Бурков А.В.

Проектирование информационных систем по технологии клиент – сервер в «Microsoft SQL Server 2008» и «Microsoft Visual Studio 2008»

Содержание

Введение

В настоящее время очень многим необходимы знания о базах данных и информационных системах. Уже давно мы, стоя в кассе по продаже билетов на поезда и самолеты не удивляемся действиям кассиров, недавно казавшимися необычными, и терпеливо ожидаем «приговора», который формируется в большом «мозге компьютера» - найдет он нам билет или не найдет. При покупке продуктов в магазине кассир считывает стоимость товаров при помощи сканера, а итоговую сумму и сдачу вычисляет «1С» - остается только проверить, правильно ли кассир считала эту сдачу. Менеджер по продажам узнает о продажах за день, неделю, месяц не по телефону а, просто открыв приложение, которое уже давно было «готово» извлечь эти данные с сервера и выдать в самом понятном менеджеру виде. Бухгалтер, выписывая вам «командировочные», заносит данные о вашей поездке в ленточную форму, которая тут же выписывает все необходимые бумаги для получения денег в кассе. У вас, наверняка, найдутся и свои примеры подобного «общения» с базами данных, которые чаще помогают, чем мешают в работе.

Если вы менеджер по продажам или закупкам, вам обязательно нужно знать, что могут программисты, и какие возможности таят в себе системы управления базами данных, как добиться, чтобы первые предоставили вам как можно больше из того, на что «способны» вторые. Кроме того, базы данных это средство для эффективной работы менеджеров. Менеджер всегда может «уговорить» программиста (разработчика баз данных) создать для него «канал», по которому можно при помощи запросов «вынимать» из базы данных любую информацию.

Из всех систем Windows-программирования «Microsoft Visual Studio 2008» - наиболее удобное, простое и эффективное средство для разработки интерфейса с базами данных. Еще большие возможности открываются при использовании «Microsoft SQL Server 2008» для доступа к информации, хранимой в базах данных.

Главной целью данной работы является познакомить читателя с методами разработки в таких системах как «Microsoft SQL Server 2008» и «Microsoft Visual Studio 2008». Данная работа предназначена для читателя уже знакомого с программированием и разработкой баз данных информационных систем.

Работа состоит из введения, двух частей и списка литературы. Работа имеет следующую структуру:

Введение. Содержит общую информацию по работе.

Часть 1. Содержит теоретические и практические занятия по разработке информационных систем в «Microsoft SQL Server 2008» и «Microsoft Visual Studio 2008»:

- ***Занятие №1.*** Описывает виды информационных систем и их основные понятия. Также здесь представлена история «Microsoft SQL Server 2008»;
- ***Занятие №2.*** Здесь представлены основные компоненты Microsoft SQL Server 2008. Описано создание файла данных и управление им при помощи команд языка T-SQL;
- ***Занятие №3.*** Описывает типы данных и свойства полей, а также создание и заполнение таблиц;
- ***Занятие №4.*** Содержит информацию по созданию запросов и фильтров. Описывает вычисление при помощи оператора SELECT и встроенных функций;
- ***Занятие № 5.*** Описывает создание динамических запросов при помощи хранимых процедур;

- ***Занятие №6.*** Содержит информацию по созданию пользовательских функций;
- ***Занятие №7.*** Рассматривает обеспечение целостности данных при помощи диаграмм и триггеров;
- ***Занятие №8.*** Содержит общую характеристику языка «Visual Basic 2008», историю его создания и системные требования. Также описывает объекты связи и мастер подключений;
- ***Занятие №9.*** Описывает интерфейс информационных систем и процесс создание интерфейса пользователя;
- ***Занятие №10.*** Рассматривает стандартные объекты для отображения данных и программное управление информационной системой;
- ***Занятие №11.*** Описывает объект для отображения табличной информации DataGridView и настройку его свойств и столбцов;
- ***Занятие №12.*** Содержит описание отчётов и объектов для работы с ними.

Часть 2. Содержит набор лабораторных работ для самостоятельного выполнения:

- ***Лабораторная работа №1.*** Содержит информацию об установке «Microsoft SQL Server 2008» и его настройке;
- ***Лабораторная работа №2.*** Описывает создание в «Microsoft SQL Server 2008» пустого файла данных и журнала транзакций;
- ***Лабораторная работа №3.*** Содержит информацию о создании, настройке и заполнении таблиц в новой базе данных;
- ***Лабораторная работа №4.*** Содержит информацию по созданию в «Microsoft SQL Server 2008» запросов и фильтров;
- ***Лабораторная работа №5.*** Описывает хранимые процедуры;
- ***Лабораторная работа №6.*** Содержит информацию о создании в «Microsoft SQL Server 2008» пользовательских функций;
- ***Лабораторная работа №7.*** Описывает создание диаграмм и триггеров, применяемых для обеспечения целостности данных;
- ***Лабораторная работа №8.*** Содержит начальные сведенья об интерфейсе среды разработки «Microsoft Visual Studio 2008», информацию по созданию пустого проекта и его подключению к базе данных, расположенную в «Microsoft SQL Server 2008»;
- ***Лабораторная работа №9.*** Описывает создание в «Microsoft Visual Studio 2008» главной кнопочной формы, а также простых ленточных форм для работы с данными;
- ***Лабораторная работа №10.*** Содержит информацию по созданию усложнённых ленточных форм для работы с данными, а также реализацию вычислимых полей;
- ***Лабораторная работа №11.*** Описывает создание табличных форм для отображения данных, а также реализацию сортировки, поиска и фильтрации информации из базы данных;
- ***Лабораторная работа №12.*** Показывает создание ленточных отчётов.

Часть 3. Содержит формы контроля знаний, полученных в ходе изучения данного курса:

- ***Задания для самостоятельного выполнения.*** Задания, которые могут быть использованы в качестве курсовых или самостоятельных работ;
- ***Билеты к экзамену.*** Билеты для проведения экзамена по данному курсу, содержат как теоретические, так и практические задания;

- ***Контрольный тест.*** Тест предназначенный для проверки остаточных знаний, или проведения зачёта по данному курсу.

Литература. Содержит список рекомендуемых книг для более углублённого изучения данного курса.

В работе используются следующие сокращения и обозначения:

- Аббревиатура ЛКМ обозначает левую кнопку мыши;
- Аббревиатура ПКМ обозначает правую кнопку мыши;
- Аббревиатура БД обозначает базу данных;
- Шрифтом «`Courier New`» выделяется код, набираемый в процедурах обработчиков событий;
- В двойные кавычки заключаются имена собственные, а также надписи на объектах;
- В замечания вынесена дополнительная не имеющая важного значения информация.

Часть 1. Лекционный курс

<u>Занятие 1.</u> Виды информационных систем. Основные понятия информационных систем. История Microsoft SQL Server 2008

Цели:
1. Изучить основные виды информационных систем
2. Определить преимущества и недостатки технологии Файл-Сервер и технологии Клиент-Сервер
3. Основные понятия информационных систем

Виды информационных систем

Информационные системы – это комплекс средств, предназначенных для хранения, упорядочивания и анализа больших объёмов информации.

Информационные системы бывают электронными и не электронными. К неэлектронным информационным системам относятся:

- Каталог в библиотеке;
- Регистратура в больнице;
- Библиотека.

К электронным информационным системам относятся:

- База данных отдела кадров предприятия;
- Записная книжка в мобильном телефоне;
- Сеть Интернет.

Существует три вида информационных систем:

1) База данных – система для хранения больших объёмов структурированной информации (информации, которая вводиться по шаблону) определённого типа. К базам данных относятся следующие информационные системы:

- каталог библиотеки;
- регистратура больницы;
- записная книжка мобильного телефона;
- база данных отдела кадров).

2) База знаний – система для хранения большого объема неструктурированной информации различных типов. К базам данных относятся следующие информационные системы:

- библиотека;
- сеть Интернет.

3) Информационно-аналитическая система – система, предназначенная как для хранения, так и для анализа хранимой информации

- Excel;
- STATISTICA;
- SPSS;
- 1С бухгалтерия;
- 1С предприятие.

Все электронные информационные системы делятся на два класса по способу хранения информации:

1. Не сетевые информационные системы, работающие по технологии файл-сервер. Данные системы работают на отдельно стоящем компьютере, без использования компьютерной сети (Excel, STATISTICA, SPSS);
2. Сетевые информационные системы, работающие по технологии клиент-сервер. Данные системы работают на компьютере, подключённом к компьютерной сети (Интернет).

Основное отличие технологии клиент-сервер от технологии файл-сервер заключается в способе хранения информации, суть технологии файл-сервер заключается в следующем – интерфейс информационной системы и данные, с которыми она работает храниться на одном компьютере (локально).

Замечание:

1) Клиентами сети являются компьютеры пользователей, подключенные к сети. Клиенты получают доступ к серверу через сеть. Иногда клиенты сети называют клиентскими компьютерами.
2) Сервер сети – компьютер, который управляет сетью. Все ресурсы сервера доступны клиентам сети, то есть любое изменения данных на сервере сразу видно всем клиентам сети.

В информационных системах, построенных по технологии клиент-сервер, информация хранится на сервере, а интерфейс информационной системы хранится на клиентских компьютерах, через него пользователи информационной системы получают доступ к данным.

Преимущества и недостатки технологии Файл-Сервер:

+ простота разработки;
+ независимость от компьютера сети;
+ высокая защита от несанкционированного доступа;
- не оперативное обновление данных на нескольких компьютерах;
- высокая стоимость компьютеров для работы в такой системе;
- сложность изменения структуры данных.

Преимущества и недостатки технологии Клиент-Сервер:

+ простая синхронизация данных;
+ низкая стоимость аппаратного обеспечения (мощным должен быть только сервер);
+ оперативное изменение структуры данных;
- низкая защита от несанкционированного доступа;
- зависимость от компьютерной сети;
- высокая стоимость.

Основные понятия информационных систем

Любая информационная система или база данных (с точки зрения их создания) в языках программирования состоят из трёх компонентов:

1. Файл данных – файл, находящийся на локальном компьютере или на сервере, который содержит внутри себя структуру данных. К структуре данных относятся таблицы, запросы и фильтры, а также хранимые процедуры, пользовательские функции диаграммы и триггеры;
2. Объект связи – объект языка программирования, осуществляющий связь между файлом данных и интерфейсом информационной системы;
3. Интерфейс информационной системы – комплекс средств, осуществляющий взаимодействие системы с конечными пользователями. Он может находиться как на клиентском компьютере, так и на сервере.

Разработка ИС по технологии клиент-сервер состоит из нескольких этапов:

1. На сервер в компьютерной сети устанавливаются серверная СУБД (Например, Microsoft SQL Server, My SQL, Oracle), устанавливается серверная часть СУБД. Если реализуется web-интерфейс, то на сервер ставиться программа web-сервер (Например, Apache);
2. Если реализуется клиентские приложения, то на все клиентские части сети ставиться клиентская часть (данный шаг не обязателен и выполняется только в том случае, если пользователи информационной системы имеют возможность управлять сервером);
3. Настраивается серверная часть СУБД, клиентские части СУБД и web-сервер;
4. Определяется структура данных (связи между таблицами и типы данных полей), также определяются первичные и вторичные таблицы в запросах;
5. На сервере создаются таблицы и запросы, выполняющиеся на стороне сервера. Перед созданием запросов, таблицы заполняются начальными данными. Также создаются хранимые процедуры, пользовательские функции, диаграммы и триггеры;
6. В случае использования клиентского приложения, при помощи языка программирования создаются объекты связи, они подключаются к таблицам, запросам и хранимым процедурам. Также на них создаются запросы и хранимые процедуры, выполняемые на стороне сервера;
7. Создаются формы;
8. Создаются отчёты;
9. Система заполняется реальными данными.

Замечание: При создании и заполнении таблиц информационной системы необходимо следовать 3 правилам:

1) В таблицах не должно быть повторяющихся групп записей. Это достигается введением индексных полей, то есть сортировкой записей;
2) В таблице не должно быть полей с одинаковыми именами. Это достигается разбиением одной таблицы на несколько, с последующим связыванием их запросом;
3) Не должно быть правил при заполнении таблиц, это достигается хаотичностью заполнения таблиц базы данных.

Информационная система, которая удовлетворяет этим условиям, называется нормализованной информационной системой или базой данных.

История Microsoft SQL Server 2008, его версии и системные требования

Родоначальником серии SQL Server и его основой является язык запросов SQL. Данный язык был создан компанией IBM в начале 1970г. прошлого века. Изначально он назывался SEQVEL (Structured English Query Language) В основу языка SQL, используемого в SQL Server, легла разновидность языка T-SQL (Transact – SQL).

В начале 80г. фирма IBM и в частности в то время ее подразделениями Microsoft и Sybase создается первая версия сетевой СУБД, которая называлась SQL Server версия 1.0, для операционной системы IBM OS/2. После этого под эту операционную систему было выпущено еще 3 версии SQL Server. В середине 80-х г. компания Microsoft и Sybase отделяются от фирмы IBM, и Microsoft начинает работу над своей операционной системой Windows, и вместе с компанией Sybase начинает развитие SQL Server.

В середине 90-х г. (в частности в 1995г) Microsoft создала операционную систему Windows NT и вместе с компанией Sybase выпускает первую версию SQL Server для Windows версии 4.1.

После этого компания Sybase разрывает свои отношения с Microsoft и Microsoft создает Microsoft SQL Server 6.0. Данная версия была предназначена для работы в

операционной системе Windows NT 95и 98. В 1999г. выходит версия Microsoft SQL Server 7.0, которая стала одной из самых популярных серверных СУБД в мире. В 2000г. выходит 8-я версия Micrsoft SQL Server 2000. В 2005 году выходит новая версия сервера, основанная на новой технологии NET, а в 2008 году выходит её улучшенная версия Microsoft SQL Server 2008.

Занятие 2. *Основные компоненты Microsoft SQL Server 2008. Создание файла данных. Управление базами данных при помощи команд языка T-SQL*

Цели:
1. Изучить систему основных компонентов Microsoft SQL Server 2008
2. Понять процесс создания файла данных
3. Освоить управление базами данных при помощи команд языка T-SQL

Основные компоненты Microsoft SQL Server 2008

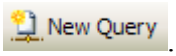
Все компоненты Microsoft SQL Server 2008 запускаются из меню «Пуск \ Программы \ Microsoft SQL Server 2008. В Microsoft SQL Server 2008 входят следующие компоненты:

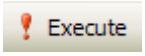
1. Deployment Wizard – мастер по выводу информации хранимой на сервере;
2. SQL Server Installation Center – сентр установки SQL Server 2008;
3. Reporting Services Configuration Manager – менеджер службы настройки отчётов;
4. SQL Server Configuration Manager – менеджер настройки сервера;
5. SQL Server Error and Usage Reporting – служба протоколирования работы сервера и служба отчётов об ошибках;
6. Microsoft Samples Overview – ссылка на сайт корпорации Microsoft, где можно просмотреть примеры работы с сервером;
7. SQL Server Books Online - полная справочная система по Microsoft SQL Server 2008. Она содержит справки, как по программированию, так и по администрированию сервера;
8. SQL Server Tutorials – учебники по работе с сервером;
9. Data Profile Viewer – просмотр профилей по работе с данными;
10. Execute Package Utility – инструменты по сжатию данных;
11. Database Engine Tuning Advisor – мастер настройки ядра базы данных;
12. SQL Server Profiler – настройка профилей по работе с данными;
13. Import and Export Data – импорт и экспорт данных;
14. SQL Server Business Intelligence Development Studio – интегрированная среда разработки Business Intelligence Development Studio;
15. SQL Server Management Studio – графическая оболочка для управления сервером и разработки баз данных.

Создание файла данных

Новую БД можно создать, используя стандартные команды языка T-SQL. Для создания новой БД необходимо сделать активную БД «Master». Это можно сделать либо выбором ее из выпадающего списка БД на панели инструментов, либо набором команды USE Master на вкладке нового запроса.

Замечание: Все команды языка T-SQL набираются на вкладке нового запроса (SQLQuery). Для того чтобы создать новый запрос на панели инструментов необходимо

нажать кнопку 

. Для выполнения команд языка T-SQL на панели инструментов необходимо нажать кнопку 

или на вкладке нового запроса набрать команду GO.

Замечание: В Microsoft SQL Server БД состоит из двух частей:

- Файл данных – файл, имеющий расширение mdf и где находятся все таблицы и запросы;
- Файл журнала транзакций - файл, имеющий расширение ldf, содержит журнал, где фиксируются все действия с БД. Данный файл предназначен для восстановления БД в случае её выхода из строя.

Для создания нового файла данных используется команда CREATE DATABASE, которая имеет следующий синтаксис:

```
CREATE DATABASE <Имя БД>
(Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsixe=<Макс.размер>,]
[FileGrowth=<Шаг>])
[LOG ON
(Name=<Логическое имя>,
FileName=<Имя файла>
[Size=<Нач.размер>,]
[Maxsixe=<Макс.размер >,]
[FileGrowth=<Шаг>])
```

Здесь Имя БД – имя создаваемой БД
Логическое имя – определяет логическое имя файла данных БД, по которому происходит обращение к файлу данных.
Имя файла – определяет полный путь к файлу данных.
Нач.размер – начальный размер файла данных в Мб.
Макс.размер – максимальный размер файла данных в Мб.
Шаг – шаг увеличения файла данных, либо в Мб либо в %.

Параметры в разделе LOG ON аналогичны параметрам в разделе CREATE DATABASE. Однако они определяют параметры журнала транзакций.

Пример: Создать БД «Students», расположенную в файле «D:\Students.mdf» и имеющую начальный размер файла данных 1мб., максимальный размер файла данных 100мб. и шаг увеличения файла данных равный 1мб. Файл журнала транзакций данной БД имеет имя «StudentsLog» и расположен в файле «D:\Students.ldf». Данный файл имеет начальный размер равный 1мб., максимальный размер равный 100мб. и шаг увеличения равный 1мб.

```
CREATE DATABASE Students
(Name = Students,
FileName = 'D:\Students.mdf',
Size = 1Mb,
Maxsixe = 100Mb,
FileGrowth= 1Mb)
LOG ON
```

```
(Name = StudentsLog,
FileName = 'D:\Students.ldf',
Size = 1Mb,
Maxsixe = 20Mb,
FileGrowth = 1Mb)
```

Управление базами данных при помощи команд языка T-SQL

В языке запросов T-SQL с БД возможны следующие действия:

1. Отображение сведений о БД: `EXEC sp_helpdb <Имя БД>`;
2. Изменение параметров БД: `EXEC sp_dboption <Имя БД>, <Параметр>, <Значение>`;
3. Добавления новых файлов, удаление файлов и переименования файлов, входящих в БД:

   ```
   ALTER DATABASE <Имя БД>
   ADD FILE (<Параметры>)|
   REMOVE FILE <Логическое имя файла>|
   MODIFY FILE (<Параметры>)
   ```

   где, раздел ADD FILE - добавляет файл, REMOVE FILE – удаляет, а раздел MODIFY FILE – изменяет параметры файла;
4. Сжатие всей БД: `DBCC SHRINKDATABASE <Имя БД>`;
5. Сжатие конкретного файла БД: `DBCC SHRINKFILE <Логическое имя файла>`;
6. Переименование БД: `EXEC CP_RENAMEDB <Имя БД>,<Новое имя БД>`;
7. Удаление БД: `DROP DATABASE <Имя БД>`.

Замечание: Вышеперечисленные команды используют следующие параметры:

- <Имя БД> - имя БД с которой производится действие;
- <Параметр> - изменяемый параметр;
- <Значение> - новое значение изменяемого параметра;
- <Параметры> - параметры файла БД, аналогичные параметрам, используемым в команде CREATE DATABASE;
- <Логическое имя файла> - логическое имя файла, входящего в БД;
- <Новое имя БД> - новое имя БД.

На этом мы заканчиваем рассмотрение файлов данных. Дополнительную информацию можно найти в лабораторной работе №2.

Занятие 3. *Таблицы. Типы данных и свойства полей. Создание и заполнение таблиц*

Цели:
1. Изучить таблицы и типы данных полей
2. Освоить создание таблиц и основные операции с таблицами

Таблицы. Типы данных полей

Вся информация в базе данных хранится в таблицах. Таблицы это обычные таблицы для хранения данных. Таблицы состоят из записей.

Запись это строка в таблице. Вся информация обрабатывается по записям.

Каждая запись состоит из полей. Поле это столбец таблицы. Каждое поле имеет три характеристики:

1. Имя поля – используется для обращения к полю;
2. Значение поля – определяет информацию, хранимую в поле;
3. Тип данных поля – определяет какой вид информации можно хранить в поле.

В SQL сервер используется следующие типы данных:

- **Битовые типы данных** которые содержат последовательности нулей и единиц: Binary(n) и Varbinary(n), где n длина. Содержимое полей типа Binary всегда равно n, разница заполняется пробелами. Varbinary размер поля равен n или большему;
- **Целочисленные типы данных** – типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): Tinyint (0-255), Smallint (±32000), Int (±2000000000), Bigint ($\pm 2^{63}$);
- **Типы данных для хранения дробных чисел:** Real семь знаков после запятой, Float(m) может хранить числа из m знаков, максимальное m=38, Decimal(m n) дробные числа с m знаков до запятой и n после;
- **Специальные типы данных:** Bit – логический тип данных.является заменой логическому типу Boolean в Visual Basic, Text - тип для хранения больших объемов текста, одно поле может хранить до 2 Гб текста, Image – тип данных для хранения до 2Гб рисунков, RowGUID – уникальный идентификатор строки таблицы, SQL_Variant - аналогичен типу Variant в Visual Basic;
- **Типы данных даты и времени:** Datetime (от 1.01.1953 до 3.12. 1999). SmallDatetime (от 1.01.19 до 6.07 2079);
- **Денежные типы данных для хранения финансовой информации:** Money ($\pm 10^{15}$ и 4 знака после нуля), Smallmoney (± 20000,0000);
- **Автоматически обновляемые типы данных** - аналоги счетчиков, но в данной роли они не используются: RowVersion уникальный идентификатор строки. TimeStamp – закодированное дата и время создания строки.

Создание таблиц

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого можно в новом запросе можно набрать команду: USE <Имя БД>, либо на панели инструментов необходимо выбрать в выпадающем списке рабочую БД. После выбора БД можно создавать таблицы.

Таблицы создаются командой

```
CREATE TABLE <Имя таблицы>(<Имя поля1> <Тип1> [IDENTITY
NULL|NOTNULL],<Имя поля2> <Тип2>, … )
```

Здесь <Имя таблицы> – имя создаваемой таблицы;
<Имя поля> – имена полей таблицы;
<Тип> – типы полей;
<IDENTITY NULL|NOT NULL> – поле счётчик.

Замечание: Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Пример: Создать таблицу «Студенты», содержащую поля: Код студента (первичное поле связи, счётчик), ФИО, Адрес, Код специальности (вторичное поле связи):

```
CREATE TABLE Студенты
([Код студента] Bigint Identity,
ФИО Varchar(20),
Адрес Varchar(100),
[Код специальности] Bigint)
```

Замечание: Если необходимо создать вычислимое поле, то в команде Create Table у вычислимого поля вместо типа данных нужно указать выражение.

Пример: рассчитать средний балл студента по трем его оценкам.

```
CREATE TABLE Оценки
(ФИО Varchar(20),
Оценка1 int,
Оценка2 int,
Оценка3 int,
[средний балл] = (Оценка1+ Оценка2+ Оценка3)/3)
```

Замечание: Получение информации о таблице осуществляется применением команды: EXEC SP_HELP <Имя таблицы>. Удаление таблицы осуществляется командой: DROP TABLE <Имя таблицы>.

Заполнение таблиц

В SQL Server 2008 заполнение таблиц производится при помощи следующей команды:

```
INSERT <Имя таблицы> [(<Список полей>)]
VALUES (<Значения полей>)
```

где <Имя таблицы> – таблица, куда вводим данные, (<Список полей>) – список полей, куда вводим данные, если не указываем, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую, (<Значения полей>) – значение полей через запятую.

В качестве значений можно указать к константу Default, то есть будет поставлено значение по умолчанию, либо можно подставить оператор Select. Здесь он используется как инструмент вычисления формул.

Пример: Добавление записи имеющей следующие значения полей ФИО = Иванов, Адрес = Москва, Код специальности = 5 в таблицу «Студенты».

```
INSERT Студенты (ФИО, Адрес, [Код специальности])
VALUES ('Иванов А.А.', 'Москва', 5)
```

Удаление отдельных столбцов и отдельных строк из таблицы

Из таблицы можно удалить все столбцы, либо отдельные записи. Это осуществляется командой

```
DELETE <Имя таблицы>
[WHERE <Условие>]
```

где <Условие> - условие, которым удовлетворяют удаляемые записи, если условие не указаны, то удаляются все столбцы таблицы. Если условие указано то удаляются записи поля которых соответствуют условию.

Пример: Удалить записи из таблицы «Студенты», у которых поле Адрес = Москва.

```
DELETE Студенты
WHERE Адрес = 'Москва'
```

Изменение данных в таблице

Для этого используется следующая команда:

```
UPDATE <Имя таблицы>
SET
<Имя поля1> = <Выражение1>,
[<Имя поля2> = <Выражение2>,]
…
[WHERE <Условие>]
```

Здесь <Имя поля1>, <Имя поля2> - имена изменяемых полей, <Выражение1>, <Выражение 2> - либо конкретные значения, либо NULL, либо операторы SELECT. Здесь SELECT применяется как функция.
<Условие> – условие, которым должны соответствовать записи, поля которых изменяем.

Пример: В таблице «Студенты» у студента Иванова А.А. поменять адрес Москва на Йошкар-Ола, а код специальности вместо 5 поставить 3.

```
UPDATE Студенты
SET
Адрес = 'Йошкар-Ола',
[Код специальности] = 3
WHERE ФИО = 'Иванов А.А.'
```

Замечание: в качестве выражения можно использовать математические формулы.
Например: `SET [Средний балл]= (Оценка1+ Оценка2+ Оценка3)/3)` вычисляет поле «Средний балл» как среднее полей «Оценка1», «Оценка2» и «Оценка3».

При этом поля «Оценка1», «Оценка2» и «Оценка3» должны уже существовать и тип данных поля «Средний балл» должен быть с плавающей запятой (Например Real).

Замечание: Если необходимо из таблицы удалить все записи, но сохранить ее структуру, для этого используют команду TRUNCATE TABLE <Имя таблицы> при этом все данные будут удалены, но сама таблица останется.

На этом мы заканчиваем рассмотрение таблиц. Дополнительную информацию можно найти в лабораторной работе №3.

Занятие 4. *Создание запросов и фильтров. Вычисление при помощи оператора SELECT. Встроенные функции*

Цели:
1. Изучить создание запросов и фильтров
2. Понять процесс выполнения вычислений при помощи оператора SELECT. Встроенные функции

Создание запросов и фильтров

Запросы предназначены для связи одной или нескольких таблиц, также они могут осуществлять отбор отдельных полей из таблицы и производить фильтрацию данных согласно условию, наложенному на одно или несколько полей, такие запросы называют фильтрами.

Для реализации запросов используют специальный язык запросов SQL (Standard Query Language).

В ИС Запросы могут находиться как на стороне клиентского приложения, так и на стороне сервера. Если запрос храниться на стороне клиента, то он прописывается внутри объекта связи. В этом случае клиентское приложение не зависит от файла данных. Файл данных содержит только таблицы, поэтому, мы легко можем модифицировать клиентское приложение, не затрагивая файл данных, но в этом случае запрос передается серверу через сеть, что может вызвать проблемы с безопасностью.

Если запрос храниться или выполняется на сервере, то сам запрос выступает в качестве компонента БД, вся передача информации происходит внутри файл данных, т.е. внутри самого сервера, клиентскому приложению только передаются результаты выполнения запроса. В этом случае обеспечивается высокая защита данных, но в случае изменения запроса придется менять сам файл данных.

Все запросы делятся на:

1. статические;
2. динамические

Структура статических запросов неизменна в ходе работы с программой, а динамические запросы могут меняться в зависимости от ситуации.

Замечание. Обычно динамические запросы могут быть реализованы только при помощи запросов, выполняющиеся на стороне клиента. Если необходимо реализовать динамические запросы, которые выполняются на стороне сервера, то в этом случае необходимо использовать хранимые процедуры.

Хранимые процедуры – SQL запрос, хранимый на стороне сервера и этот запрос имеет параметры, которые подставляются внутрь SQL кода. При вызове хранимой процедуры необходимо передавать внутрь ее значения параметра.

В основном запрос или хранимая процедура либо реализует связь между таблицами, либо осуществляет фильтрацию данных, некоторые SQL запросы также могут производить вычисления.

В случае связей между таблицами одна таблица всегда выступает первичной, а другая вторичной, связь происходит при помощи полей связи. При связи сопоставляются записи с одинаковыми значениями полей связи. Первичная таблица всегда заполняется первой, а ее поле связи заполняется автоматически (тип данных - счетчик). Вторичная

таблица всегда заполняется после заполнения первичной таблицы, значения ее поля связи подставляется из значений поля связи первичной таблицы. Поля связи должны иметь одинаковый тип данных.

Существует четыре вида связи между таблицами:

1. Одна к одной – одному полю в первичной таблице соответствует одно поле во вторичной таблице;
2. Одна ко многим - одному полю в первичной таблице соответствует несколько полей во вторичной таблице;
3. Многие к одной – нескольким полям в первичной таблице соответствует одно поле во вторичной таблице;
4. Многие ко многим – одному полю в первичной таблице соответствует несколько полей во вторичной таблице и наоборот.

Запросы с первым видами связи называются простыми, а с остальными видами связи – сложными. Если в БД есть хотя бы две связанных таблицы, то БД - реляционной.

Чтобы создать запрос необходимо сделать активной БД для которой создается запрос, затем в рабочей области редактора запросов создать запрос с помощью команды SELECT, имеющей следующий синтаксис:

```
SELECT [ALL|DISTINCT]
[TOP|PERCENT n]
<Список полей>
[INTO <Имя новой таблицы>]
[FROM  <Имя таблицы >]
[WHERE <Условие>]
[GROUP BY <Поле>]
[ORDER BY <Поле > [ASC|DESC]]
[COMPUTE AVG|COUNT|MAX|MIN|SUM(<Выражение>)]
```

Здесь параметры ALL|DISTINCT показывают, какие записи обрабатываются: ALL обрабатывает все записи, DISTINCT только уникальные, удаляются повторения записей.

TOP n определяет какое количество записей обрабатывают, если указан PERCENT, то n указывает процент от общего числа записей. <Список полей> - здесь указываются отображаемые поля из таблиц через запятую.

Замечания:

1) Если имена отображаемых полей в разных таблицах не повторяются, то мы можем указывать только имена столбцов или полей без указания самих полей (ФИО, Должность). Если отображаются поля из разных таблиц с одинаковыми именами нужно указывать и имя таблицы <Имя поля>.<Имя таблицы>;
2) Здесь же можно присваивать псевдонимы полям, следующим образом <Имя поля> AS <Псевдоним>
3) Если необходимо вывести все поля из таблицы, то их можно заменить значком «*»

Раздел INTO. Если присутствует этот раздел, то на основе результатов запроса создается новая таблица. Параметр INTO это имя новой таблицы.

Раздел FROM. Здесь указываются таблицы и запросы, через запятую, которые участвуют в новом запросе.

Замечание: В разделе FROM так же можно задавать сложные связи, связь поля одной таблицы, с несколькими полями другой таблицы. В этом случае раздел FROM будет иметь следующий вид:

```
FROM <Таблица1> INNER JOIN <Таблица2> ON <Таблица1>.<поле1>
оператор <Таблица2>.<поле2> …
```

Здесь устанавливается взаимосвязь Таблицы 1 и Таблицы2 по Полю1 и Полю2 в зависимости от оператора сравнения. Таких разделов INNER JOIN может быть сколько угодно.

Раздел WHERE. Данный раздел используют для создания простых запросов, в этом случае в качестве условия указываем связываемые поля, либо этот раздел используют для создания фильтров, здесь указывают условия отбора. В условиях отбора мы можем использовать стандартные логические операторы NOT, OR, AND.

Замечание: В своем стандартном виде запросы могут реализовывать только статичные фильтры, но не динамические. Для реализации динамических фильтров используются хранимые процедуры.

Раздел GROUP BY – определяет поле для группировки записей в запросе.

Раздел ORDER BY – определяет поле для сортировки записей в запросе. Если указан параметр ASC, то будет производиться сортировка по возрастанию, если DESC – по убыванию. По умолчанию используется сортировка по возрастанию.

Раздел COMPUTE позволяет в конце результатов выполнения запроса вывести некоторые итоговые вычисления по запросу. Возможны следующие виды вычислений: AVG – средняя параметра; COUNT – количество значений параметра не равных NULL; MAX и MIN – максимальные и минимальные значения параметра; SUM – сумма всех значений параметра, где <Выражение> – сам параметр. В качестве параметра обычно выступают какие либо поля таблиц, участвующих в запросе.

Пример: Данный запрос связывает две таблицы Сотрудники и Должности по полям Код. При своем выполнении он отображает первые 20 процентов сотрудников из обеих таблиц. Из таблицы сотрудники отображаются все поля, а из таблицы Должности только поле должность. В конце результатов выводится количество отображенных сотрудников.

```
SELECT TOP 20 PERCENT *.Сотрудники, Должность.Должности
FROM Сотрудники, Должности
WHERE Код.Сотрудники = Код.Должности
COMPUTE COUNT (ФИО.Сотрудники)
```

Пример: Данный запрос из таблицы Операции выводит все записи, значение поля Месяц у которых равняется «Май». Данные в результате группируются по полю операция и сортируются по сумме операции. В конце результатов запроса отображается общая сумма отобранных операций за май. Результаты данного запроса сохраняются в таблице «Сделки за май».

```
SELECT ALL Операция, Сумма
INTO [Сделки за Май]
FROM Операции
WHERE Месяц = 'Май'
GROP BY Операция
ORDER BY Сумма
COMPUTE SUM (Сумма)
```

Замечание: В данном запросе при обозначении названия полей таблицы явно не указываются, так как использовалась только одна таблица.

Выполнение вычислений при помощи оператора SELECT. Встроенные функции

Кроме связывания таблиц и отбора данных оператор SELECT может использоваться для вычислений. В этом случае он имеет синтаксис:

```
SELECT <Выражение>
```

где <выражение> - какое-то математическое выражение или функция. Выражение имеет стандартный вид (как в Visual Basic), оно может включать в себя встроенные функции сервера.

Замечание: Мы можем использовать встроенные функции и выражения в вычисляемых полях при создании таблиц.

В SQL Server существуют следующие встроенные функции разбития на группы.

Математические функции

Замечание: В качестве параметров функции будем указывать соответствующий им тип данных.

ABC (numeric) - модуль числа;
ACOS/ASIN/ATAN (Float) – арккосинус, арксинус, арктангенс в радианах;
COS/SIN/TAN/COT (Float) – косинус, синус, тангенс, котангенс;
CEILING (Numeric) – наименьшее целое, большее или равное параметру в скобках;
DEGREES (Numeric) – преобразует радианы в градусы;
EXP(Float) – экспонента, e^x;
FLOOR(Numeric) – наименьшее целое меньшее или равное выражению numeric;
LOG(Float) – натуральный логарифм ln;
LOG10(Float) – десятичный логарифм $\log_{10}$;
PI () – число пи;
POWER (Numeric,y) – возводит выражение Numeric в степень y;
RADIANS (Numeric) – преобразует градусы в радианы;
RAND () – генерирует случайное число типа данных Float, расположенное между нулем и единицей;
ROUND (Numeric, Длина) – округляет выражение Numeric до заданной Длины (количество знаков после запятой);
SIGN (Numeric) – выводит знак числа +/- или ноль;
SQUARE (Float) – вычисляет квадрат числа Float;
SQRT (Float) – вычисляет квадратный корень числа Float.

Примеры использования математических функций:

`SELECT ABC(-10)` результат 10
`SELECT SQRT (16)` результат 4
`SELECT ROUND (125.85,0)` результат 125
`SELECT POWER (2,4)` результат 16

Строковые функции

Строковые функции позволяют производить операции с одной или несколькими строками.
‘Строка1’+ ‘Строка2’ присоединяет Строку1 к Строке2;
ASCII(Char) – возвращает ASCII код с самого левого символа выражения Char;
CHAR(Int) – выводит символ соответствующий ASCII коду в выражении Int;
CHARINDEX(Образец, Выражение) – выводит позицию Образца выражения, то есть где находится Образец в Выражении;
DIFFERENCE(Выражение1, Выражение2) – сравнивает два выражения, выводит числа от 0 до 4: 0 – выражения абсолютно различны; 4 – выражения абсолютно идентичны. Оба выражения типа данных Char;
LEFT(Char, Int) – выводит из строки Char Int символов слева;
RIGHT(Char, Int) – выводит из строки Char Int символов справа;
LTRIM(Char) – удаляет из строки Char пробелы слева;
RTRIM(Char) – удаляет из строки Char пробелы справа;
WCHAR(Int) – выводит выражение Int в формате Unicode;
REPLACE(Строка1, Строка2, Строка3) – меняет в Строке1 все элементы Строка2 на элементы Строка3;
REPLICATE(Char, Int) – повторяет строку Char Int раз;
REVERSE(Char) - производит инверсию строки Char, то есть располагает символы в обратном порядке;
SPACE(Int) – выводит Int пробелов;
STR(Float) – переводит число Float в строку;
STUFF(Выражение1, Начало, Длина, Выражение2) – удаляет из Выражения1 начиная с позиции символа Начало количество символов равное параметру Длина, вместо них подставляет Выражение2;
SUBSTRING(Выражение, Начало, Длина) – из Выражения выводится строка заданной Длины начиная с позиции Начало;
UNICODE(Char) – выводит код в формате Unicode первого символа в строке Char;
LOWERC(Char) – переводит строку Char в маленькие буквы;
UPPER(Char) – переводит строку Char в заглавные буквы.

Примеры применения строковых функций:

`SELECT ASCII('G')` результат 71.
`SELECT LOWER('ABC')` результат abc.
`SELECT Right('ABCDE')` результат CDE
`SELECT REVERCE('МИР')` результат РИМ.

Замечание. Во всех строковых функциях значения выражения типа Char заключаются в одинарные кавычки.

Функции дат

Замечание: в некоторых функциях дат используется так называемая часть дат, которая кодируется специальными символами:
dd – число дат (от 1 до 31);
dy – день года (число от 1 до 366);
hh – значение часа (0-23)
ms – значение секунд (от 0 до 999)
mi – значение минут (0-59)
qq – значение (1-4)

mm – значение месяцев (1-12)
ss – значение секунд (0-59)
wk – значение номеров недель в году
dw – значение дней недели, неделя начинается с воскресенья (1-7).
yy – значение лет (1753 -999)

Функции дат предназначены для работы с датами или времени. Существуют несколько следующие функции дат:
DATEADD(часть, число, date) – добавляет к дате date часть даты умноженное на число;
DATEDIFF(часть, date1, date2) – выводит количество частей даты между date1 и date2;
DATENAME(часть, date) – выводит символьное значение частей даты к заданной дате (название дней недели);
DATEPART(часть, date) – выводит числовое значение части даты из заданной даты (номер месяца);
DAY(date) – выводит количество дней в заданной дате;
MONTH (date) – выводит количество месяцев в заданной дате;
YEAR(date) – выводит количество лет в заданной дате;
GETDATE() – выводит текущую дату установленную на компьютере;

Замечание: Даты выводятся в Американском формате: месяц/день/год.

Примеры функции работ с датами:

`SELECT DATEADD(dd,5,11/20/07)` результат Nov/25/2007.
`SELECT DATEDIFF(dd,11/20/07, 11/25/07)` результат 5 дней.
`SELECT DATENAME(mm, 11/20/07)` результат November.
`SELECT DATEPART(mm, 11/20/07)` результат 11.

Замечание: В выражениях оператора SELECT можно использовать операции сравнения. В результате будет либо истина TRUE, либо ложь FALSE. Можно использовать следующие операторы: =, <, >, >=, <=, <>, !<(не меньше), !>(не больше), !=(не равно). Приоритет операции задается круглыми скобками.

Системные функции

Системные функции предназначены для получения информации о базе данных и ее содержимом. В SQL сервере существуют следующие системные функции:
COL_LENGTH(таблица, поле) – выводит ширину поля;
DATALENGTH(выражение) – выводит длину выражения;
GETAHSINULL(имя БД) – выводит допустимо или недопустимо использовать в БД значение NULL;
IDENTINCR(таблица) – выводит шаг увеличения поля счетчика в таблице;
IDENT_SEED(таблица) – выводит начальное значение счетчиков в таблице;
ISDATE(выражение) – выводит единицу, если выражение является датой и ноль, если не является;
ISNUMERIC(выражение) – выводит единицу, если выражение является числовым и ноль, если не числовым;
NULIFF(выражение1, выражение2) – выводит ноль если выражение1 равно выражению 2.

Агрегатные функции

Агрегатные функции – позволяют вычислять итоговые значения по полям таблицы.

AVGL(поле) – выводит среднее значение поля;
COUNT(*) – выводит количество записей в таблице;
COUNT(поле) – выводит количество всех значений поля;
MAX(поле) – выводит максимальное значение поля;
MIN(поле) – выводит минимальное значение поля;
STDEV(поле) – выводит средне квадратичное отклонение всех значений поля;
STDEVP(поле) – выводит среднеквадратичное отклонение различных значений поля;
SUM(поле) – суммирует все значений поля;
TOP n [Percent] – выводит n первых записей из таблица, либо n% записей из таблицы;
VAR(поле) – выводит дисперсию всех значений поля;
VARP(поле) – выводит дисперсию всех различных значений поля.

Примеры использования агрегатных функций:

`SELECT AVG(возраст) FROM Студенты` – выводит средний возраст студента из таблицы «Студенты».

`SELECT COUNT(ФИО) FROM Студенты` – выводит количество различных ФИО из таблицы «Студенты».

`SELECT Top 100 FROM Студенты` – выводит первые 100 студентов из таблицы «Студенты».

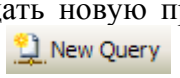
На этом мы заканчиваем рассмотрение запросов и фильтров. Дополнительную информацию можно найти в лабораторной работе №4.

Занятие 5. Создание динамических запросов при помощи хранимых процедур

Цель: изучить процесс создания динамических запросов при помощи хранимых процедур

Хранимая процедура – SQL запрос, который имеет параметры, то есть он выполняется как обычная процедура (мы задаем ее имя и передаем в хранимую процедуру значение параметров.) В зависимости от значения параметров хранимой процедуры мы получаем тот или иной результат запроса.

Замечание. В SQL сервере хранимые процедуры реализуют динамические запросы, выполняемые на стороне сервера.

Рассмотрим создание хранимых процедур при помощи команд языка SQL. Чтобы отобразить хранимые процедуры рабочей БД панели «Object Explorer» нужно выделить пункт «Stored Procedures». Чтобы создать новую процедуру при помощи команд языка SQL нужно щелкнуть ЛКМ по кнопке 

на панели инструментов. В рабочей области окна сервера появится вкладка SQLQuery1.sql, где нужно набрать код новой процедуры., который имеет следующий синтаксис:

```
CREATE PROCEDURE <Имя процедуры>
[@<Параметр1> <Тип1>[=<Значение1>],
@<Параметр2> <Тип2>[=<Значение2>], …]
[WITH ENCRYPTION]
AS <Команды SQL>
```

Здесь Имя процедуры – имя создаваемой хранимой процедуры.
Параметр1, Параметр2, … - параметры передаваемые в процедуру.
Значение1, Значение2, … - значения параметров по умолчанию.
Тип1, Тип2, … - типы данных параметров.
WITH ENCRYPTION – включает шифрование данных.
Команды SQL - SQL запрос, который выполняется при запуске процедур.

Замечание: SQL запрос включает в себя параметры, если параметры сравниваются с какими то полями или выражениями, то они должны иметь точно такой же тип данных как эти поля или выражения.

Замечание: После создания процедура помещается в раздел Stored Procedures текущей БД на панели «Object Explorer». Если дважды щелкнуть по процедуре ЛКМ то она откроется для редактирования на вкладке «SQLQuery».

Чтобы посмотреть информацию о хранимой процедуре необходимо выполнить команду

```
EXEC SP_HELPTEXT <Имя процедуры>
```

Хранимые процедуры могут быть запущены следующей командой

```
EXEC <Имя процедуры> [<Параметр1>, <Параметр2>, …]
```

здесь <Имя процедуры> - имя выполняемой процедуры.
Параметр1, Параметр2, … – значение параметров.

Пример: Создание хранимой процедуры, который выводит имя студентов, у которых средний балл больше заданной величины:

```
CREATE PROCEDURE СрБАЛЛ
@X Real
AS
SELECT *
FROM Студенты
WHERE
(Оценка1+ Оценка2+ Оценка3)/3>@X
```

Команда вызова вышенабранной процедуры выглядит следующим образом:

```
EXEC СрБАЛЛ 4
```

Команда выводит всех студентов, у которых средний балл больше 4.

На этом мы заканчиваем рассмотрение хранимых процедур. Дополнительную информацию можно найти в лабораторной работе №5.

Занятие 6. Пользовательские функции

Цели:
1. Изучить порядок создания пользовательских функций
2. Освоить применение пользовательских функций

Пользовательские функции очень похожи на хранимые процедуры. Так же в них можно передавать параметры и они выполняют некоторые действия, однако их главным отличием от хранимых процедур является то, что они выводят (возвращают) какой то результат. Более того, они вызываются только при помощи оператора SELECT, аналогично встроенным функциям. Все пользовательские функции делятся на 2 вида:

1. Скалярные функции – функции, которые возвращают число или текст, то есть одно или несколько значений;
2. Табличные функции – функции, которые выводят результат виде таблицы.

Для создания новой пользовательской функции используется команда CREATE FUNCTION имеющая следующий синтаксис:

```
CREATE FUNCTION <Имя функции>
([@<Параметр1> <Тип1>[=<Значение1>],
 @<Параметр2> <Тип2>[=<Значение2>], . . .])
RETURNS <Тип>/TABLE
AS
RETURN([<Команды SQL>])
```

Здесь Имя функции – имя создаваемой пользовательской функции.
Параметр1, Параметр2, . . – параметры передаваемые в функцию.
Значение1, Значение2, … - значения параметров по умолчанию.
Тип1, Тип2, . . . – типы данных параметров.

После служебного слова RETURNS в скалярных функциях ставится тип данных результата, который возвращает скалярная функция, либо ставится служебное слово TABLE в табличных функциях.

После служебного слова RETURN ставится SQL команда самой функции.

Замечание: После служебного слова RETURN может быть несколько команд, которые располагаются между словами BEGIN и END. В этом случае служебное слово RETURN не ставится.

Замечание: Тип данных параметра должен совпадать с типом данных выражения, в котором он используется.

Замечание: Если используются несколько SQL команд и BEGIN и END, то перед END нужно ставить команду RETURN <результат функции>.

Пример (скалярная пользовательская функция): Функция для вычисления среднего 3 чисел:

```
CREATE FUNCTION Среднее
(@ X1_Int,@X2_Int,@X3_Int)
RETURNS Real
AS
```

```
BEGIN
DECLARE_@Res_Real
@Res =(@X1+@x2+@x3)/3
RETURN_@Res
END
```

Замечание: Команда DECLARE создает переменную Res для хранения дробных чисел (тип данных Real).

Представленная выше пользовательская функция реализована при помощи нескольких команд SQL, но ее можно реализовать при помощи одной функции следующим образом:

```
CREATE FUNCTION Среднее
(@X1_Int,@X2_Int,@X3_Int)
RETURNS Real
AS
RETURN (SELECT (@X1+@x2+@x3)/3)
```

Созданная функция, вычисляющая среднее 6, 3 и 3, запускается следующим образом:

```
SELECT Среднее (6, 3, 3)
```

Результат будет 4.

Пример (табличная пользовательская функция): Из таблицы Студенты выводятся поля ФИО, дата рождения и столбец возраст, который вычисляется как разница дат в годах, между датой рождения и текущей датой (параметр CurDate).

```
CREATE FUNCTION Возраст
(@CurDate Date)
RETURNS TABLE
AS
RETURN (SELECT ФИО, [Дата рождения], Возраст = DATEDIFF
(yy,[Дата рождения], @CurDate)
FROM Студенты)
```

Данная функция вызывается следующим образом:

```
SELECT Возраст ('12/17/2007')
```

В результате отобразятся студенты с их возрастом на 17 декабря 2007 года.

На этом мы заканчиваем рассмотрение пользовательских функций. Дополнительную информацию можно найти в лабораторной работе №6.

Занятие 7. Целостность данных. Диаграммы и триггеры

Цели:
1. Изучить порядок обеспечения целостности данных
2. Понятие порядок создания триггеров

Целостность данных

При работе БД должна обеспечиваться целостность данных. Под целостностью данных понимают обеспечения целостности связей между записями в таблицах при удалении записей из первичных таблиц. То есть, при удалении записей из первичных таблиц автоматически должны удаляться связанные с ними записи из вторичных таблиц.

В случае несоблюдения целостности данных в со временем в БД накопится большое количество записей во вторичных таблицах связанных с несуществующими записями в первичных таблицах, что приведёт к сбоям в работе БД и её засорению неиспользуемыми данными.

Для обеспечения целостности данных в SQL Server используют диаграммы и триггеры.

Диаграммы – это компоненты БД, которые блокируют удаление записей из первичных таблиц если существуют связанные с ними записи во вторичных таблицах. Следовательно, диаграммы предотвращают нарушение целостности данных. В SQL Server диаграммы создаются при помощи мастера диаграмм, его описание представлено в лабораторной работе.

Триггеры – это аналог процедур обработчиков событий в Visual Basic. То есть они выполняют команды SQL если происходят какие либо действия с таблицей (Например: добавление, изменение или удаление записей). При помощи триггеров можно организовать автоматическое удаление записей из вторичной таблицы при удалении связанной с ними записи из первичной таблицы.

Рассмотрим создание триггеров при помощи языка SQL.

Создание триггеров

В SQL Server существуют два вида триггеров:

1. Триггеры выполняемые после события, произошедшего с таблицей (Полный аналог процедур событий в Visual Basic);
2. Триггеры выполняемые вместо события, происходящего с таблицей. В этом случае событие (добавление, изменение или удаление записей) не выполняется, а вместо него выполняются SQL команды заданные внутри триггера.

Первый вид триггеров применяется для обработки событий таблиц, а второй – для обеспечения целостности данных, то есть удаление записей из вторичной таблицы при удалении связанной с ними записи из первичной таблицы.

Замечание: Триггеры создаются для конкретной таблицы и выполняются автоматически если с таблицей, для которой они были созданы происходит событие (добавление, изменение или удаление записей).

Для создания триггера на вкладке нового запроса необходимо набрать команду CREATE TRIGGER, имеющую следующий синтаксис:

```
CREATE TRIGGER <Имя триггера>
ON <Имя таблицы>
FOR <AFTER|INSTEAD OF> <INSERT|UPDATE|DELETE>
[WITH ENCRYPTION]
AS <Команды SQL>
```

Здесь Имя триггера – это имя создаваемого триггера.
Имя таблицы – имя таблицы, для которой создаётся триггер.
Если используется параметр ALTER, то триггер выполняется после события, а если параметр INSTEAD OF, то выполняется вместо события.
Параметры INSERT, UPDATE и DELETE определяют событие, при котором (или вместо которого) выполняется триггер.
Параметр WITH ENCRYPTION – предназначен для включения шифрования данных при выполнении триггера.
Команды SQL – это SQL команды, выполняемые при активизации триггера.

Рассмотрим примеры создания различных триггеров для таблицы «Студенты».

Пример: Создаёт триггер «Добавление», выводящий на экран с сообщение «Запись добавлена» при добавлении новой записи в таблицу «Студенты»

```
CREATE TRIGGER Добавление
ON Студенты
FOR AFTER INSERT
AS PRINT 'Запись добавлена'
```

Пример: Создаёт триггер «Изменение», выводящий на экран с сообщение «Запись изменена» при изменении записи в таблице «Студенты»

```
CREATE TRIGGER Изменение
ON Студенты
FOR AFTER UPDATE
AS PRINT 'Запись изменена'
```

Пример: Создаёт триггер «Удаление», выводящий на экран с сообщение «Запись удалена» при удалении записи из таблицы «Студенты»

```
CREATE TRIGGER Удаление
ON Студенты
FOR AFTER DELETE
AS PRINT 'Запись удалена'
```

Пример: В данном примере вместо удаления студента из таблицы «Студенты» выполняется код между BEGIN и END. Он состоит из двух команд DELETE. Первая команда удаляет все записи из таблицы «Оценки», которые связаны с записями из таблицы «Студенты». То есть у которых Оценки.[Код студента] равен коду удаляемого студента. Затем из таблицы «Студенты» удаляется сам студент.

```
CREATE TRIGGER УдалениеСтудента
ON Студенты
INSTEAD OF DELETE
AS
BEGIN
```

```
DELETE Оценки
FROM deleted
WHERE deleted.[Код студента]=Оценки.[Код студента]
DELETE Студенты
FROM deleted
WHERE deleted.[Код студента]=Студенты.[Код студента]
END
```

Замечание: Здесь удаляемая запись обозначается служебным словом deleted.

Замечание: Для обеспечения целостности данных триггеры используют обычно вместе с диаграммами, но мы можем применять такие триггеры и без диаграмм, однако мы не можем применять диаграммы без триггеров.

На этом мы заканчиваем рассмотрение диаграмм и триггеров. Дополнительную информацию можно найти в лабораторной работе №7.

<u>Занятие 8.</u> Общая характеристика языка Visual Basic 2008. История создания и системные требования. Объекты связи. Мастер подключений

Цели:

1. Изучить общую характеристику языка, историю создания и системные требования

2. Рассмотреть объекты связи

3. Освоить работу мастера подключений

Общая характеристика языка. История создания и системные требования

Язык программирования Visual Basic 2008 входит в состав пакета Microsoft Visual Studio 2008. Он позволяет создавать приложения для ОС Windows 2000, XP, VISTA и ОС Windows Mobile, Windows Pocket PC.

Microsoft Visual Basic 2008 обладает следующими особенностями:

1. Для работы программ, написанных на этом языке, необходимо чтобы была установлена библиотека Microsoft Net. Frame Work 2.0
2. Возможность создавать различные части проекта на различных языках программирования, входящих в Visual Studio.
3. Возможность использования новых визуальных эффектов доступных Windows XP.
4. Возможность конвертации проектов Visual Basic более ранних версий.
5. Большая ориентация на сетевые технологии.
6. Более упрощенная работа с БД. Ориентация на язык форматирования XML. В состав Visual Studio входит SQL Server Express – урезанная клиентская версия SQL Server 2008.
7. Автоматическое подключение всех доступных компонентов.

По сравнению с Visual Basic6.0, Visual Basic 5.0 и Visual Basic2005, Visual Basic 2008 обладает большими системными требованиями. Для его работы необходим компьютер, имеющий следующую конфигурацию:

1) Процессор Pentium 600 МГерц и выше,
2) 256 Мб памяти.
3) Для установки только Visual Basic необходим 1Гб свободного места на диске, а для установки всего пакета Visual Studio необходимо 4Гб .

Visual Basic 2008 основывается на ядре Visual Basic 6.0, который входит в состав Visual Studio 6.0. И был создан в 1998. После создания Visual Studio 6.0, он получил большое распространение в мире. В 2003г была создана новая версия Visual Basic Net. Его главным отличием была большая ориентация на компоненты сети, использования библиотеки Microsoft Frame Work 1.0. улучшились графические спецэффекты программы. В 2005 году на основе Visual Basic Net создается Visual Basic 2005, а в 2008 году создаётся улучшенная версия языка Visual Basic 2008.

Создание интерфейса клиентского приложения в Visual Basic происходит несколько этапов:

1) Создаётся проект;
2) В проекте создаются объекты связи, которые подключаются к файлу данных;
3) Создаются формы;
4) Создаются отчёты.

Создание нового проекта рассматривается в лабораторной работе и литературе посвящённой Visual Basic. Остановимся более подробно на объектах связи.

Объекты связи

Объекты связи - это объекты проекта, осуществляющие обмен информацией между интерфейсом БД и файлом данных.

Объекты связи всегда находятся на клиентской машине. Они осуществляют доступ к файлам данных, передавая информацию в интерфейс БД, и содержат внутри себя запросы, выполнения на стороне клиента.

Замечание: Объекты связи также могут ограничивать доступ к информации и осуществлять защиту информации, хотя для защиты информации и ограничения доступа лучше использовать сам сервер.

Существует три технологии используемых в объектах связи:

- технология ADO;
- технология RDC;
- технология ADO.Net.

ADO является более старой технологией. Её суть заключается в следующем: подключение к конкретной таблице или запросу, осуществляется через отдельный объект связи, т.е. все настройки и средства для работы с данными хранятся внутри конкретного объекта связи и были заложены туда при его проектировании.

Согласно технологии RDC файлы данных рассматриваются в качестве устройств, т.е. для работ с БД нам необходим драйвер. Объект связи, работающий по технологии RDC, при работе с файлом данных сначала обращается к драйверу БД, который в свою очередь обращается к файлу данных.

Технология ADO.Net является смесью технологий ADO и RDC. Объекты связи работающие по этой технологии работают аналогично объектам работающим по технологии ADO, однако, объекты связи входят в состав пакета Microsoft Net Framework, и автоматически обновляются вместе с этим пакетом.

Плюсы и минусы технологии ADO:
+ независимость от драйверов БД, установленных в операционной системе;
+ простое программирование;
- невозможность работать с новыми типами БД;
- невозможность обновлять список поддерживаемых БД.

Плюсы и минусы технологии RDC:
+ возможность работать с современными БД;
+ возможность добавлять новые виды БД;
- зависимость от драйверов, установленных в системе;
- более сложное программирование.

Плюсы и минусы технологии ADO.Net:
+ возможность работать с современными БД;
+ возможность добавлять новые виды БД;
- зависимость от пакета Microsoft Net Framework;
- более сложное программирование.

Замечание: Мы можем создавать динамические запросы и запросы, выполненные на стороне сервера только в технологии RDC и ADO.Net.

Мастер подключений

В Visual Basic 2008 по сравнению с Visual Basic 6.0 подключение проекта к файлу БД можно произвести двумя способами: при помощи мастера подключений и вручную, создавая объекты связи и настраивая их свойства. Начнем рассмотрение создания подключения с помощью мастера.

Как говорилось выше, объекты связи обеспечивают доступ к файлам данных. Создание подключения состоит из создания следующих объектов:

- DataSet (Набор данных) – обеспечивает подключение формы к конкретной БД на сервере (в нашем случае это БД Students);
- BindingSource (Источник связи) – обеспечивает подключение к конкретной таблице (в нашем случае к таблице специальности), а также позволяет управлять таблицей;
- TableAdapter (Адаптер таблиц) – обеспечивает передачу данных с формы в таблицу и наоборот.
- TableAdapterManager (Менеджер адаптера таблиц) – управляет работой объекта TableAdapter;
- BindingNavigator (Панель управления таблицей) – голубая панель с кнопками управления таблицей, расположенная в верхней части формы.

Можно создать и подключить все эти объекты вручную, но удобнее воспользоваться мастером. Работа с мастером подключений состоит из нескольких этапов:

1) Запуск мастера;
2) Выбор типа источника данных: БД, сетевой источник или объект;
3) Настройка строки подключения «Connection String». Настройка заключается в выборе вида БД (либо Access, либо SQL Server), а также в выборе сервера и файла данных. В случае необходимости можно задать логин и пароль;
4) Сохранение строки подключения. При ее сохранении можно менять параметры подключения без использования Visual Basic. Но при сохранении строки подключения в файл велика вероятность несанкционированного подключения к БД;
5) Выбор таблиц или запросов включённых в соединение. Также можно выбрать их отдельные поля;
6) Завершение работы мастера подключений.

Боле подробные инструкции по работе с мастером подключений можно найти в лабораторной работе.

Замечание: После окончания работы мастера подключений. В обозревателе в «Solution Explorer» появится дополнительный файл набора данных с расширением xsd. Этот файл содержит в себе схему данных из источника данных, а также позволяет редактировать источник данных (при открытии этого файла появляется окно похожее на конструктор запросов в Access или SQL Server), в этом окне также можно редактировать поля таблиц.

Замечание: В одном проекте может быть несколько наборов данных, то есть можно запускать мастер подключений сколько угодно раз. Новые наборы данных добавляются на вкладку «Data Sources» и появляется новые данные с расширением xsd.

Настройка связи подключение вручную

В Visual Basic 2008, как и в Visual Basic 6.0 мы можем создавать объекты связи вручную и их настраивать. Для связи Visual Basic 2008 использует три объекта связи, причем они работают все вместе, плюс к этому были существует объект BindingNavigatior

(Панель навигации) – эта панель обеспечивает полное управление источником данных (добавление, удаление, перемещение по записям).

Рассмотрим создание и настройки соответствующих объектов связи в порядке очередности:

1) Создание подключения начинается с создания объекта DataSet. Объект DataSet не может сам подключится к источнику данных перед его созданием необходимо настроить «Data Sources» (оконное меню Data \ Add Data Sources). После создания объекта DataSet появляется окно «Add DataSet». В нем необходимо в выпадающем списке «Typed DatSet» выбрать источник данных из «Data Sources». Фактически «DataSet» аналогичен Connection из Visual Basic 6.0. После выбора источника данных в списке «Typed DataSet» появится строка Windows Application <имя источника>. После этого в окне можно нажать кнопку «Ok». Имя источника данных будет записана в свойство DataSetName объекта DataSet.
2) После создания объекта DataSet создается объект BindingSource. Этот объект играет ту же роль, что и Command в Visual Basic 6.0, он позволяет подключится к таблицам, запросам и фильтрам из файла данных. После его создания необходимо настроить следующие свойства:
   - DataSource – указанный объект DataSet;
   - DataMember – указывает таблицу, запрос или фильтр, которые будут отображаться на форме.

   Следующие свойства необязательны для настройки:
   - Filter – свойство для фильтрации данных, в нем записывается условие отбора для какого-то поля;
   - Sort – сортировка информации
   - Allow New – позволяет добавлять новые записи.
3) После добавления DataSet и BindingSource автоматически будет добавлен объект TableAdaper. После чего уже можно добавлять объекты для отображения данных, однако, при этом нельзя будет управлять информацией.
4) Для управления источником данных создаётся объект BindingNavigatior. Затем его необходимо подключить к объекту BindingSource. Для этого в свойстве BindingSource объекта BindingNavigatior необходимо указать созданный ранее объект BindingSource.

   Затем можно настроить внешний вид панели навигации при помощи следующих свойств:

   AddNewItem – отображает кнопку для добавления новой записи;
   DeleteItem – отображает кнопку для удаления текущей записи;
   AddNextItem – отображает кнопку для добавления новой записи после текущей;
   MoveFirstItem – отображает кнопку для перехода к первой записи;
   MoveNextItem – отображает кнопку для перехода к следующей записи;
   MovePreviousItem – отображает кнопку для перехода к предыдущей записи;
   MoveLastItem – отображает кнопку для перехода к последней записи;
   CountItem – отображает общее количество записей;
   Position Item – отображает номер текущей записи.

На этом мы заканчиваем рассмотрение подключение источника данных к проекту. Дополнительную информацию можно найти в лабораторной работе №8.

Занятие 9. Интерфейс информационных систем. Создание интерфейса пользователя

Цели:
1. Рассмотреть интерфейс информационных систем
2. Изучить порядок создания интерфейса пользователя
3. Освоить подключение объектов к источнику данных при помощи окна свойств

Интерфейс информационных систем

В системах построенных по технологии клиент-сервер существует два вида интерфейса:

- Интерфейс, реализуемый при помощи клиентского приложения;
- Web –интерфейс.

Интерфейс, реализуемый при помощи клиентского приложения – это компьютерная программа, устанавливаемая на клиентские компьютеры, предназначенная для работы с файлами данных через сеть. Основными элементами клиентских приложений являются формы (окно программы) и отчёты.

Элементы управления на форме называется объектами. Каждый объект обладает своим набором свойств, событий и методов.

- Свойства объекта – это его характеристики (высота, ширина и т.д.);
- События объекта – это события операционных систем или события инициируемые пользователем, на которые может реагировать объект (нажатие кнопки);
- Методы объекта – действия, которые можно производить с объектом в ходе выполнения программ.

В БД все объекты форм делятся на два класса:

- Объекты управления – объекты, осуществляющие управление БД (Например: Кнопка или Выпадающий список);
- Объекты для отображения информации – элементы, отображающие содержимое таблиц, запросов или фильтров, позволяющие добавлять изменять и удалять информацию, и проводить ее анализ.

Все формы в клиентском приложении делятся на три группы:

1. Формы для работы с данными – формы, содержащие как объекты управления, так и объекты просмотра данных. Такие формы предназначены для отображения, изменения, удаления и анализа данных;
2. Кнопочные формы – формы, содержащие только объекты управления, предназначаются для открытия всех других форм.

Замечание: Кнопочная форма, которая появляется первой после запуска программы, называется, главной кнопочной формой.

3) Информационные и служебные формы – формы, содержащие только элементы управления, предназначены для отображения служебной информации (справки), несвязанной с таблицами, запросами и фильтрами, либо для выполнения служебных операций не связанных с данными (Например: форма с калькулятором)

Замечание: Существует два вида дизайна форм:
1) Ленточные формы - формы, выводящие информацию по одной записи;
2) Табличные формы – формы выводящие информацию в виде таблицы.

Замечание: Объекты связи используются только в клиентском интерфейсе. В web-интерфейса функции объекта связи выполняет сервер.

Основой web-интерфейса являются страницы (файл с расширенным htm или html). Работа со страницами осуществляется с помощью программы - браузера. Изначально страницы находятся на сервере, пользователь сначала загружает их на свой компьютер с сервера, а затем с помощью страниц пользователь работает с файлом данных.

Замечание: В web- интерфейсе отсутствуют отчёты, их роль выполняют сами страницы.

Создание интерфейса пользователя

Создание интерфейса при помощи окна «Data Sources»

Visual Basic 2008 позволяет создавать не сложный интерфейс БД, без помощи панели объектов и окна свойств, лишь использую окно «Data Sources». В окне «Data Sources» после подключения источника данных отображается все таблицы, запросы, фильтры данных и их поля. В Visual Basic 2008 как и в Visual Basic 6.0 можно перетаскивать источники данных, соответственно таблицы, запросы, фильтры прямо из окна «Data Sources» на форму (в Visual Basic 6.0 окно «Data Sources» заменено на окно «Data Environment»). Главное отличие Visual Basic 2008 является то, что при перетаскивании можно выбирать для каждого поля источника данных объект, который будет отображать его содержимое.

Замечание: Таким способом можно создавать только определённые объекты для отображения данных поля, и набор этих объектов зависит от типа данных поля.

Создание объектов для отображения данных перетаскиванием состоит из двух шагов:

- Для каждого поля таблицы, запроса, или фильтра выбирается объект, который будет отображать его содержимое. Для этого необходимо щелкнуть мышью по полю в окне «Data Sources», рядом с именем поля появится кнопка, со стрелкой, щелкнув мышью стрелке, отобразится выпадающее меню с объектами, которые могут отображать информацию, содержащуюся в поле. Для полей стандартными объектами являются: TextBox, ComboBox, Label, LinkLabel, ListBox, LinkLabel. Для полей типа данных Дата Время (DateTime) возможно использования объекта DataTimePicker. Для полей логических типов данных возможно использование объекта CheckBox. Для отображения таблиц, запросов или фильтров целиком возможно два варианта отображения: 1) При помощи объекта DataGridView – информация из таблицы, запроса или фильтра отображается в виде таблицы; 2) DetalledView – отображение всех полей источника данных в TextBox по отдельности

Замечание: В выпадающем меню с вариантами выбора объектов имеется пункт «Customize» (Настройки), который позволяет выбрать дополнительные допустимые объекты для отображения информации.

- после выбора объектов для отображения необходимо их поместить на форму, перетаскивая мышью с панели «Data Sources» в нужное место на форме.

Замечание: При помещении первого объекта на форму на ней автоматически создаются объекты для связей с файлом данных и объекты по навигациям по источникам данных (DataSet, BindingSource, TableAdapter, BindingNavigator).

Замечание: По умолчанию панель навигации располагается в верхней части формы. Эту панель можно прикрепить около различных краев формы. Для этого необходимо воспользоваться меню действий объектов. Это меню схоже с окном «Property Pages» из Visual Basic 6.0, но кроме основных свойств объектов оно содержит и действия, которые можно производить с объектами. Чтобы вызвать это меню необходимо выделить объект. В его правом верхнем углу появится (квадратик со стрелочкой) при нажатии этой кнопки появляется выпадающее меню с настройками и действия с объектом. Например, чтобы поменять место положение навигации панели надо в этом меню выбрать настройку Docking.

Замечание: При перетаскивании на форму полей источников данных автоматически создаются подписи к ним (Label).

Замечание: После перетаскивания с созданным объектом можно работать как и с обычным объектом Visual Basic.

Подключение объектов к источнику данных при помощи окна свойств

Visual Basic 2008 позволяет подключать источники данных к объектам без использования перетаскивания, то есть вручную, с использованием панели свойств, как в Visual Basic 6.0. Для этого на форму помещается объект, который будет подключаться к источнику данных. Его выделяют, затем на панели свойств разворачивается группа свойств «DataBindings» она содержит два свойства:

- Text – определяет таблицу, запрос или фильтр, из которого выводятся данные в объект;
- Tag – определяет поле, выбранного в свойстве Text источника данных, которое отображается в объекте.

На этом мы заканчиваем рассмотрение простых ленточных форм для работы с данными. Дополнительную информацию можно найти в лабораторной работе №9.

<u>Занятие 10.</u> Стандартные объекты для отображения данных. Программное управление информационной системой

Цели:
1. Изучить стандартные объекты для отображения данных
2.Рассмотреть программное управление информационной системой

Стандартные объекты для отображения данных

Способ создания объектов для отображения данных описанный ранее, позволяет создавать только ограниченный набор объектов. Однако, Visual Basic 2008 позволяет подключать источники данных практически к любому объекту, который может быть создан на форме. Это можно сделать при помощи перетаскивания поля источника данных из окна «Data Sources» на объект на форме.
Операция состоит из двух шагов:

- При помощи панели объектов (слева) на форме создается какой-то объект. Объекты несвязанные с источником данных называют несвязанными объектами;
- Вновь созданный объект связывается с источником данных. Для этого на объект нужно перетащить поле таблицы запроса или фильтра из окна «Data Sources».

<u>Замечание:</u> При перетаскивании поля из окна «Data Sources» необходимо учитывать его тип данных. Объект на форме должен поддерживать тип данных подключаемого к нему поля.

<u>Замечание:</u> В случае подключения объекта к источнику данных, способом, описанным выше, подпись к объекту не создаётся автоматически и её надо создавать вручную с помощью объекта Label.

Наиболее часто в БД используются следующие объекты для отображения информации:

1. Текстовое поле (TextBox)
2. Надпись (Label)
3. Надпись со ссылкой (LinkLabel)
4. Календарь (DataPicker)
5. Переключатель (CheckBox)
6. Таблица (DataGridView)
7. Список (ListBox)
8. Выпадающий список (ComboBox)
9. Текстовое поле с маской ввода (MaskedTextBox)

TextBox – отображает текст и числовые поля, это наиболее часто употребляемый объект для отображения данных. Его можно создавать либо перетаскиванием из окна «Data Sources», либо подключить вручную. Создание этого объекта, перетаскиванием возможно почти у полей любых типов данных.
Label – полностью аналогичен объекту TextBox, но не позволяет изменить данные. Этот объект используется для отображения заблокированных неизменяемых полей.

LinkLabel – специальный объект для отображения ссылок на адреса в Интернете. Его используют для отображения текстовых полей, если в них хранятся адреса Интернета или какой-то компьютерной сети. Это новый объект, ему не было аналога в Visual Basic 6.0.
DataPicker – специальный объект, предназначенный для отображения полей типа данных «Дата/Время» в виде календаря.
CheckBox – объект используется для отображения логических полей, может быть создан перетаскиванием только для логических полей.
DataGridVilew – объект, отображающий источник данных (таблицу, запрос или фильтр) в виде таблицы.
ListBox – список отображающий значения полей и позволяющий выбирать значения полей из списка. Более того, пункты списка можно задавать, используя другой источник данных.
ComboBox – объект подобный объекту LiasBox, однако информация отображается не в списке, а выпадающем списке.
MaskedTextBox – нестандартный объект, предназначенный для отображения и ввода информации по заранее заданному шаблону (маске). Этот объект может быть создан только при помощи панели объектов и его подключение осуществляется либо перетаскиванием на него поля из окна «Data Sources», либо заданием его свойств вручную. По своим свойствам он ничем не отличается от объекта TextBox. Единственное дополнительное свойство у этого объекта это свойство Mask. Для этого нужно щелкнуть по кнопке действий объекта в верхнем правом углу объекта. Затем в списке действий выбрать пункт «Edit Mask». В появившемся окне выбрать шаблон ввода, то есть маску (Mask).

Замечание: Тип данных отображаемой информации должен совпадать с типом данных маски.

Замечание: Объекты ListBox и ComboBox могут использоваться для заполнения полей с кодами, то есть списки заполняются информацией из одной таблицы, а при выборе пункта списка его код подставляется в другую таблицу. Для этого на форму располагают не подключенный ListBox или ComboBox, затем открываем его меню действий. В меню действий в указанной последовательности выполняют следующие шаги:

- Установить галочку «Use Data Bound Items»,
- В выпадающем списке «Data Source» выбрать пункт «Other Data Source» и там выбрать нужную таблицу.
- В выпадающем списке «Display Member» и указываем поле, которое отображается в списке.
- В выпадающем списке «Value Member» указываем поле, которое подставляем при выборе пункта списка.
- В выпадающем списке «Selected Value» указываем поле, куда подставляется выбранное в «Value Member» значение.

Программное управление информационной системой

В Visual Studio 2008 добавлять, удалять записи и перемещаться ним можно как используя объект Navigator, так и используя обычные кнопки. Рассмотрим создание кнопок для управления записями. В Visual Studio 2008 отсутствует объект «RecordSet» все операции с записями осуществляются с использованием объекта «BindingSource».

В Visual Basic 2008 для добавления новой записи из таблицы «Студенты» используется команда вида СтудентыBindingSource.AddNew. Вместо метода AddNew можно использовать методы:

- MoveNext (перейти к следующей);

- MoveFirst (Перейти к первой);
- Move Previous (Перейти к предыдущей);
- Move Last (Перейти к последней);
- Delete (Удалить запись).

У объекта BindingSource имеется свойство Filter. Оно аналогично Filter у объекта RecordSet в Visual Basic 6.0. Его использование ничем не отличается от исполнения такого же свойства в Visual Basic 6.0. То есть в свойстве Filter задаётся строка, определяющая условие отбора записей в динамических фильтрах, выполняемых на стороне клиента. Данная строка имеет следующий синтаксис:

```
<Поле1><Оператор1><Выражение1>
[AND|OR <Поле2><Оператор2><Выражение2>…]
```

Здесь <Поле1>, <Поле2> ... – поля на которые накладываются условия;
<Оператор1>, <Опрераторы2> - операторы сравнения, участвующие в условиях;
<Выражение1>, <Выражение2> - выражения с которыми сравниваются поля. Под выражениями понимаются, константы, переменные, формулы, функции и свойства объектов.

Пример: Из таблицы «Студенты» необходимо отобразить студента, у которого значение поля ФИО равно «Петров».

```
СтудентыBindingSource.Filter = «ФИО = 'Петров'»
```

Обычно при формировании запроса при помощи свойства Filter задания условий отбора используют либо списки ListBox, либо выпадающие списки ComboBox.

Замечание: Если мы используем ComboBox для создания динамического фильтра, то в меню действий параметры «Value Member» и «Selected Value» настраивать не надо.

Пример: Имеется таблица «Студенты», которая отображается на форме в DataGridView. Необходимо на форме поместить ComboBox с фамилиями студентов. При выборе ФИО и нажатием на кнопку отобразить данные только по выбранному студенту.

В этом случае в меню действий ComboBox в параметре «Data Source» указываем «Other Data Source/Студенты». Затем в «Display Member» выбираем ФИО. В коде кнопки прописываем следующую команду:

```
СтудентыBindingSource.Filter = «ФИО=’» & ComboBox1.Text  & «’»
```

После нажатия кнопки в DataGridView отображаются данные по студенту, выбранному в выпадающем списке ComboBox1.

Дополнительную информацию можно найти в лабораторной работе №10.

<u>Занятие 11.</u> Объект для отображения табличной информации DataGridView. Настройка свойств столбцов в DataGridView

Цель:
1. Изучить объекты для отображения табличной информации DataGridView
2. Рассмотреть настройка свойств столбцов в DataGridView

Объект для отображения табличной информации DataGridView

Объект DataGridView предназначен для отображения всей информации из таблиц, запросов или фильтров на форме в виде таблицы. Этот объект может быть создан как вручную (с последующим его подключением), так и перетаскиванием всего источника данных из окна «Data Sources». Однако наиболее часто его создают перетаскиванием всей таблицы, запроса или фильтра из окна «Data Sources» на форму.

При перетаскивании этого объекта на форму, как и в случае с другими объектами появляется панель навигации. Она выполняет функции: перемещение по записям, добавление, удаление и сохранение записей. После создания объекта DataGridView можно настраивать как свойства всего объекта, так и свойства отдельных столбцов. Начнём с настройки свойств всего объекта. Настройка данных свойств осуществляется в основном через меню действий. Возможны следующие настройки:
Chose Data Source – источник данных, отображаемый в таблице;
Enable Adding - добавлять записи;
Enable Deleting - разрешается пользователям удалять записи;
Enable Editing - разрешается пользователям изменять значения полей таблицы;
Enable Column Reordering - разрешается пользователям изменять порядок столбцов, просто перетаскивая их мышью.

Также в меню действий возможны следующее действия с таблицей:
Dock in parent container – вписать объект в форму;
Preview Data – появляется окно с предварительным просмотром таблицы;
Add Query – добавляет SQL – запрос, который выполняется на стороне клиента;
Add Column – добавление нового столбца в таблицу;
Edit Columns – настойка свойств отдельных столбцов таблицы.

Теперь перейдём к настройке отдельных столбцов таблицы.

Настройка свойств столбцов в DataGridView

Если в меню действий выбрать пункт «Edit Columns», то появляется окно, где можно добавлять, удалять и редактировать столбцы. Для этого в списке столбцов левой части окна выбираем столбец, а в правой – настраиваем его свойства. Наиболее часто настраиваются следующие свойства:

1. Name – имя столбца;
2. AutoSizeMode – подгонка ширины столбца по его содержимое;
3. ColumnType – определяет внешний вид ячеек столбца (какой объект для отображения информации находится в ячейках столбца);
4. DataPropertyName – имя, отображающего в столбце поля;
5. Frozen – фиксация столбца (столбец не передвигается при прокручивании

таблицы);
6. HeaderText – текст заголовка столбца;
7. Width – ширина поля;
8. MaxInputLength – максимально вводимая длина текста;
9. MinimumWidth – минимальная ширина столбца;
10. ReadOnly – блокировка столбца для редактирования данных;
11. Resizable – разрешает менять ширину столбца;
12. SortMode – сортировка данных в таблице по этому столбцу;
13. ToolTipText – всплывающая подсказка для столбца;
14. Visible – делает столбец невидимым.

Замечание: Для добавления новых столбцов в окне «Edit Columns» необходимо нажать кнопку Add, а для удаления кнопку Remove.

Замечание: Если необходимо настроить внешний вид всех ячеек таблицы, то для этого необходимо выделить объект DataGridView и на панели свойств зайти в свойство DefaultCellStyle. Появится окно со свойствами всех ячеек таблицы.

Замечание: В объекте DataGridView имеется возможность сортировки данных. Для этого используется метод Sort, имеющий следующий синтаксис:

```
DataGridView.Sort(<Имя столбца>, <Порядок сортировки>)
```

где DataGridView – это имя объекта, <Имя столбца> - это имя столбца (свойство Name) по которому происходит сортировка записей в таблице, параметр <Порядок сортировки> определяет порядок сортировки и может принимать два значения:

- `System.ComponentModel.ListSortDirection.Ascending` – сортировка по возрастанию;
- `System.ComponentModel.ListSortDirection.Descending` – сортировка по убыванию.

Замечание: Доступ к отдельным ячейкам таблицы можно получить через подобъект Item. Обращение к нему осуществляется следующим образом:

```
DataGridView.Item(i, j).<Свойство>
```

Здесь DataGridView – это имя объекта, i – горизонтальная координата ячейки, а j – вертикальная, <Свойство> - это настраиваемое свойство ячейки.

Пример: В верхнюю левую ячейку таблицы записать слово «Привет» и сделать цвет текста в ячейке красным.

```
DataGridView.Item(0, 0).Value = «Привет»
DataGridView.Item(0, 0).Style.ForeColor = Color.Red
```

Здесь DataGridView – это имя объекта, свойство Value определяет содержимое ячейки таблицы, свойство Style.ForeColor определяет цвет текста в ячейке. Нумерация столбцов и строк в таблице начинается с нуля.

На этом мы заканчиваем рассмотрение табличных форм для работы с данными. Дополнительную информацию можно найти в лабораторной работе №11.

Занятие 12. Отчёты. Объекты для работы с отчётами

Цели:

1. Изучить порядок создания отчетов

2. Рассмотреть объекты для работы с отчетами

Отчёты

Клиентские приложения осуществляют вывод информации на печать с помощью отчетов. Отчеты так же, как и формы состоят из объектов и сами являются объектами, но между отчетами и формами есть отличия:

1. Отчёты содержат только объекты для отображения информации (Например, подписи, рисунки, текстовые поля, геометрические фигуры и линии), но не содержат объекты для управления системой (Например, кнопки или выпадающие списки);
2. В отчётах сразу же выводиться все записи из источника данных (таблицы, запроса или фильтра) и вывод производится на листы;
3. Отчёт нельзя создать без наличия в системе принтера, так как настройки внешнего вида отчёта берутся из настроек драйвера принтера;
4. В отличие от форм отчёты состоят из пяти разделов:
    - Заголовок – верхняя часть первого листа отчёта. В заголовке располагают название отчёта и некоторую служебную информацию. Например, герб и юридический адрес фирмы или имя автора отчёта.
    - Примечание – нижняя часть последнего листа отчёта. В примечание помещают итоговую информацию по отчёту (Например, общий объём продаж, всех сделок представленных в отчёте) и место для печати и подписи руководителя.
    - Верхний колонтитул – верхняя часть каждого листа отчёта. В данный раздел помещают номера листов отчёта и дополнительную служебную информацию. Например, дату и время создания отчёта.
    - Нижний колонтитул – нижняя часть каждого листа отчёта. В данном разделе располагается та же информация что и в верхнем колонтитуле, но не дублирует информацию из верхнего колонтитула.
    - Область данных – средняя часть каждого листа отчёта.

Замечание: Существует два вида дизайна отчетов:

- Ленточный дизайн - выводит информацию по каждой записи отдельно. То есть для каждого поля каждой записи отображается название поля и его значение;
- Табличный дизайн – выводит информацию в виде таблицы. То есть в заголовок отчёта помещают названия полей, а в области данных под названием полей отображаются их значения.

Объекты для работы с отчётами

Работа с отчётами в Visual Basic2008 состоит из нескольких этапов:

1. Создаётся пустой отчёт;
2. В отчёт помещают объекты для отображения информации;
3. Создаётся форма для отображения отчёта;
4. На форму помещают объект Reportvewer, отображающий отчёты;

5. К объекту Reportvewer подключают созданный ранее отчёт.

Создание отчётов и отображающих их форм подробно рассмотрено в лабораторной работе. Остановимся более подробно на отчёте и объектах, используемых при его создании.

Для создания пустого отчёта в оконном меню необходимо выбрать пункт «Project\Add New Item…» и в появившемся окне в разделе «Reporting» дважды щёлкнуть ЛКМ по пункту «Report» появится вкладка с пустым отчётом. Теперь в отчёт необходимо добавить объекты для отображения данных.

Замечание: Работа с объектами в отчёте полностью аналогична работе с объектами на форме. То есть мы можем перетаскивать поля в отчёт из окна «Data Sources» либо можем создать объекты в отчёте вручную, а затем подключить их к полям через панель свойств.

Замечание: В отчётах все объекты делятся на объекты контейнеры, объекты для отображения данных и объекты оформления.

- Объекты контейнеры – это объекты, содержащие объекты для отображения данных и определяющие дизайн отчёта.
- Объекты для отображения данных – это объекты, отображающие значения полей источника данных или дополнительную служебную информацию.
- Объекты оформления – объекты, применяемые только для оформления отчёта.

Рассмотрим объекты для отображения данных, к ним относятся:

- TextBox – текстовое поле ввода, предназначено для отображения значений полей и любой текстовой информации. Если объект TextBox используется для отображения информации из источника данных, и он находится вне объекта контейнера, то в нём будет отображено значение выбранного поля только первой записи из источника данных;
- Image – объект отображающий содержимое полей с графической информацией либо отображающий рисунки (графические файлы);
- Chart – объект, отображающий график или гистограмму построенную по информации из источника данных.

Теперь рассмотрим объекты контейнеры. В отчёт можно поместить следующие объекты контейнеры:

- Table – таблица выводит информацию в виде таблицы с ограниченным количеством столбцов и неограниченным количеством строк. То есть в количество строк в таблице зависит от объёма выводимых данных;
- Matrix - таблица выводит информацию в виде таблицы с неограниченным количеством столбцов и строк. То есть в количество строк и столбцов в таблице зависит от объёма выводимых данных;
- List – объект выводящий информацию в виде списков;
- Subreport – объект содержащий внутри себя дополнительный отчёт, созданный ранее.

Наконец, рассмотрим объекты оформления. К ним относятся:

- Line – отображает линию;
- Rectangle – отображает прямоугольник, используется для группировки полей.

Замечание: Работа со строками, столбцами или ячейками объектов Table, Matrix и List осуществляется как в программе «Microsoft Excel».

Замечание: В ячейках объектов Table, Matrix и List можно печатать текст, как и в ячейках таблиц «Microsoft Excel». Чтобы поместить в ячейку значение поля его можно перетащить

из окна «Data Sources» в ячейку, либо в ячейке написать код, имеющий следующий синтаксис: `= Fields!<Имя поля>.Value` где Имя поля – это имя отображаемого поля. Аналогично можно производить вычисления в ячейках.

Пример: В ячейке отобразить средний бал трёх полей: Оценка1, Оценка2 и Оценка3. для решения этой задачи в ячейке необходимо набрать код:

```
= (Fields!Оценка1.Value+Fields!Оценка2.Value+ Fields!Оценка3.Value)/3
```

На этом мы заканчиваем рассмотрение отчётов. Дополнительную информацию можно найти в лабораторной работе №12.

Литература

1. Кузьменко В.Г. Базы данных в Visual Basic и VBA. Самоучитель. – М.: ООО «Бином-Пресс», 2004 г. – 416с.;
2. Пирогов В.Ю. SQL Server 2005: программирование клиент-серверных приложений. Спб.: БХВ-Петербург, 2006. – 336с.;
3. Стивенс Р. Программирование баз данных. М.: ООО «Бином-Пресс», 2007 г. – 384с.;
4. Уолтерс Роберт, Коулс Майкл, Рей Роберт, Феррачати Фабио, Дональд Фармер SQL Server 2008. Ускоренный курс для профессионалов, Вильямс - Москва – Санкт Петербург – Киев, 2008 – 768с.