- *«выражение»* - задает отдельное значение;
- *«выражение» TO выражение»;*
- *«IS <оператор-сравнение>выражение»,* где оператор-сравнение – это { = | > | < | > | <= | >= }.

Если «проверяемое выражение» совпадает с одним из выражений параметра «список-сравнение», то выполняется «оператор-1»; если «проверяемое выражение» удовлетворяет несколько предложений Case, выполняются только операторы первого Case; если «проверяемое выражение» не находит своего значения в предложении Case, выполняются <ИначеОператоры>, расположенные после предложения Case Else.

Операторы для организация циклических процессов предназначены для выполнения последовательности операторов цикла. Применяются такие операторы циклов: **for... Next, for Each... Next, While... Wend, Do... Loop.**

Оператор цикла с параметром For... Next (рисунок 2.8) выполняет операторы тела цикла заданное число раз, имеет такой формат:

for V = V1 TO V2 [Step V3]
Операторы тела цикла
[Exit for]
Next V

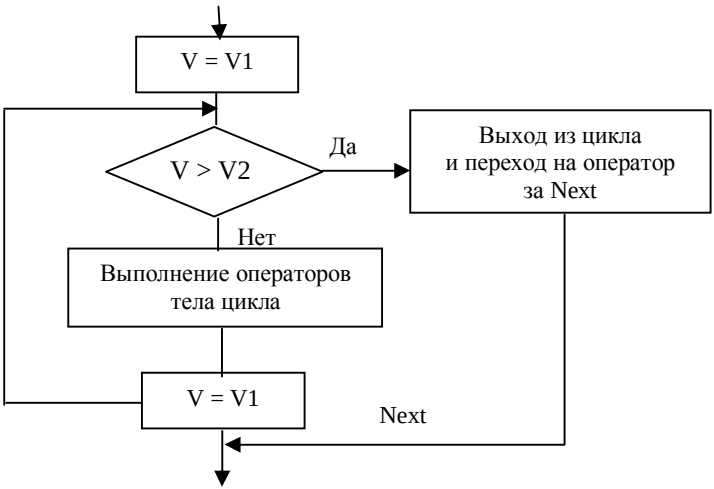

Рисунок 2.8 – Схема работы оператора For... Next

Назначение параметров: V – числовая переменная (счетчик цикла); V1 – начальное значение параметра цикла, первоначально присваивается переменной V, V2 – конечное значение параметра цикла; V3 – шаг цикла (приращение), не обязательный параметр, по умолчанию он равен +1. Параметры V1, V2, V3 – арифметические выражения.

Шаг цикла — это число, прибавляемое к значению счетчика. *Операторы тела цикла* — это последовательность операторов в теле цикла, которая будет выполнена заданное число раз.

Необязательный оператор *Exit For* может находиться в любом месте цикла и обеспечивает выход из цикла, не дожидаясь выполнения условия завершения цикла.

Работа оператора for... Next: после получения управления вычисляются, если необходимо, выражения V1 и V2, затем они сравниваются, и пока V1 будет меньше или равно V2, выполняется последовательность операторов (может быть и один оператор) тела цикла, пока не встретится оператор Next. В конце каждого шага параметр цикла V увеличивается оператором Next на величину V3; выполняется проверка счетчика на конечное значение; как только выражение V1 превысит выражение V2, управление передастся на выполнение оператора, записанного после оператора конца цикла NEXT (происходит выход из цикла). Перед выполнением цикла все величины V1, V2, V3, если это необходимо, вычисляются.

Оператор Next устанавливает границу, за которой заканчивается цикл. Он изменяет значение переменной V на величину шага цикла V3.При положительном значении шага параметр увеличивается на величину шага, при отрицательном — уменьшается.

Пример. Найти и вывести на печать сумму квадратов чисел (S) от 1 до 50. Опишем цикл с оператором FOR (а) и оператором DO (б):

```
а) IntЧисло = 1
            IntS = 1
   For intЧисло = 1 TO 50 Step 1
            S = S * intЧисло
   Next intЧисло
б) IntЧисло = 1
            IntS = 1
   Do While intЧисло < 50
            S = S * intЧисло
   Loop
```

Цикл с неизвестным числом повторений While... Wend используется для выполнения операторов в теле цикла до тех пор, пока условие сохраняет значение «истина». Формат оператора:

```
While <условие>
   тело цикла
Wend
```

Параметр *условие* — любое числовое выражение; *тело цикла* — последовательность операторов. Ключевое слово While обозначает «пока». Значение «условие» вычисляется перед каждым выполнением тела цикла. На каждом шаге выполнения тела цикла управление передается оператору While. Как только значение «условие» примет значение «ложь», тело цикла пропускается и управление передается на оператор, записанный после оператора Wend. Операторы тела цикла выполняются, пока истинно некоторое условие.

2.8 Примеры программы на VBA

Пример 1

Рассмотрим пример модуля, начинающегося следующим образом:

```
Public A1 As String
Private A2 As Integer
Dim A3 As Single
Sub Proc1()
   Dim A4 As Integer
   Static A5 As Integer
   A1 = "Текстовая строка 1"
   A2 = 2
   A3 = 3.14
   A4 = A4 + 4
   A5 = A5 + 5
   MsgBox A4
   MsgBox A5
End Sub
Sub Proc2()
   Proc1
   MsgBox A1
   MsgBox A2
   MsgBox A3
   MsgBox A4
   MsgBox A5
   Proc1
End Sub
```

Здесь переменная A1 определена на уровне всего проекта (использовано ключевое слово Public), переменные A2 и A3 определены на уровне модуля, переменная A4 – только на уровне процедуры Proc1, а переменная A5 хотя и определена в теле процедуры Proc1, но описана как статическая переменная.

При вызове процедуры Proc2 произойдет следующее: из этой процедуры будет в свою очередь вызвана процедура Proc1, которая присвоит значения всем пяти переменным A1, A2, A3, A4 и A5, а затем покажет текущие значения переменных A4 и A5 в диалоговом окне.

После завершения этой процедуры будут выведены текущие значения переменных A1 – A5 из процедуры Proc2. При этом оказывается, что переменные A1 – A3 сохранили свои значения, поскольку они описаны на уровне модуля, а переменные A4 и A5 принимают пустые значения, поскольку областью действия этих переменных являются процедуры, в которых они используются. Никакие изменения этих переменных внутри одной из процедур не имеют отношения к аналогичным переменным из другой процедуры — на самом деле это разные переменные, просто для них используются совпадающие имена.

Затем происходит еще один вызов процедуры Proc1, и она вновь начинает изменять и выводить на экран значения переменных A4 и A5. При этом переменная A4 вновь получит значение 4, поскольку при новом вызове процедуры для этой переменной будет заново выделена память и она будет инициализирована пустым значением. В отличие от A4, переменная A5, описанная как статическая переменная, сохранит свое прежнее значение от предыдущего вызова этой процедуры, и в результате ее значение при повторном вызове окажется равным 10.

Пример 2

```
Select Case x
Case Is < 2
y = 4 * x ^ 2 + 11 * x
MsgBox "Переменная x = " & x & Chr(13) & "Результат Y = " & y
Case 2
y = 5 - 8 * x ^ 2
MsgBox "Переменная x = " & x & Chr(13) & "Результат Y = " & y
Case 6 To 25
y = 2 * x - 5 * x
MsgBox "Переменная x = " & x & Chr(13) & "Результат Y = " & y
Case Is >= 26
y = Abs(x - 4 * x ^ 2)
MsgBox "Переменная x = " & x & Chr(13) & "Результат Y = " & y
Case Else
MsgBox "Переменная не соответствует ни одному из предложенных диапазонов"
End Select
```

В данном случае иллюстрируется работа оператора Select Case.

2.9 Работа в окне редактора кода VBE в Microsoft Access

В окне базы данных модулям посвящена отдельная вкладка – **Модули.** Окно модуля появляется при открытии существующего модуля (с помощью кнопки **Конструктор)** или при создании нового (после нажатия кнопки **Создать**).

Модули формы и отчета создаются с помощью кнопки **Программа,** которая появляется на панели инструментов при переходе в режим конструктора формы или отчета, а также с помощью команды **Программа** из меню **Вид.**

Модули создаются в редакторе Visual Basic Editor (VBE). Редактор VBE содержит окно просмотра объектов, окно просмотра свойств объектов и окно редактирования кода, а также имеет средства компиляции и отладки написанных программ.

Чтобы открыть окно редактора, достаточно открыть любой модуль Access 2000.

Окно редактора имеет три панели:

– **Панель проекта (Project). На** ней представлено иерархическое дерево модулей приложения.

– **Панель свойств (Properties).** Дает возможность просматривать и корректировать свойства входящих в проект объектов, представленных на панели проекта.

– **Панель редактора кода** (Code). Позволяет одновременно открывать несколько окон данного типа для различных модулей.

Для настройки параметров редактора используется диалоговое окно **Options,** которое открывается при вызове команды **Options** из меню **Tools** окнаVBA. Это окно имеет четыре вкладки: **Editor, Editor Format, General и Docking.**

Новому модулю по умолчанию присваивается стандартное имя *Module1.*

При создании нового модуля Access 2000 добавляет в раздел *описания* оператор **Option Compare Database**, с помощью которого устанавливается режим сравнения текстовых данных в модуле.

Помимо области описания, модуль может включать одну или несколько процедур, которые отсутствуют в новых модулях. Если в левом нижнем углу окна модуля нажата кнопка **Procedure View,** то отображается только одна, выбранная пользователем, процедура, а если кнопка **Full Module View** — все содержимое модуля.

Для создания процедуры (подпрограммы или функции) нажмите кнопку **Программа** на панели инструментов **База данных** в окне Access 2000. В результате ее активизации открывается редактор кода VBA.

Создадим простую процедуру-функцию. Предположим, что на основе цен, выраженных в рублях, должны быть автоматически определены и указаны в отдельном поле цены в американских долларах. Мы будем использовать форму *GoodsPrice,* содержащую поля *Goods, Price* и *PriceD.*

Тело созданной процедуры состоит из одного оператора присваивания (=).

Для решения данной задачи необходимо создать процедуру, определяющую значение поля *PriceD* на основе содержимого поля *Price*.

- Чтобы открыть новый стандартный модуль, щелкните в окне базы данных на ярлыке **Модули** в списке **Объекты,** а затем нажмите кнопку **Создать** на панели инструментов окна базы данных.
- В открывшемся окне Microsoft Visual Basic выберите команду **Procedure** в меню **Insert.**
- В диалоговом окне Add Procedure выберите тип Function и введите имя *RubleDollar*. В области Scope (Область определения) установите переключатель **Public.**
- Далее введите в окно модуля следующий код:

Public Function *RubleDollar*()
'Пересчет цены в рублях в цену в долларах
Forms ! *GoodsPrice* ! PriceD = Forms ! *GoodsPrice* ! Price / 29
End Function

- Выберите в меню **File** редактора команду **Save**, чтобы сохранить модуль с созданной функцией. Присвойте модулю имя *RubleDollar*.

В результате его выполнения вычисляется значение цены в долларах, которое заносится в соответствующее поле формы. После ввода каждой строки процедуры программа проверяет синтаксис введенного оператора. При обнаружении синтаксической ошибки на экране появляется соответствующее сообщение.

Редактирование модуля осуществляется в специальном окне, которое открывается в результате нажатия кнопки Конструктор в окне базы данных. При этом на вкладке **Модули** должно быть отмечено имя подлежащего обработке модуля.

Код модуля и тексты подпрограмм/функций редактируются так же, как документы в обычном текстовом редакторе.

Усложним рассмотренный ранее пример пересчета цен. Значение курса валют зададим как аргумент для функции пересчета *RubleDollar*. В случае изменения курса валют пользователю не придется редактировать текст процедуры – будет достаточно изменить только аргумент функции при обращении к ней.

При описании процедуры, зависящей от аргументов, имя аргумента принято вводить в скобках за именем процедуры в строке с ключевым словом Function/Sub:

Public Function RubleDollar(Rate)
'Пересчет цены в рублях в цену в долларах
Forms ! GoodsPrice ! PriceD = Forms ! GoodsPrice ! Price / Rate
End Function

При вызове такой функции значение аргумента указывается в скобках после имени функции:

= RubleDollar(29)

Как правило, процедуры-функции ориентированы на обработку событий. Access 2000 позволяет связывать разработанные процедуры-функции с вычисляемым полем несколькими способами (например, можно создать процедуру-функцию, которая будет вызываться при выполнении щелчка на некотором поле, из макроса или из другой процедуры). Способ связывания зависит от типа процедуры-функции.

Результат выполнения процедуры-функции обычно применяется:

— в качестве значения по умолчанию для поля таблицы;

— в качестве значения критерия для запросов или фильтров;

— в качестве содержимого поля.

При создании модулей наряду с процедурами-функциями широко применяются *подпрограммы*. На языке VBA подпрограммы называют *процедурами-подпрограммами* и обозначают ключевым словом Sub. В отличие от процедур-функций (функций) процедуры-подпрограммы (подпрограммы), выполняя свои задачи, могут не возвращать результирующие значения. Подпрограмму можно запускать из других процедур, но ее нельзя вызвать так, как функцию, указав имя в выражении или операторе.

Код VBA (последовательность операторов) выполняется при вызове функции или подпрограммы. Во время вычисления выражения, в котором использована функция, выполняется код этой функции. Кроме того, код можно выполнить:

— вызывая функцию, связанную с некоторым событием, например *On-Print(Печать),*

— вызывая функцию с помощью макрокоманды **ЗапускПрограммы;**

— вызывая функцию или подпрограмму в окне **Immediate.**

Если подпрограмму необходимо связать с событием или вызвать с помощью макрокоманды **ЗапускПрограммы,** следует оформить ее как функцию или построить дополнительные функции вызова подпрограммы.

2.10 Инициализация кода

В Access 2000 существуют следующие способы запуска программ Visual Basic

– **Создание процедуры обработки события.** Процедура обработки события выполняется при выполнении пользователем действия, которое вызывает событие. Например, программу, выполняющуюся при открытии формы или отчета, включают в процедуру обработки события Click (Нажатие кнопки) для кнопки, при нажатии которой будет открываться эта форма или отчет.

– **Функция в выражении.** Вызов функции в выражении или в окне проверки редактора Visual Basic. Например, функции применяются в выражениях, определяющих вычисляемые поля в формах, отчетах или запросах. Выражения используются для указания условий в запросах и фильтрах, а

также в макросах и макрокомандах, в инструкциях и методах Visual Basic и в инструкциях SQL

– **Вызов процедуры в другой процедуре** или в окне проверки редактора Visual Basic. Если некоторая программа выполняется часто, можно поместить ее в процедуру **Sub.** Это позволяет не повторять программу Visual Basic выполняющую нужную операцию, в каждой процедуре, а написать ее один раз в общей процедуре, и затем выполнять эту общую процедуру каждый раз, когда требуется выполнить данную операцию

– **Запуск в редакторе VBA.** Выбор команды **Run Sub/UserForm** в меню **Run** редактора Visual Basic для запуска процедуры без аргументов. В окне программы поместите курсор в процедуру, которую требуется выполнить а затем выберите команду **Run Sub/UserForm** из меню **Run.**

– Макрокоманда **ЗапускПрограммы** в макросе. Макрокоманда **ЗапускПрограммы,** выполняемая в макросе, вызывает встроенную функцию Visual Basic или функцию, созданную пользователем. Для запуска процедуры Sub или процедуры обработки события следует создать функцию, которая вызывает процедуру Sub или процедуру обработки события, и с помощью макрокоманды **ЗапускПрограммы** вызвать эту функцию.

Пример 1

Откройте любую базу данных. Выберите объект базы данных – ***Модуль***. Нажмите кнопку ***Создать***. В меню ***Вставка*(*Insert*)** выберите – ***Процедура(Procedure)***. Укажите ***имя(Name)*** процедуры и ***тип(Type)*** вставляемого элемента (функция-Function или программа-Sub). Укажите: общая (Public) – процедура доступна для всех процедур во всех модулях или личная (Private) – процедура доступна для других процедур только в том модуле, в котором она объявлена. Объявите переменные a,b,c – целого типа:

```
dim a,b,c as integer
```

Введите переменные a,b:

```
a=2
b=500
```

или

```
a=InputBox(«Введите a»,»Ввод данных»)
b= InputBox(«Введите b»,»Ввод данных»)
```

Вычислите c и выведите результат:

```
c=a*b
MsgBox "Результат c = " & c
```

Сохраните модуль, выбрав в пункте меню File пункт Save

Для выполнения модуля выберите в пункте меню Run команду Run Sub.

Если необходимо отредактировать модуль, откройте его в режиме Конструктора и отредактируйте.

Контрольные задания

Выполнить индивидуальное задание, выбрав в нижеследующей таблице вариант в соответствии с номером студента в журнале. Для исходных данных создать два модуля (в первом модуле ввод данных выполнить с помощью оператора присваивания , во втором модуле ввод данных выполнить с использованием функции InputBox).

0	$y=\begin{cases}-x^2+2x, & \text{если } x<20\\ 5-11x^2, & \text{если } x=22\\ 2x^2-7ч, & \text{если } 33<x<52\\ \lvert x-2x^2\rvert, & \text{если } x\ge 82\end{cases}$	5	$y=\begin{cases}-x^2+2x, & \text{если } x<20\\ 5-11x^2, & \text{если } x=22\\ 2x^2-7ч, & \text{если } 33<x<52\\ \lvert x-2x^2\rvert, & \text{если } x\ge 82\end{cases}$
1	$y=\begin{cases}2x^2+3x, & \text{если } x<20\\ 5-5x^2, & \text{если } x=25\\ 2x^2-8, & \text{если } 30<x<50\\ \lvert x-2x^2\rvert, & \text{если } x\ge 78\end{cases}$	6	$y=\begin{cases}x^2+11x, & \text{если } x<2\\ 5-3x^2, & \text{если } x=2\\ 12x-8, & \text{если } 3<x<50\\ \lvert 3x-2x^2\rvert, & \text{если } x\ge 85\end{cases}$
2	$y=\begin{cases}12x^2+5x, & \text{если } x<-20\\ 5-11x^2, & \text{если } x=-5\\ 16x-15, & \text{если } -3<x<5\\ \lvert 25x-2x^2\rvert, & \text{если } x\ge 7\end{cases}$	7	$y=\begin{cases}4x^2+11x, & \text{если } x<2\\ 5-8x^2, & \text{если } x=2\\ 2x-5x, & \text{если } 6<x<25\\ \lvert x-4x^2\rvert, & \text{если } x\ge 26\end{cases}$
3	$y=\begin{cases}5x^2-7x, & \text{если } x<-2\\ 5-2x^2, & \text{если } x=2\\ 3x+11, & \text{если } 3<x<5\\ \lvert x-15x^2\rvert, & \text{если } x\ge 8\end{cases}$	8	$y=\begin{cases}x^2-11x, & \text{если } x<10\\ 5+12x^2, & \text{если } x=12\\ 32x-18, & \text{если } 15<x<50\\ \lvert 15x+22x^2\rvert, & \text{если } x\ge 60\end{cases}$
4	$y=\begin{cases}3x^2+x, & \text{если } x<2\\ 15-x^2, & \text{если } x=2\\ 2x-11, & \text{если } 3<x<5\\ \lvert 5x-12x^2\rvert, & \text{если } x\ge 8\end{cases}$	9	$y=\begin{cases}x^2+11x, & \text{если } x<20\\ 5-x^2, & \text{если } x=25\\ 2x-8, & \text{если } 30<x<50\\ \lvert x-2x^2\rvert, & \text{если } x\ge 78\end{cases}$

3 УПРАВЛЕНИЕ БАЗОЙ ДАННЫХ

3.1 Администратор базы данных

Администратор базы ***данных*** - это лицо или группа лиц, в обязанности которых входят: определение информационного содержания базы данных, внутренней структуры, сохранения и стратегии доступа к базе данных, обеспечение охраны информации, ее целостности, контроль производительности работы базы данных и реагирование на требования пользователей.

База данных функционирует в соответствующей социальной среде со множеством пользователей. Пользователи могут предъявлять противоречивые требования к базе данных, поэтому нужно постоянно искать компромиссное решение. База данных находится под централизованным управлением. Архитектура СУБД создается по концепции многоуровневой организации. Четкость и эффективность ее работы обеспечиваются администрированием базы данных.

Функции администратора баз данных объединены в следующие группы: администрирование предметной области, баз данных, безопасности данных и приложений. Поддерживаются функции администрирования специальными служебными программами-утилитами. Для администрации баз данных предусмотрены соответствующие инструкции.

Пользователь базы данных

Пользователь (user) — это физическое или юридическое лицо, которое пользуется услугами компьютерной системы для получения информации либо решения соответствующих задач. Существуют различные *категории пользователей:* аналитик, системный программист, прикладной программист, администратор системы, оператор компьютера, конечный пользователь (потребитель — получает информацию — или источник данных— решает прикладную задачу). Пользователями базы данных могут быть прикладные программы и программные комплексы. Доступ пользователя к базе данных обеспечивается СУБД.

Просмотр и печать сведений о базе данных и ее объектах

Просмотр списка объектов текущей базы данных. Для просмотра всей иерархии объектов файла базы данных следует при открытой базе данных дать команду **«Файл» → «Свойства базы данных»** →вкладка «Состав».

Просмотр объектов. Скрытые объекты по умолчанию можно отобразить в окне базы данных, не отменяя их атрибуты скрытия:

- команда **«Сервис» → «Параметры»→[**Параметры];
- вкладка **«Вид»;**
- установить или снять флажок «Скрытые объекты».

Отображение или скрытие системных объектов в окне базы данных по умолчанию. Дать команды «Сервис»→«Параметры»→выбрать вкладку

«Вид» ~* установить или снять флажок «системные объекты». Напомним, что при создании новой базы данных автоматически создаются системные объекты, чьи имена начинаются символом «Usys».

Просмотр и печать характеристик объекта базы данных выполняется командами пункта **«Сервис»:**

- команды «Сервис»→ «Анализ»→ «Архивариус»→ [Архивариус];
- вкладка соответствующего типа объекта, который желательно рассмотреть или вывести на печать (для полного списка объектов — вкладка **«Все типы объектов»**);
- выбрать объекты для просмотра (печати);
- нажать кнопку **«Параметры»→ «ОК» →«ОК»;**
- Выполнить требуемое действие: для печати — кнопка **«Печать»** на панели инструментов, для сохранения описания в виде таблицы — в меню **«Файл» → «Сохранить как таблицу».**

3.2 Резервирование базы данных

Резервирование есть процесс создания копии программы, базы данных для создания архива или сохранения особенно ценных файлов. Резервирование как процесс является стратегией в современных СУБД: восстановление в базе данных последней операции после того как определенная аппаратная либо программная ошибка сделала базу данных непригодной для работы. Этот процесс начинается с последней, резервной, копии базы данных.

Создание резервной копии базы данных:

1 закрыть базу данных. При работе в сети убедиться в том, что все пользователи закрыли базу данных;

2 скопировать файл базы данных в резервный каталог папки «Мой компьютер» с помощью проводника Windows или программой архивации файлов MS Backup, либо другой программой архивации.

Для архивации отдельных объектов базы данных (таблиц, запросов, форм, отчетов, макросов, модулей) можно создать пустую базу данных, а затем импортировать в нее нужные объекты из исходной базы данных.

Понятие репликации базы данных

Репликация (replication — копия, копирование) — это создание не одной копии, а набора копий базы данных.

Репликация позволяет пользователям, работающим за разными компьютерами, обмениваться изменениями, вносимыми в базу данных, и осуществлять одновременный доступ к собственным копиям. Изменения, вносимые в основную реплику, распределяются по всем реплицируемым объектам — таблицам, формам, запросам, отчетам, макросам, модулям.

В процессе работы с репликами (копиями базы данных) можно синхронизировать данные с помощью специальных программ Access. Синхронизи-

ция (synchronization) — организация действий в определенной последовательности или параллельно—поддерживает целостность базы данных при ее использовании многими пользователями, обеспечивает временное упорядочение действий параллельных процессов с базой данных. .

Отдельные экземпляры копий базы данных используются различными пользователями: можно взять базу данных с собой в командировку на портативном компьютере, распространить базу данных по многочисленным филиалам фирмы и др. После работы с копиями базы данных ее можно синхронизовать.

В создаваемом наборе копий базы данных одна копия является *основной репликой* (копией) — именно в основную копию разрешается вносить изменения (в структуру таблиц, форм, запросов, отчетов, макросов и модулей). На основе любой реплики можно создавать новые реплики.

Копии базы данных могут находиться на исходном компьютере и на любом компьютере локальной сети, а также на компьютере, подключенном через Internet к локальной сетке.

С копиями базы данных можно работать, изменять данные. Для поддержания базы данных, которая в виде ее копий используется многими пользователями, выполняется ее синхронизация. Если в одной и той же записи в две реплики внесены различные изменения, то возникает конфликтная ситуация, которую помогает разрешить синхронизация.

Для организации работы с реплицированной базой данных, отслеживания вносимых изменений в структуру и данные Access автоматически добавляет скрытые системные таблицы.

Перед преобразованием базы данных в основную реплику рекомендуется сделать в другой папке запасную копию базы данных. Если этого не сделать, то Access сам предложит создать резервную копию базы данных (.bak). Реплики можно создавать, только имея право на открытие базы данных в монопольном режиме.

Для *преобразования базы данных в основную реплику и создания реплики следует:*

- открыть базу данных;
- дать команды **«Сервис» → «Репликация» → «Создать реплику»→** [ ]

В окне диалога система предупреждает, что перед созданием реплики базу данных необходимо закрыть, и настоятельно рекомендуется создать резервную копию базы данных перед ее преобразованием. После создания реплики Access выдаст следующее сообщение: «Приложение Access преобразовало (имя базы данных.mdb) в основную реплику набора». Изменения структуры базы данных допускаются, как уже упоминалось, только в основной реплике; изменение данных могут выполняться как в основной, так и в любой другой реплике набора.

Дополнительные реплики создаются той же командой **«Создать реплику».**

Синхронизация реплик— это процесс обновления двух реплик при взаимной передаче обновленных записей и объектов. Реплики обмениваются данными как в одностороннем, так и двустороннем порядке.

После внесения изменений в структуру основной реплики (например, перед печатью объектов базы данных) следует проводить синхронизацию данных в основной и не основной реплике: по команде **«Синхронизация»** происходит обмен данными между элементами набора реплик.

Последовательность действий:

- открыть основную или любую другую реплику;
- команда **«Сервис»→ «Репликация»→ «Синхронизация».**

Конфликты при синхронизации базы данных просматриваются и устраняются командой **«Сервис»→ «Репликация» →«Устранить конфликты».** Выбирается нужный вариант, например из двух различных изменений по одной и той же записи.

Выход из конфликтной ситуации между элементами набора реплик:

- открыть проверяемую базу данных;
- указать таблицу-источник, которая могла вызвать конфликт, дать команду **«Сервис» → «Репликация» → «Удалить конфликт»;**
- при наличии конфликта система задает вопрос о необходимости его разрешения; ответить **«Да»;**
- для каждой конфликтной записки принять решение и нажать соответствующую кнопку: **«Сохранить существующую»** или **«Заменить на конфликтную».**

3.3 Оптимизация базы данных

Для повышения эффективности работы с базой данных разрабатываются и применяются специальные программные средства. Access анализирует быстродействие базы данных с помощью Мастера анализатора.

Для запуска Мастера анализатора нужно при открытом окне базы данных дать команды: **«Сервис»→ «Анализ»→ «Быстродействие»** →[окно Мастера] (рисунок 3.1).

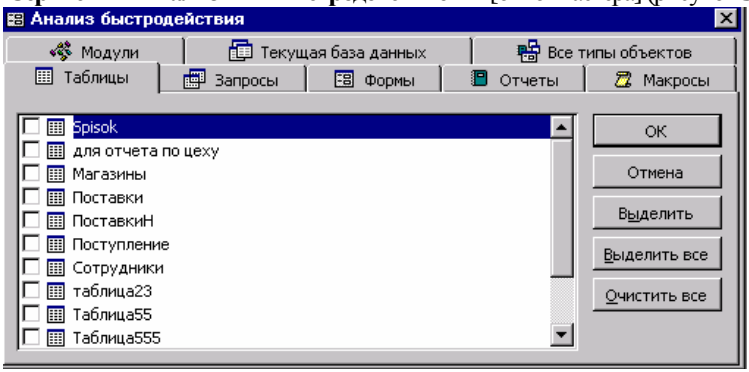

Рисунок 3.1 –. Начальное окно анализатора быстродействия

Затем выбрать категорию анализируемых объектов (текущая база данных, таблицы, отчеты и др.) → указать конкретный объект путем установки флажков рядом с их именами → для запуска анализатора быстродействия щелкнуть кнопку **«ОК».** Мастер анализатор выполнит задание и выдаст сообщение (рисунок 3.2) о результатах анализа в виде списка пожеланий и предложений и «мыслей».

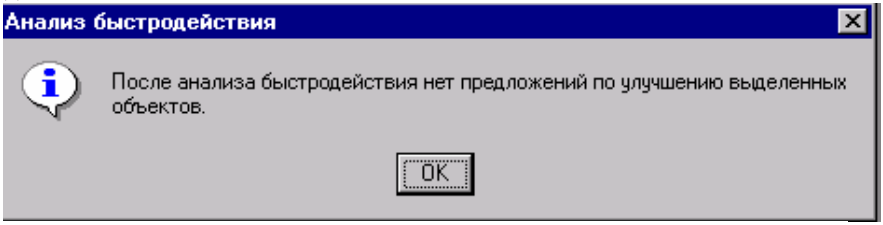

Рисунок 3.2 – Окно анализатора быстродействия с результатами анализа

Выбрать (выделить) подходящее предложение и нажать в окне анализатора кнопку **«Оптимизировать»** — Мастер оптимизатор внесет в базу данных соответствующие изменения.

К «мыслям» анализатора следует относиться критически, потому что принятие отдельных «мыслей» может привести к появлению большого объема дополнительной работы и мизерному повышению быстродействия.

3.4 Восстановление поврежденной базы данных

СУБД Access следит за состоянием базы данных: при появлении повреждений во время манипуляций с ней дает соответствующее сообщение; имеет средства для восстановления базы данных.

Восстановление открытой базы данных:

команды: **«Сервис» → «Служебные программы» →«Восстановить базу данных»→** выдастся сообщение об успешном восстановлении базы данных и указывается путь к базе данных.

Восстановление неоткрытой базы данных:

1 закрыть активную базу данных;

2 подать команды **«Сервис»→ Служебные программы» → «Восстановить базу данных»→** [Восстановление базы данных];

3 указать имя восстанавливаемой базы данных и каталог (папку);

4 нажать кнопку **«Восстановить».**

3.5 Защита базы данных

Для предотвращения несанкционированного доступа к объектам базы данных предназначены средства защиты информации, которые входят в СУБД. С их помощью системе указываются пользователи, которым дано или не дано разрешение на доступ к объектам базы данных.

При большом количестве пользователей можно определить их группы и разрешить им доступ к базе данных (рисунок 3.3). Стандартный файл рабочей группы (System.mdw) создается при установке Access программой Setup. Файл System.mdw содержит информацию обо всех пользователях и группах и применяется по умолчанию при запусках Access. Последний сохраняет информацию о правах доступа при создании базы данных в файле базы данных. Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.

База данных, созданная пользователем рабочей группы, принадлежит ее создателю и рабочей группе. Каждая рабочая группа и каждый пользователь имеют уникальные внутренние идентификаторы. При копировании базы данных в другую папку или компьютер вместе с файлом базы данных перемещаются и права на доступ.

Компьютерные системы защиты информации бывают открытыми или закрытыми. В открытой системе информация открыта для всех пользователей (кроме информации, имеющей специальную защиту), в закрытой — только по назначении доступа.

Существуют два способа защиты базы данных: вставка пароля и защита на уровне пользователей.

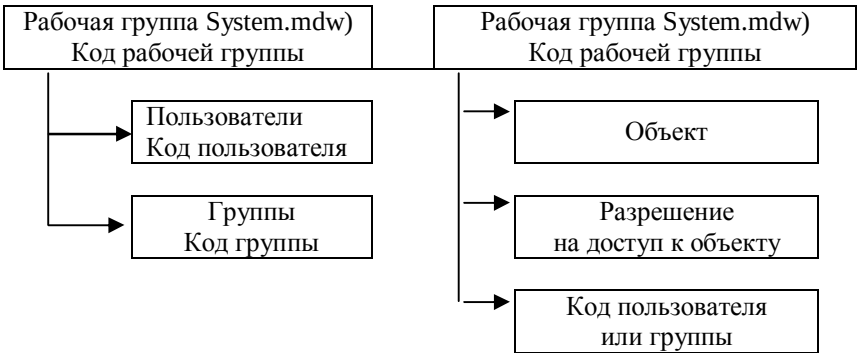

Рисунок 3.3 – Схема защиты базы данных

Для открытия базы данных можно вставить пароль. После этого перед каждым открытием базы данных появляется окно для ввода пароля, который действует только при открытии базы данных. После ее открытия все объекты базы данных становятся доступными.

Установка пароля:

1 закрыть базу данных; при работе в сети и в ней закрыть базу данных;
2 создать резервную копию базы данных и сохранить ее;
3 на панели инструментов дать команду «Файл» → «Открыть»;
4 установить флажок «Монопольный доступ» и открыть базу данных;

5 **дать команду «Сервис» → «Защита» → «Задать пароль базы данных»;**

6 ввести пароль в поле **«Пароль»** (с учетом регистра);

7 подтвердить пароль повторным его вводом в поле **«Подтверждение»** и нажать кнопку **«ОК».** Пароль установлен. При открытии базы данных будет появляться окно диалога для ввода пароля.

Защита базы данных на уровне пользователей

Пользователю нужно идентифицировать себя и ввести пароль. Пользователи идентифицируются как члены группы в файле рабочей группы. *Рабочая группа* — это группа пользователей, которые в сети совместно пользуются базой данных. Файл рабочей группы содержит данные о пользователях группы и считывается во время запуска. Этот файл содержит следующую информацию: имена учетных записей пользователей, пароли пользователей, имена групп, в которые входят пользователи.

При установке MS Access создается стандартная рабочая группа, которая содержит один встроенный код пользователя с названием Admin и два встроенных кода группы (User и Admins). Access автоматически загружает пользователя с его кодом и предоставляет все права и привилегии.

Встроенный код группы User включает в себя всех пользователей (членов) группы и автоматически представляет полные права доступа к объектам базы данных. Пользователям предоставляется разрешение на доступ к каждому объекту базы данных.

Для каждого файла рабочей группы при создании файла устанавливается ее уникальный внутренний идентификатор. По умолчанию в эту группу включен только пользователь Admin. Пользователи группы Admin могут определять и изменять учетные записи пользователей и групп, устанавливать и менять личные пароли, имеют полный доступ к любым базам данных, созданным при использовании этого файла рабочей группы. Любая база данных, которая создана с применением конкретного файла этой, группы, наследует ее код.

Типы разрешений, которые могут быть назначены базе данных или ее объектам: открытие/запуск (Open I Run), монопольный доступ (Open | Exclusive), чтение макета (Read | Desing), изменение макета (Modify Desing), разрешение администратора (Administer), чтение данных (Read Data), обновление данных (Updata Data), вставка данных (Insert Data), удаление данных (Delete Data).

Проверка разрешения пользователю или группе на доступ к объектам базы данных выполняется в такой последовательности:

1 открыть нужную базу данных. Необходимо быть собственниками базы данных и ее объектов или иметь разрешение администратора на доступ;

2 дать команду **«Сервис»→ «Защита»→ «Разрешение»** → [Разрешение]. В окне имеется две вкладки: «Разрешение» и «Смена владельца». Вы-

брать вкладку **«Разрешение».** В окне будут отображены пользователи и группы, которые определены в базе данных;

3 выбрать **«Admin»** (группа) или **«Users»** (пользователи) — в списке «Имя объектов» будут выведены объекты базы данных. В окне можно изменить тип объекта базы данных в раскрывающемся списке «Тип объекта». После этого в части окна **«Разрешения:»** флажки укажут на явные разрешения доступа.

Вкладка **«Смена владельца»** позволяет назначить нового владельца (пользователя или группу).

Мастер защиты базы данных

Для установки защиты базы данных на уровне пользователя Access имеет Мастер защиты. Перед непосредственным запуском Мастера защиты необходимо:

- зарегистрироваться под именем владельца базы данных или в файле рабочей группы, который применялся при создании базы данных, и быть членом группы Admins;
- сформировать новую рабочую группу с уникальным кодом, отличным от кода рабочей группы, где создавалась база данных;
- ввести новый код пользователя, отличный от Admin (для владельцев базы данных Admin нельзя установить защиту). Если пользователь зарегистрирован в первичной рабочей группе как пользователь Admin, то Мастер предложит создать новую рабочую группу;
- удалить для всех объектов разрешения с группы Users. *Запуск мастера защиты данных:*

1 дать команду **«Сервис»→«Защита»→«Мастер»**→первое окно диалога *Мастер защиты.* Здесь содержится краткое описание работы Мастера и предлагается два варианта: «Создать файл рабочей группы» или «Изменить текущий файл рабочей группы». При выборе опции «Создать файл рабочей группы» Мастер создаст новый файл рабочей группы, создаст в этой рабочей группе новый код пользователя (не Admin) и установит защиту для базы данных в новой рабочей группе и нового пользователя;

2 перейти во второе окно Мастера путем нажатия кнопки **«Далее»** →[Мастер защиты]. По умолчанию Мастер защищает все объекты базы данных. Можно отменить (снять флажок) защиту отдельных объектов;

3 нажать кнопку **«Далее»** — перейти в третье окно. Мастер предлагает создать одну или несколько дополнительных групп. В списке имен групп можно выбрать нужную группу и увидеть разрешения на доступ для этой группы;

4 нажать кнопку **«Далее»** — перейти в четвертое окно Мастера, где представлены разрешения универсальной группы Users;

5 нажать кнопку **«Далее»** — перейти в пятое окно Мастера и добавить пользователей (если нужно) в файл рабочей группы; для каждого пользователя задать пароль и уникальный личный код;

6 нажать кнопку **«Далее»** — перейти в шестое окно Мастера, где можно включить пользователей в нужные группы;

7 нажать кнопку **«Далее»** — перейти в седьмое окно Мастера, где задать имя для резервной копии файла базы данных;

8 нажать кнопку **«Готово»** для завершения работы Мастера.

3.6 Публикация данных в WEB

В настоящее время Web-технология в среде Internet становится доступной всем пользователям, желающим получить и распространить информацию. MS Access 2000 имеет средства для создания Web-приложен и баз данных. Web – это система, основанная на механизме гиперссылки, предназначенная для поиска данных в Internet и работы с данными.

Публикация статических данных

Общая схема приема запроса от пользователя (броузера) Web-сервером и пересылки ему затребованных файлов приведена на рисунке 3.4.

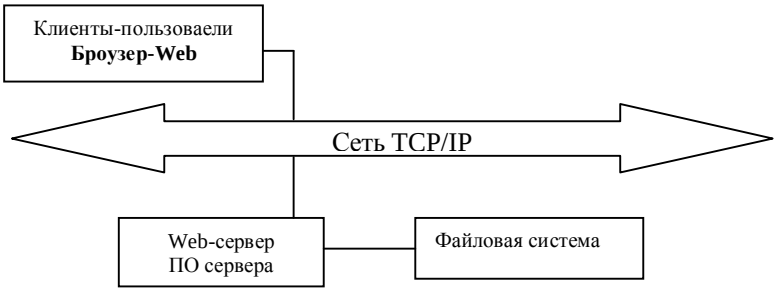

Рисунок 3.4 – Упрощенная схема функционирования WWW

Сетевой протокол определяет тип пакетов данных. В качестве стандарта в WWW принят протокол TCP/IP (Transmission Control Protocol/ Internet Protocol) — протокол обмена информацией, понятный всем компьютерам сети. Протокол TCP описывает способ обмена данными между компьютерами в Internet, а протокол IP — адресацию (индивидуализацию) компьютеров в Internet.

Основными протоколами в Web для отправки и получения информации являются HTTP (Hipertext Transport Protocol) — для передачи Web-страниц — и FTP (File Transfer Protocol)—для передачи файлов.

Web-страница содержит текстовые файлы и коды форматирования. Страницы с текстовыми файлами называются *статическими*. Форматирование текста выполняется HTML-файлами (Hypertext Markup Language). Редактирование Web-страницы, которая является программной средой для доступа к данным, происходит в HTML-файле.

Данные в Access публикуются в Web-страницах только в виде статических страниц. Web-страница создается в формате HTML. Язык HTML — это система разметки документов для их последующей публикации в сети WWW. Заметим, что Web-страница, созданная в этом формате, не может обновляться — она отражает текущее состояние данных. Обновить страницу можно только повторением процедуры ее экспорта после каждого изменения базы данных.

Создание статических страниц:

1 выбрать в окне базы данных объект для публикации; если он уже открыт, то сделать окно объекта активным;

2 дать команды **«Файл»** → **«Экспорт»** → [Экспорт объекта]

3 в поле «Тип файла» в раскрывшемся списке выбрать пункт «Документы HTML»;

4 в поле «Папка» указать путь сохранения. При отсутствии права доступа к общей папке на Web-сервере выбирается временная рабочая область на компьютере пользователя; •

5 имя файла присваивается автоматически в окне **«Имя файла»**, но его можно изменить: расширение файла .html;

6 установить флажок «Сохранить формат»;

7 щелкнуть кнопку **«Сохранить» →[ ];**

8 в окне диалога **«Параметры вывода в формате HTML»** можно оставить пустым поле **«Шаблон HTML»** или указать имя файла шаблона;

9 нажать кнопку **«ОК».** Web-страница создана.

Созданную для широкого использования Web-страницу надо сохранить (опубликовать) на Web-сервере. Копирование Web-страниц на Web-сервер выполняет программа Front Page Server Extensions, установленная на Web-сервере.

Последовательность действий при сохранении публикации Web-страниц на Web-сервере: .

1 на рабочем столе открыть папку «Мой компьютер» (дважды щелкнуть);

2 открыть Web-папку →[Web-папки];

3 из содержимого Web-папки выбрать нужный Web-сервер. Если его нет в этом окне, то в этом же окне дважды щелкнуть по ярлыку «Добавление папки Web». Ввести URL сервера; задать имя для сервера и нажать кнопку **«Готово»;**

4 последовательно открыть папки Web-сервера, пока не будет найдена папка, в которую решено поместить Web-страницу;

5 двойным щелчком открыть папку «Мой компьютер» или «Сетевое окружение» и найти копируемый файл (созданную Web-страницу);

6 перетянуть файл из Windows Проводника в окно >Ш>-папки.

В окне Web-папки выполняются операции копирования файлов с Web-сервера на Web-броузер, удаления файлов. После публикации на Web-сервере в поле «Адрес» зафиксирован адрес URL.

Web-папки являются компонентами Office 2000 и частью клиентского программного обеспечения.

Динамические страницы

Схема выполнения динамического запроса к базе данных в отличие от выполнения статического запроса более расширена(рисунок 3.5). В этом случае Access выполняет функцию файлового сервера баз данных для публикации текущего содержимого базы данных В динамических страницах отображаются последние изменения в базе данных — статические же страницы связаны с базой данных.

Для получения пользователями актуальных данных, отражающих последние изменения в базе данных, создаются динамические страницы ASP (Active Server Pages). Web-страницы ASP включают в себя команды VBScript, которые выполняются на Web-сервере.

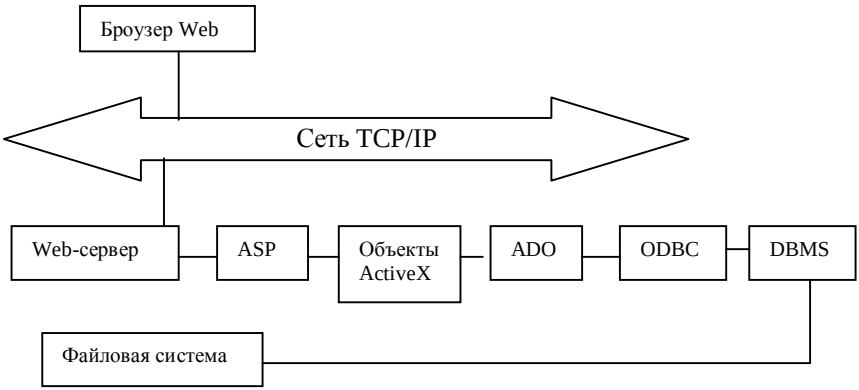

Рисунок 3.5 – Общая схема выполнения динамического запроса к базе данных

ASP — это Wsb-страница специального типа, которая состоит из HTML и команд Visual Basic.

Объекты ActiveX—это модули программного обеспечения для отображения в формах диаграмм, картинок, выполнения вычислений над данными таблиц или запросов.

ADO (ActiveX Data Objects) — это система объектов данных. Можно использовать как универсальный интерфейс для открытия набора записей, работы с ними и выполнения запроса.

ODBC (Open Database Connectivity) — стандарт открытого доступа к данным, формализованный интерфейс обмена данными между различными СУБД.

DBMS (Database Management System) — система управления базами данных, набор программ, которые позволяют создать базу данных и работать с ней. Это программная оболочка, находящаяся между собственно базой данных и пользователем. DBMS управляет всеми запросами пользователя на те или иные действия, которые надо выполнить в базе данных.

При работе с динамическими страницами броузер Web не выходит прямо на файловую систему сервера, а открывает Web-страницу специального типа (ASP). Команды Visual Basic открывают базу данных, выполняют запрос н формируют его результат. Web-сервер отправляет сформированный результат запроса броузеру, открывшему эту страницу.

Таким образом, для публикации текущего содержимого базы данных Access в этой схеме выступает в качестве файлового сервера баз данных. Основная работа со страницей доступа к данным выполняется в Web-броузере. Необходимые компоненты ActiveX устанавливаются на Web-броузере.

Страница доступа к данным в MS Access 2000 является одним из объектов базы данных и обеспечивает доступ к данным в Internet. С помощью этой страницы Internet Explorer находит и просматривает данные в базе данных Access. В корпоративной интрасети публикуются страницы доступа к данным в виде Web-страниц, а пользователи, у которых установлен Office 2000 и Internet Explorer не ниже 3-й версии, получают доступ к этим данным (т. е. имеют возможность находить, изменять и просматривать их). Чтобы открывать Web-страницы доступа к данным, пользователю интрасети нужно только иметь установленный MS Office 2000.

Чтобы увидеть страницы доступа к данным, надо в окне базы данных щелкнуть кнопку **«Страницы»;** откроется список страниц доступа к данным в базе данных. В окне базы данных в левом верхнем углу расположены три кнопки:

«Открыть»—открывается существующая страница доступа к данным;

«Конструктор» — открывается страница в режиме Конструктора;

«Создать» — создается новая страница доступа к данным с помощью Мастера или самостоятельно.

Для создания объектов окно базы данных содержит три ярлыка (для запуска нужно дважды щелкнуть по ярлыку):

- создание страницы доступа к данным в режиме Конструктора;
- создание страницы доступа к данным с помощью Мастера;
- изменение существующей Web-страницы.

Страницы доступа к данным есть способ разработки приложений и их распространение.

Создание динамических страниц:

1 выделить в окне базы данных объект; если он уже открыт, то сделать его окно активным;

2 дать команду **«Файл»→ «Экспорт»** →[Экспорт объекта];

3 в поле **«Тип файла»** в раскрывшемся списке выбрать пункт **«Microsoft Active Server Page»;**

4 в поле **«Папка»** указать путь сохранения (временная рабочая область на компьютере пользователя);

5 имя файла можно изменять: расширение файла (Web-страницы) .asp;

6 щелкнуть кнопку **«Сохранить»** →[Настройка выводов файлов ASP];

7 в поле **«Название источника»** обязательно ввести имя файлового источника данных стандартного интерфейса (ODBC);

8 **нажать кнопку «OK»; Web-страница сохранена.**

Создание Web-страниц с помощью Мастера:

1 в окне базы данных выбрать кнопку **«Страницы»**, а затем ярлык «Создание страницы доступа к данным с помощью Мастера»; откроется Окно **«Мастер страниц»,** инициализируется среда мастера;

2 в первом окне Мастера страниц в поле **«Таблицы и запросы»** выбрать таблицу-источник или запрос;

3 в списке **«Доступные поля»** выбрать нужные поля и клавишами навигации перенести их в список «Выбранные поля»;

4 нажать кнопку **«Далее»** — перейти во второе окно Мастера, где можно указать уровень группировки. Заметим, что поле задания уровня группировки делает страницу необновляемой. Поэтому, не задавая группировки, переходим в третье окно Мастера;

5 при необходимости на третьем шаге задать порядок сортировки набора записей;

6 в четвертом (последнем) окне Мастера ввести название создаваемой страницы — в начале страницы доступа к данным. В этом окне есть два переключателя: «Открыть страницу» — для открытия страницы в режиме страницы — и «Изменить макет страницы» — для открытия в режиме Конструктора;

7 установить флажок «Применить тему к странице» и нажать кнопку **«Готово»** → [Тема];

8 в окне **«Тема»** выбрать любую тему, установленную на компьютере, а также набор цветов, шрифтов и графики. Закончить нажатием кнопки «Готом»».

Открытие страницы доступа к данным

Созданная Web-страница запоминается Access и включается в список объектов базы данных во внешних файлах. Открыть Web-страницы можно: .

- классическим способом: дважды щелкнуть по ее имени в окне базы данных;

- командой **«Файл» → «Открыть»**→[Открытие файла БД]. Выбрать пункт «Web-страницы» {Журнал, Мои документы, Рабочий стол, Избранные, Web-папки). Найти страницу и нажать кнопку **«Открыть».**

Страницы, сохраненные в пунктах «Избранное» и «Web-страницы», можно читать в Access (как броузер) с Web-сервера.

Сохранение страниц доступа к данным выполняется по команде «Файл» → «Сохранить» и «Файл» →«Сохранить как».

Специализированный редактор FrontPage предназначен для улучшения внешнего профессионально-эстетического вида. В этом редакторе открываются и редактируются файлы HTML и Active Server Page на локальном диске или на Web-сервере.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Microsoft Access 2000: справочник. Под ред. Ю. Колесникова. – СПб.: Питер, 1999.
2. Вейскас Дж. Эффективная работа с Microsoft Access 2000. – СПб.: Питер, 2000.
3. Винтер П. Microsoft Access 97: справочник. – СПб.: Питер, 1998.
4. Информатика для юристов и экономистов / Симонович С.В. и др. – СПб: Питер, 2001.
5. Информатика: Учебник / Под ред. проф. Н.В. Макаровой. – 2-е изд. — М.: Финансы и статистика, 1998.
6. Новейший самоучитель работы на компьютере. Под ред. С.В. Симоновича. – М.: Десс; Инфорком-Пресс, 1999.
7. Основы экономической информатики: Учеб. пособие/ А.Н. Морозевич, Н.Н. Говядинова, В.Г. Левашенко, Б.А. Железко и др.; Под ред. проф. А.Н. Морозевича. – Мн.: Новое знание, 2001.
8. Савицкий Н.И. Технологии организации, хранения и обработки данных: Учеб пособие. – М.: ИНФРА-М, 2001.
9. Хансен Г., Хансен Дж. Базы данных: разработка и использование: Пер. с англ. – М.: БИНОМ, 1999.
10. Экономическая информатика: Учебник для вузов / Под ред. В.В. Евдокимова. – СПб.: Питер Паблишинг, 1997.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ В ЯЗЫК SQL 3
1.1 Назначение языка SQL 3
1.2 Структура команд языка SQL 4
1.3 Описание таблиц 6
1.4 Манипулирование данными 8
1.5 Формирование запросов (команда SELECT) 10
1.6 Объединение таблицы по принципу »Сама с собой» 16
1.7 Представления 19
1.8 Определение прав доступа к данным 20
1.9 Использование SQL с другими языками программирования 25
2 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРИКЛАДНЫХ ПРОГРАММ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ 26
2.1 Средства автоматизации обработки данных в системе управления базой данных 26
2.2 Создание макроса 28
2.3 Основы программирования на языкеVBA. Основные типы данных и их описание 33
2.4 Математические и логические функции и операторы 36
2.5 Ввод и вывод данных 37
2.6 Понятие процедуры. Типы процедур 37
2.7 Операторы языка VBA 42
2.8 Примеры программы на VBA 47
2.9 Работа в окне редактора кода VBE в Microsoft Access 49
2.10 Инициализация кода 51
3 УПРАВЛЕНИЕ БАЗОЙ ДАННЫХ 54
3.1 Администратор базы данных 54
3.2 Резервирование базы данных 55
3.3 Оптимизация базы данных 57
3.4 Восстановление поврежденной базы данных 58
3.5 Защита базы данных 58
3.6 Публикация данных в WEB 62
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ 67

Учебное издание
Юрий Павлович ЛЫЧ

Технологии организации, хранения и обработки данных
Часть 2. Основы программирования в СУБД ACCESS

Пособие для самостоятельной подготовки к занятиям
Редактор
Технический редактор
Корректор
Подписано в печать г. Формат бумаги 60х84 1/16.
Бумага газетная. Гарнитура Таймс. Печать офсетная.
Усл. печ. л. . Уч.-изд. л. . Тираж 300 экз.
Зак. № 896. Изд. № 3201.
Редакционно-издательский отдел БелГУТа, 246653, г. Гомель, ул. Кирова, 34.
Лицензия ЛВ № 57 от 22.10.97 г.
Типография БелГУТа, 246022, г. Гомель, ул. Кирова, 34.
Лицензия ЛП № 360 от 25.07.99 г.