Паньгина Нина Николаевна

ЗАНЯТИЕ 6. РАБОТА С БАЗАМИ ДАННЫХ СРЕДСТВАМИ VISUAL BASIC

Visual Basic позволяет достаточно легко создавать приложения для Windows, в том числе и программы для работы с базами данных – Databases.

По определению *база данных* (БД) – это совокупность взаимосвязанных, тщательно структурированных данных, собранных с целью хранения, накопления, поиска и выдачи в определенной форме.

Наиболее популярными с начала 80-х гг. стали и до сих пор остаются реляционные (relational) БД. Данные в таких базах хранятся в одной или нескольких таблицах, связанных между собой. Каждая таблица имеет собственный, заранее определенный набор именованных колонок (полей). Поля таблицы обычно соответствуют атрибутам хранящихся данных (сущностей). Количество строк (записей) в таблице может быть достаточно велико, и каждая запись соответствует отдельной сущности.

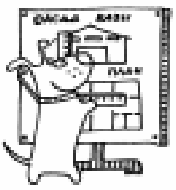
Существуют специальные программные средства для работы с реляционными БД, так называемые системы управления базами данных (СУБД) – DataBase Management System (DBMS). Наиболее распространенными на сегодняшний день являются СУБД Oracle, MySQL, Informix, Visual FoxPro, MSSQL и MS Access. Кроме этого, визуально-ориентированные языки программирования, например, такие как Visual Basic, Delphi, Visual C++, обладают удобными и мощными средствами для работы с базами данных.

Рассмотрим некоторые возможности работы с базами данных в Visual Basic на примере базы данных «Планеты Солнечной системы и их спутники».

РАЗРАБОТКА И СОЗДАНИЕ БД

Разработка и создание базы данных (БД) включает следующие этапы: проектирование БД, формирование и наполнение БД.

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ



Для проектирования БД следует определить:

- какие данные будут храниться в базе;
- табличную структуру данных;
- связи между таблицами.

Информацию для БД «Планеты Солнечной системы и их спутники» можно найти как в международных, так и в Российских Интернет-ресурсах.

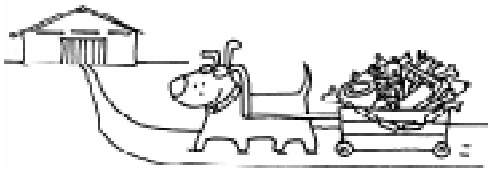
Международные ресурсы:
http://www.solarviews.com/eng/query.htm – планетарный браузер;
http://nssdc.gsfc.nasa.gov/planetary/planetfact.html;
http://seds.lpl.arizona.edu/nineplanets/nineplanets.
Российские ресурсы:
http://ggreen.chat.ru/
http://symbolist.ru/rus/astrology/planets – символы планет;
http://www.astrolab.ru/astrsimv.html – символы планет;
http://cddb.sai.msu.su/ng/main.htm и др.

Выберем из приведенных источников информации три основные таблицы (см. ниже).

Данные таблицы 1 в транспонированном виде (то есть строки таблицы 1) будут составлять поля будущей таблицы

Planet, а столбцы таблицы 1 будут составлять отдельные записи Planet. Рекомендуется для полей определить краткие названия (на английском языке). Используемые далее имена полей таблиц выделены жирным шрифтом.

2. ФОРМИРОВАНИЕ И НАПОЛНЕНИЕ БАЗЫ ДАННЫХ



Создадим новую базу данных (предполагается знание пакета MS Access) Astronomy.mdb. Организуем таблицу **Planet**, импортируя подготовленные данные таблицы 1. Поля Name, Atmosphere – текстовые; Magnetic – логическое поле, остальные числовые.

С предложением Access создать счетчик записей – согласиться и переобозначить поле Number. Это поле необходимо для сортировки планет по их удаленности от Солнца.

Аналогично создадим таблицу спутников планет, используя данные таблицы 3. Исходные данные следует откорректировать, чтобы название планеты присутствовало для каждой записи вместе с названием спутника. Также импортируем данные в таблицу **Satellite** с полями Name_P, Moon, Date (соответственно для названия плане-

Таблица 1. **Характеристики планет Солнечной системы (Planet)**

Название **Name**	**Меркурий**	**Венера**	**Земля**	**Марс**	**Юпитер**	**Сатурн**	**Уран**	**Нептун**	**Плутон**
Масса (10^{24} кг) **Mass**	0.33	4.87	5.97	0.642	1899	568	86.8	102	0.0125
Диаметр (км) **Diameter**	4879	12104	12756	6794	142984	120536	51118	49528	2390
Плотность (кг/м^3) **Density**	5427	5243	5515	3933	1326	687	1270	1638	1750
Гравитация (м/сек^2) **Gravity**	3.7	8.9	9.8	3.7	23.1	9	8.7	11	0.6
Скорость отрыва (км/сек) **Escape_V**	4.3	10.4	11.2	5	59.5	35.5	21.3	23.5	1.1
Длительность дня (час) **Day**	4222.6	2802	24	24.7	9.9	10.7	17.2	16.1	153.3
Перигелий (10^6 км) **Perihelion**	46	107.5	147.1	206.6	740.5	1352.6	2741.3	4444.5	4435
Афелий (10^6 км) **Aphelion**	69.8	108.9	152.1	249.2	816.6	1514.5	3003.6	4545.7	7304.3
Период орбиты (сут) **O_Period**	88	224.7	365.2	687	4331	10747	30589	59800	90588
Орбитальная скорость (км/сек) **Orbital_V**	47.9	35	29.8	24.1	13.1	9.7	6.8	5.4	4.7
Наклон оси (градус) **Tilt**	0.01	177.4	23.5	25.2	3.1	26.7	97.8	28.3	122.5
Средняя температура (С) **Temperature**	167	464	15	–65	–110	–140	–195	–200	–225
Давление на поверх. (атм.) **Pressure**	0	93	1	0.007					0.00001
Магнитное поле **Magnetic**	Да	Нет	Да	Нет	Да	Да	Да	Да	
Компоненты атмосферы **Atmosphere**		CO_2, N_2	N_2, O_2, Ar	CO_2, N_2, Ar	H_2, He	H_2, He	H_2, He, CH_4	H_2, He, CH_4	N_2, CH_4, CO

ты, ее спутника, года открытия спутника). Для корректности импорта таблиц следует проследить, чтобы поля не имели пробелов в начале и внутри названия.

Преобразуем при необходимости полученную БД к версии Access97, для того чтобы в качестве альтернативного средства при создании БД использовать ***Visual Data Manager,*** входящий в состав VB. С его помощью добавим в БД Astronomy таблицу символов планет – **Astrology**.

Откроем новый VB-проект, выберем команду меню Add-Ins → Visual Data Manager. В открывшемся окне (VisData) выберем в меню File → Open Database → Microsoft Access → нашу БД Astronomy.mdb. В окне Database Windows, щелкнув на таблице Planet или Satellite, просмотрим введенную информацию (импорт которой в таблицы БД можно также осуществить через меню File окна VisData).

Таблица 2. **Символы планет (Astrology)**

Планета **Astro**	Символ **Symbol**
Меркурий	☿
Венера	♀
Земля	⊕
Марс	♂
Юпитер	♃
Сатурн	♄
Уран	♅
Нептун	♆
Плутон	♇

Щелкнем правой кнопкой мыши в окне Database Windows, в контекстном меню выберем New Table, введем имя таблицы Astrology, добавим (Add Field) поле Astro (текстовое, размером 10) и поле Symbol (тип binary), образуем таблицу (Build the Table).

Для ввода данных в таблицу Astrology воспользуемся утилитой ***Data Form Designer,*** входящей в состав Visual

Таблица 3. **Спутники планет (Satellite)**

Планета **Name_P**	Спутник **Moon**	Год открытия **Date**
Земля	Луна	
Марс	Деймос	1877
	Фобос	1877
Юпитер	Адрастея	1979
	Амальтея	1892
	Ганимед	1610
	Гималия	1904
	Европа	1610
	Ио	1610
	Каллисто	1610
	Лиситея	1938
	Леда	1974
	Метис	1979
	Пасифе	1908
	Синопе	1914
	Элара	1905
Сатурн	Атлас	1980
	Калипсо	1980
	Диона	1684
	Энцелад	1789
	Эпиметей	1980
	Елена	1980
	Гиперион	1848
	Япет	1671
	Янус	1966
	Мимас	1789
	Пан	1990
	Пандора	1980
	Феба	1898
	Прометей	1980
	Рея	1672
	Телесто	1980
	Тефия	1684
	Титан	1655
Уран	Ариэль	1851
	Белинда	1986
	Бианка	1986
	Корделия	1986
	Крессида	1986
	Дездемона	1986
	Джульетта	1986
	Миранда	1948
	Оберон	1787
	Офелия	1986
	Портия	1986
	Пак	1985
	Розалинда	1986
	Титания	1787
	Умбриэль	1851
	Калибан	1997
	Сикоракс	1997
	Сетебос	1999
	Стефано	1999
	Просперо	1999
	1986U10	1999
Нептун	Деспина	1989
	Галатея	1989
	Ларисса	1989
	Наяда	1989
	Нереида	1949
	Протей	1989
	Таласса	1989
	Тритон	1846
Плутон	Харон	1978

Data Manager, вызовем ее через пункт меню Utility. В окне утилиты зададим имя формы (AstroInput), выберем таблицу данных Astrology и все ее поля. В образованной форме frmAstroInput присутствует автоматически созданный элемент управления данными Data, текстовое поле, объект OLE, а также различные кнопки (рисунок 1), действия которых определены в готовых событийных процедурах.

ОБЪЕКТ УПРАВЛЕНИЯ И КОНТРОЛЯ DATA

Объект управления и контроля (ОУК) Data определяет, какой набор записей, какие поля из таблиц БД будут связаны с элементами управления на форме для отображения выбранной информации. Также он позволяет перемещаться по записям этого набора (Recordset). ОУК Data имеет следующие важные свойства (Property):

- Connect определяет формат данных (в нашем случае Access);
- DatabaseName задает полный путь к файлу БД (Astronomy.mdb);
- RecordSource определяет источник данных – либо имя таблицы (в нашем случае таблица Astrology), либо оператор SQL, идентифицирующий логическое представление данных;
- RecordsetType описывает тип набора данных. В VB имеется три типа наборов данных:

Table (таблица) – определяет записи только из одной таблицы БД;

Dynaset (динамический набор) – обеспечивает доступ к полям и записям нескольких таблиц БД, для этого используются фильтры и метод поиска Find;

Snapshot (статический набор) – копия данных из таблиц, хранится в памяти, используется только для чтения.

Recordset – «множество записей» виртуальной таблицы БД, определяется состоянием свойства RecordSource.

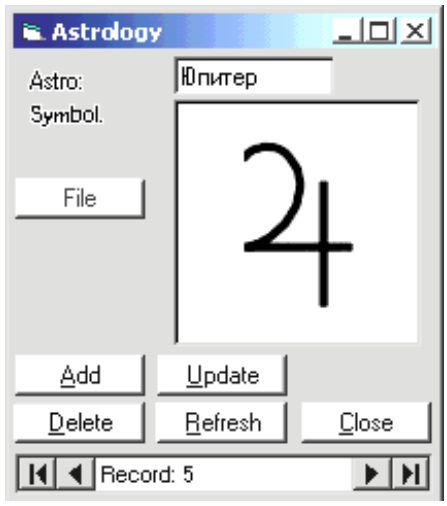


Рисунок 1

Вернемся к заданию ввода данных в таблицу Astrology. Удалим автоматически созданный OLE-объект на форме, а вместо него введем ОУК PictureBox и свяжем его с элементом управления Data.

Любые элементы управления, будь то текстовое окно или метка для отображения числовой или текстовой информации, окно рисунка или окно изображения для отображения графической информации, флажок, позволяющий отображать булевские (логические) величины, – все они подключаются к объекту Data через свойство ***DataSource*** (источник данных). Свойство же ***DataField*** (поле данных) определяет имя поля в таблице, с которой связывается конкретный элемент управления.

Зададим предлагаемый в свойстве DataSource элемент управления данными с именем Data1, а в свойстве DataField укажем для связи поле Symbol. Зададим значение свойства AutoSize=True.

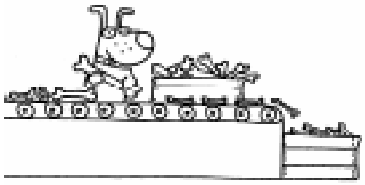
Поместим на форму элемент управления CommonDialog (включив его предварительно в список компонентов). Он будет использоваться для вызова диалогового окна открытия файла рисунка с символом планеты. Добавим также кнопку с названием «File» и процедуру обработки щелчка мыши на кнопке для загрузки выбранной картинки в PictureBox:

```
Private Sub Command1_Click()
   CommonDialog1.ShowOpen
   Picture1.Picture = LoadPicture_
   (CommonDialog1.FileName)
End Sub
```

В основном меню VB Project → Properties выберем объектом начального запуска (Startup Object) форму frmAstroInput. Запустим приложение и, с помощью кнопки Add, заполним последовательно таблицу Astrology базы данных названиями планет, а также символами планет (из соответствующих заранее подготовленных файлов рисунков) из связанных элементов управления TextBox и PictureBox.

На этом заканчивается этап формирования и наполнения базы данных «Планеты Солнечной системы и их спутники».

РАБОТА С БАЗОЙ ДАННЫХ



Работа с базой данных включает в себя следующие возможности: простейший просмотр данных, корректировка данных (добавление, удаление, изменение данных), поиск данных по различным критериям, формирование различной сложности запросов на выборку, вывод отобранных данных на печать.

В предыдущем разделе посредством Visual Data Manager, а точнее, Data Form Designer, мы научились создавать формы для просмотра информации из базы данных. В эти стандартные формы имеется возможность автоматически помещать кнопки, позволяющие вызывать соответствующие методы набора данных RecordSet элемента управления Data:

- Add – для добавления записей в БД (метод AddNew);
- Delete – для удаления записей из БД (метод Delete);
- Update – для обновления текущей записи в таблице (метод Update вызывать не обязательно, обновление происходит автоматически при переходе на другую запись);
- Refresh – для обновления таблицы в многопользовательском режиме ее использования (метод Refresh);
- Close – для выгрузки формы (метод Unload).

Кроме того, могут использоваться методы для перемещения по БД, связанные непосредственно с кнопочками объекта Data:

- MoveFirst (кнопочка |◄) – переход на первую запись,
- MoveLast – (кнопочка ►|) – переход на последнюю запись,
- MoveNext – (кнопочка ►) – переход на следующую запись,
- MovePrevious – (кнопочка ◄) – переход на предыдущую запись.

Синтаксис вышеуказанных методов такой: **`Объект БД.RecordSet.метод`**

Например, **`Data1.RecordSet.AddNew`** или **`Data1.RecordSet.MoveFirst`**

Записи в БД не обязательно последовательно просматривать одну за другой – можно ввести критерий поиска и выводить на экран только записи, удовлетворяющие условию. Для этого предназначены методы FindFirst (найти первый), FindLast (найти последний), FindNext (найти следующий), FindPrevious (найти предыдущий). Работают они примерно так же, как и методы Move, описанные выше. Разница лишь в том, что методы Find действуют уже в отобранном множестве и требуют параметр – строку с критерием поиска в виде логического выражения: **`Data1.RecordSet.FindFirst Kritery$`**, где **`Kritery$`** – символьная строка вида "**`(Условие 1)[AND ( Условие 2) OR (Условие 3) ...]`**", Условие – это есть **`[Имя поля] Операция сравнения Значение поля.`**

Если на форму, изображенную на рисунке 1, поместить еще одну командную кнопку для поиска по критерию и в событийной процедуре для этой кнопки написать соответствующий критерий поиска, то будет найдена, например, планета Нептун.

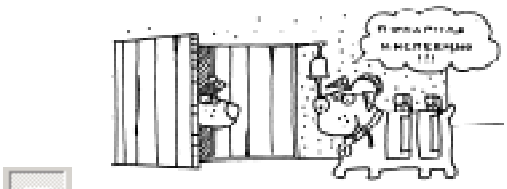
```
Private Sub Command2_Click()
   kr$ = "([Name]=" + Chr(34) +_
   "Нептун" + Chr(34) + ")"
```

```
    Data1.Recordset.FindFirst kr$
End Sub
```

Для работы с БД откроем новый проект. Создадим форму с помощью Мастера форм ***Data Form Wizard***. Если он не присутствует в меню Add-Ins, выберем команду Add-In Manager, а в открывшемся списке щелкнем пару раз на элементе VB 6 Data Form Wizard (или установим флажок Loaded/Unloaded) и один раз на кнопке OK. Теперь в меню появился нужный мастер форм, запустим его. В окне Introduction предлагается воспользоваться ранее используемыми настройками, но, как первопроходцы, проскакиваем следующие два окна, выбирая по пути тип базы данных Access и нашу БД Astronomy.mdb. В экране Form ставим указатели на типе формы Master/Detail и объекте ADO Data Control. Объект ADO (ActiveX Data Object) обеспечивает новый, более гибкий метод доступа к данным, рекомендуемый фирмой Microsoft. В следующем окне в качестве основной таблицы выберем Planet и все ее поля (клавишей ▸▸), располагая их в нужной последовательности для отображения в будущей форме (как в таблице 1) и в порядке следования (сортировка по полю Number). Далее аналогично поступаем с подчиненной таблицей (Detail) спутников планет (Satellite), которую отсортируем по полю Date (году открытия). В окне Record Source Relation устанавливаем связи таблиц. Естественной является связь таблиц по имени планеты (благо мы предусмотрели наличие сходных полей при разработке структуры таблиц), поэтому отмечаем поле Name в главной таблице и поле Name_P в подчиненной. В окне Control Selection откажемся от дополнительных кнопок на форме (Clear All), так как у нас нет необходимости редактировать данные таблицы планет. Завершим кнопкой Finish рутинную работу мастера форм.

Следующее совершенствование формы потребует участие разработчика. Как видим, в список компонентов добавились новые элементы управления: Adodc и DataGrid.

ОБЪЕКТ УПРАВЛЕНИЯ И КОНТРОЛЯ ADODC



Adodc (ADO Data Control) функционально подобен ранее рассмотренному элементу управления Data. Так как элемент использует широкие возможности для подключения к базам данных, рассмотрим лишь некоторые из них. Раскроем (щелкнув правой клавишей мыши на объекте) окно свойств Adodc. На вкладке General указаны способы подключения к базе данных.

В нашем случае мастер форм создал строку подключения к нашей базе с помощью Jet 3.51 – ядра базы данных типа Access. Для пользования объектов ADO к проекту автоматически мастером форм подключена библиотека MS ActiveX Data Objects 2.0 Library. Для работы с базами данных, совместимых с Access2000, требуется библиотека ADO версии 2.5 и Jet 4.0. Подключение соответствующей библиотеки осуществляется командой меню Project → References.

Если бы мы выбрали для подключения базы данных строку ODBC DSN (Open DataBase Connectivity Data Source Name), то необходимо было бы подключиться к ранее созданному источнику данных с помощью диспетчера ODBC, находящемуся на панели управления. Данный интерфейс обычно связан с базами данных клиент/сервер. Источник данных ODBC – это выбранная конфигурация драйвера, то есть набор функций, обеспечивающих стандартные обращения к различным форматам баз данных.

Какие конкретно данные из БД будут использоваться, указывается на вкладке RecordSource. Тип команды adCmdTable указывает на выбор конкретной таблицы, adCmdText выполняет SQL-запрос к источнику данных.

ОБЪЕКТ УПРАВЛЕНИЯ И КОНТРОЛЯ DATAGRID

Элемент управления DataGrid представляет выбранные данные в табличном виде – естественном представлении реляционных баз данных.

Основные свойства

Наряду с известными свойствами, присущими и другим объектам управления и контроля, DataGrid обладает следующими:

AllowAddNew, AllowDelete, AllowUpdate – при значении true позволяется добавлять новые, удалять и редактировать записи объекта RecordSet;

AllowSizing – при значении true позволяет менять размеры строк и столбцов в таблице;

ColumnHeaders – при значении true позволяет выводить заголовки колонок.

Раскроем окно свойств элемента управления DataGrid. На вкладке General зададим заголовок (Caption) «Спутники планеты», разрешим редактирование, удаление (при выделении строки и нажатии клавиши <Delete>) и добавление информации в ячейки DataGrid. На вкладке Keyboard оставим стандартное управление клавишами. Далее, во вкладке Columns определим начальную колонку (Column0): название (Caption) «Спутник» и поле таблицы (DataField) Moon. Аналогично, столбец 1: название – «Год открытия», поле – Date. На вкладке Layout выставим выравнивание поля «спутники» по левому краю. Поле с названием планеты таблицы Satellite не отображаем в DataGrid, так как связь Master/Detail гарантирует, что соответствующее поле автоматически выставляется (даже при добавлении нового спутника), согласно связанному полю в основной таблице.

Дополним автоматически созданную мастером форму. Подключим таблицу символов. На форму frmPlanet добавим элемент управления Data, выберем для свойства DatabaseName базу данных Astronomy.mdb, в качестве RecordSource определим таблицу Astrology. Сделаем Data невидимым (Visible=False). Добавим на форму PictureBox и свяжем его с полем Symbol источника данных Data1. Поставим дополнительную метку «Символ планеты» рядом с PictureBox (рисунок 2).

Введем дополнительные строки кода в процедуру datPrimaryRS_MoveComplete.

```
strsql = "Select Symbol from
Astrology where Astro"= & Chr$(39) &_
datPrimaryRS.Recordset.Fields("Name").Value_
& Chr(39)
Data1.RecordSource = strsql
Data1.Refresh
```

Этим мы переопределяем RecordSource элемента управления Data оператором SQL, который буквально означает: выб-

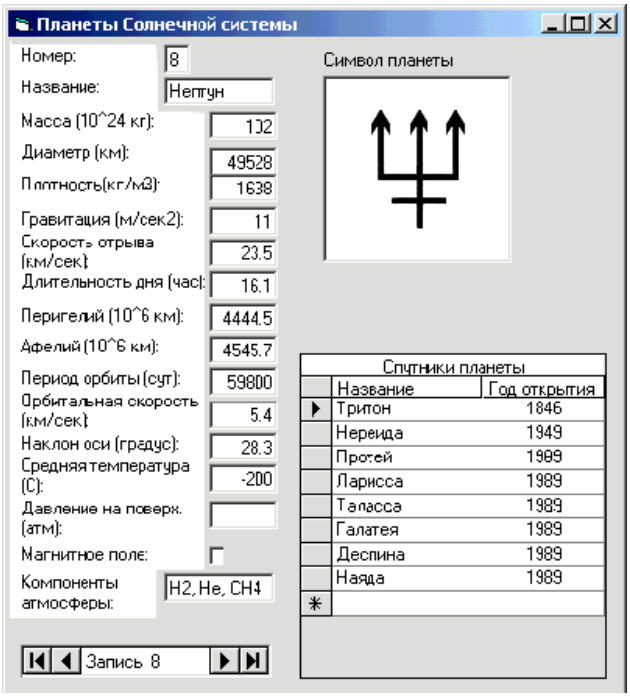


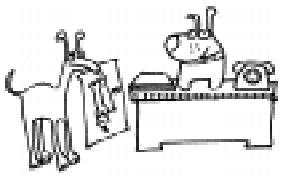
Рисунок 2

рать запись только с полем Symbol из таблицы Astrology, такую, что значение поля Astro для этой записи совпадает со значением имени планеты текущей записи набора данных элемента управления datPrimaryRS. После этого мы обновляем данные элемента управления Data

Внесем последний штрих – напишем русские названия меток согласно таблице 1. Приложение готово, можно протестировать его.

Настала пора познакомиться с операторами SQL – самым мощным средством работы с базами данных.

ВВЕДЕНИЕ В SQL



Операторы SQL (Structured Query Language – язык структурированных запросов) предназначены для формирования различного рода запросов к базам данных, которые позволяют пользователям манипулировать информацией в достаточно полной степени, как-то:

- провести выборку нужной информации из различных таблиц одной или нескольких БД, используя или образовывая логические связи во всей совокупности данных;
- модифицировать, создавать, удалять информативную или структурную информацию баз данных;
- представлять пользователю выбранные данные в необходимых форматах или виде;
- использовать функции языка программирования (VB), проводить анализ информации в БД и самом запросе.

Операторы SQL не могут использоваться в VB непосредственно, однако, строку запроса можно сохранить в БД в объекте QueryDef и в дальнейшем использовать параметром в других функциях.

SELECT – основной оператор SQL, с которым приходится работать пользователю по выборке информации из БД. Он имеет следующую форму:

SELECT [*predicate*] *fieldlist* FROM *tablelist* [WHERE *relations*] [group_options] [sort_options]

Большими буквами будем выделять ключевые слова SQL, но с равным успехом оператор может быть записан и в нижнем регистре.

Выражение *fieldlist* указывает список полей (разделенных запятой), которые должны быть включены в результирующий набор записей. Для однозначности отдельное поле может иметь впереди указатель родительской таблицы (с разделителем точка). Полю можно присвоить альтернативное имя с помощью ключевого слова AS. Это удобно, когда в качестве результирующего поля используется выражение или вложенный запрос. Вместо перечисления всех полей таблицы, можно применять символ шаблона «*».

Выражение *tablelist* содержит список таблиц из разных баз данных, откуда производится выборка информации. Также применяется удобный псевдоним для имени таблицы, который можно использовать в запросе.

Пример 1.

Сделать выборку всех характеристик каждой планеты из таблицы Planet.

```
select * from planet
```

Создадим программу для дальнейших экспериментов по освоению запросов SQL и визуализации результатов.

В новом проекте (или в предыдущем, используя форму Form1) подключим в References библиотеки для работы с БД по ADO и DAO технологии, соответственно, MS ADO 2.5 Library и MS DAO 3.51 Object Library. Функции DAO необходимы для работы с объектом QueryDef. Дополним список компонент элементами GridData и Adodc, установим их на форме (рисунок 3). Добавим на форму следующие элементы управления:

- TextBox (отмечено как «Запрос SQL»);
- ListBox (отмечен как «Список запросов»);

• кнопку Command1 с названием «Выполнить» (для выполнения запроса SQL);

• Command2 с названием «→» (для добавления строки запроса SQL из TextBox в ListBox);

• Command3 с названием «←» (для копирования выбранного элемента ListBox в TextBox);

• Command4 с заголовком «Сохранить в БД» (для сохранения выбранной строки запроса в БД в предназначенном для этого объекте QueryDef);

• Command5 с заголовком «Очистить» (для очистки TextBox);

• кнопку cmdClose («Выход») для выгрузки формы.

Ниже приведен код получившейся программы.

```
Dim Qry As QueryDef
Dim MyDb As Database
Private Sub Form_Load()
  Set MyDb =_
  OpenDatabase("astronomy.mdb")
End Sub
Private Sub Command1_Click()
  strsql = Text1.Text
  On Error GoTo er
  Adodc1.RecordSource = strsql
  DataGrid1.Visible = True
  Adodc1.Refresh
  Exit Sub
  er: DataGrid1.Visible = False
End Sub
Private Sub Command2_Click()
  List1.AddItem Replace(Text1.Text,
vbCrLf, " ")
End Sub
Private Sub Command3_Click()
  Text1.Text = List1.Text
End Sub
Private Sub Command4_Click()
  Nam = InputBox("Имя запроса?",_
  "Запрос в БД")
  Set Qry = MyDb.CreateQueryDef(Nam,
Text1.Text)
End Sub
Private Sub Command5_Click()
  Text1.Text = ""
End Sub
Private Sub cmdClose_Click()
  Unload Me
End Sub
```

Вместо приведенной учебной программы, можно использовать приложение VisData (Visual Data Manager) в среде VB. В окне SQL Statement программы VisData записывается запрос SQL, который можно выполнить (на дополнительный вопрос, является ли данный запрос SQLPass Trough, то есть запрос должен быть выполнен сервером, ответить отрицательно, так как рассматривается только локальная база данных), а также сохранить в БД. Для визуального построения запросов можно воспользоваться утилитой VisData – QueryBuilder.

Продолжим знакомство с SQL, проверяя свои действия по выбранной программе.

В списке полей оператора SELECT могут использоваться константы, символ конкатенации (&), функции VB (числовые или строковые, в зависимости от типа поля).

Пример 2.

Выразить расстояние планет от Солнца в астрономических единицах (1а.е. =149.6 млн. км – среднему расстоянию от Земли до Солнца)

```
select 'планета '&name, 'удалена от_
Солнца на ', (aphelion + _
perihelion)/(149.6*2) & ' a.e.' as_
distance from planet
```

Перед списком полей возможны ***предикаты*** ALL, DISTINCT, DISTINCTROW, TOP n [PERCENT]. Предикат ALL означает выборку всех записей, удовлетворяющих запросу (установка по умолчанию). Например, если выбирать планеты, у которых есть спутники, то название планеты будет встречаться столько раз, сколько у нее спутников. Для уникальности используется предикат DISTINCT.

Пример 3.

Получить однозначный список планет, для которых открыты спутники.

```
Select distinct name_p as [Имеют_
спутники] from satellite
```

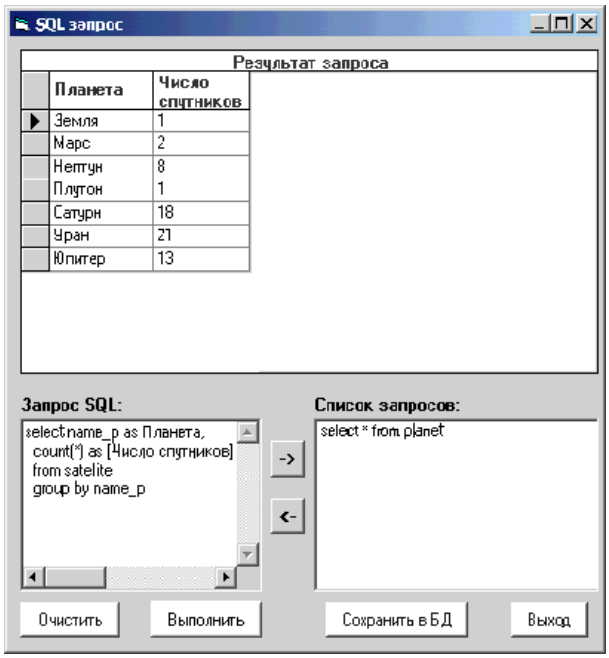


Рисунок 3

Предикат DISTINCTROW позволяет отбрасывать записи, которые совпадают полностью.

Предикат TOP n [PERCENT] включает в набор только первые n записей (процентов отобранных записей) с учетом сортировки.

Выражение *relations* после ключевого слова WHERE определяет критерий поиска для выборки строк из таблиц БД и предполагает наличие трех элементов:

- имени столбца в левой части;
- оператора сравнения;
- имени столбца, константы или перечня значений в правой части.

Операторы сравнения могут быть стандартными (=, >, <, >=, <=, <>) или операторами условия SQL.

Для операторов сравнения символьные данные должны заключаться в одинарные кавычки, а данные типа дата/время обрамляются символом #.

Пример 4.

Какая средняя температура на поверхности планеты Меркурий и есть ли там атмосфера?

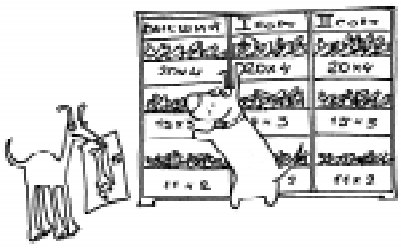
```
Select temperature, atmosphere_
from planet where_
name='Меркурий'
```

Пример 5.

Имеется ли хотя бы одна планета с орбитальной скоростью больше, чем 40.0 км/сек?

```
Select name, orbital_v from_
planet where orbital_v > 40.0
```

Операторы условия SQL:



- IN (список) – проверка на совпадение с одной из нескольких величин из списка;
- LIKE – проверка вхождения подстроки в строку, используются шаблоны:

% возможна любая последовательность символов, включая пустую,

_ (подчеркивание) любой символ, задает одну позицию в слове,

[список] одиночный символ из списка (например, [A-K, N]),

[!список] одиночный символ не из списка;

- IS NULL – признак пустого значения;
- BETWEEN ... AND ... – диапазон значений от и до, включая границы.

Возможно соединение выражений с помощью логических операторов AND, OR, NOT. Последовательность действий можно переопределять с помощью скобок.

Пример 6.

На каких планетах компонентом атмосферы является кислород и средняя температура на поверхности находится в диапазоне от –90° до 60° С? (Иными словами, где возможно наличие живой клетки?).

```
Select name, temperature, atmosphere_
from planet where (atmosphere like_
```

```
'%O%') and (temperature between -90_
and 60)
```

Сортировка записей, возвращаемых оператором SELECT, выполняется с помощью директивы ORDER BY. Сортировать можно по нескольким полям, разделенным запятыми.

Синтаксис команды сортировки:

ORDER BY *список_полей* [ASC | DESC], где

ASCending – обычная сортировка по умолчанию (в алфавитном порядке для строк, от меньшего к большему для чисел, от раннего к позднему для даты);

DESCending – обратная сортировка (по убыванию, от большего к меньшему), воздействует только на одно поле.

Пример 7.

Верно ли утверждение, что по мере уменьшения размера планеты, ее масса уменьшается?

```
Select  name, diameter, mass from_
planet order by diameter desc
```

Выбранные в операторе SELECT записи могут быть сгруппированы с помощью директивы GROUP BY по отдельным полям для получения сводных итоговых данных. Для этого используются групповые функции SQL:

- COUNT – определяет число выбранных записей;
- MIN – минимальное значение поля для записей;
- MAX – максимальное значение поля для записей;
- SUM – определяет сумму значений поля;
- AVG – получает среднее значение по полю.

Директива GROUP BY может включать необязательную директиву HAVING, которая позволяет выбрать не все группы, а лишь часть их по определенному критерию на групповые характеристики. В отличие от директивы HAVING, директива WHERE не может быть использована для задания ограничений на групповые функции.

Сортировка выполняется после отбора по директиве HAVING.

Пример 8.

Определить планеты, у которых число спутников больше трех, отсортировать эти планеты по возрастанию числа спутников.

```
select name_p as Планета, count(*)_
as [Число спутников] from satellite_
group by name_p having count(*)>3_
order by count(*)
```

Два запроса могут быть объединены в общую таблицу с помощью оператора UNION. Сортировка может быть применена только к объединенной таблице результатов. Так как в общем случае имена столбцов могут не совпадать при объединении разных таблиц (важно совпадение количества и типа столбцов), то, вместо имен в директиве ORDER BY, следует задавать их порядковые номера в предложении SELECT.

Пример 9.

На каких планетах в атмосфере присутствуют кислород и азот?

```
select name, atmosphere  from planet_
where atmosphere like '%N%'
union
select name, atmosphere  from planet_
where atmosphere like '%O%'
```

Операторы SQL могут быть вложенными. Подзапросы могут быть полезны, когда нужно выбрать строки из таблицы по условию, зависящему от состояния данных в самой таблице. Они позволяют конструировать из простых SQL-команд мощные запросы.

Пример 10.

Найти самую большую планету.

```
select name, diameter from planet_
where   diameter=   (select_
max(diameter) from planet)
```

Подзапрос можно использовать, например, в следующих формах:

- *Сравнение* [ANY | ALL] (*подзапрос*) – проверка истинности сравнения с каким-либо значением (ANY) или со всеми значениями (ALL) подзапроса;

• [NOT] IN (*подзапрос*) – проверка совпадения значения хотя бы в одной из записей подзапроса;

• [NOT] EXIST (*подзапрос*) – проверка наличия записей при выборке подзапроса.

Подзапросы могут задаваться также в предложении HAVING.

Кроме оператора SELECT, возможны следующие управляющие операторы SQL:

• DELETE FROM *имя_таблицы* [WHERE *выражение*]

Удаление (безвозвратное!) записи из таблицы по заданному критерию (если он не задан, то удаляются все записи);

• INSERT INTO *имя_таблицы* SELECT ...

Добавление группы записей в таблицу (обычно, добавление происходит данными из другой таблицы);

• UPDATE *имя_таблицы* SET *поле* = *новое_значение* [WHERE *выражение*].

Модификация значений полей в таблице (если опущена директива WHERE, то обновляются все записи). Данный оператор может быть применен, например, для какой-либо характеристики планеты при изменении единицы измерения.

Упражнения для самостоятельной работы

1. Составить SQL-запрос.

1) Определить площадь поверхности планеты Марс.

2) Какие планеты в диаметре меньше, чем Земля?

3) Какая планета солнечной системы самая малая и самая большая?

4) Для каких планет не известно, есть ли у них магнитное поле?

5) Сколько спутников у каждой планеты?

6) Какие планеты имеют больше естественных спутников, чем Земля?

7) Найти планеты, имеющие одинаковое количество спутников.

8) Найти планеты в таблице Planet, для которых нет данных в таблице Satellite.

9) Какие планеты входят в тройку самых больших планет (название, размер, отсортировать по размеру).

10) В каком году открыто наибольшее количество спутников? (С чем это связано?)

11) Определите для планет, какие спутники найдены последними по времени (упорядочить по дате).

2. Что можно сказать по поводу следующих утверждений? Дайте обоснованый ответ с помощью SQL-запроса. Есть ли исключения?

12) Чем дальше планета от Солнца, тем она холоднее.

13) Чем дальше планета от Солнца, тем ее период вращения вокруг Солнца больше.

14) Чем больше диаметр планеты, тем больше у нее спутников.

15) Чем больше масса планеты, тем больше у нее спутников.

3. Провести исследование при работе с БД.

16) Что можно сказать относительно орбитальной скорости, в зависимости от удаленности планеты от Солнца?

17) Провести проверку уникальности названий спутников.

18) Какие из спутников (и каких планет) названы в честь героев трагедии Шекспира «Отелло»?

Литература:

1. *Б. Хантер.* Мои ученики работают на компьютерах (книга для учителя). М.: «Просвещение», 1989.

Паньгина Нина Николаевна, преподаватель ОИ и ВТ школы лицея № 8, г. Сосновый Бор Ленинградской области.