Разработка на трислойно клиент-сървър приложение

Разработване на потребителски интерфейс на настолно приложение с *Windows Forms*

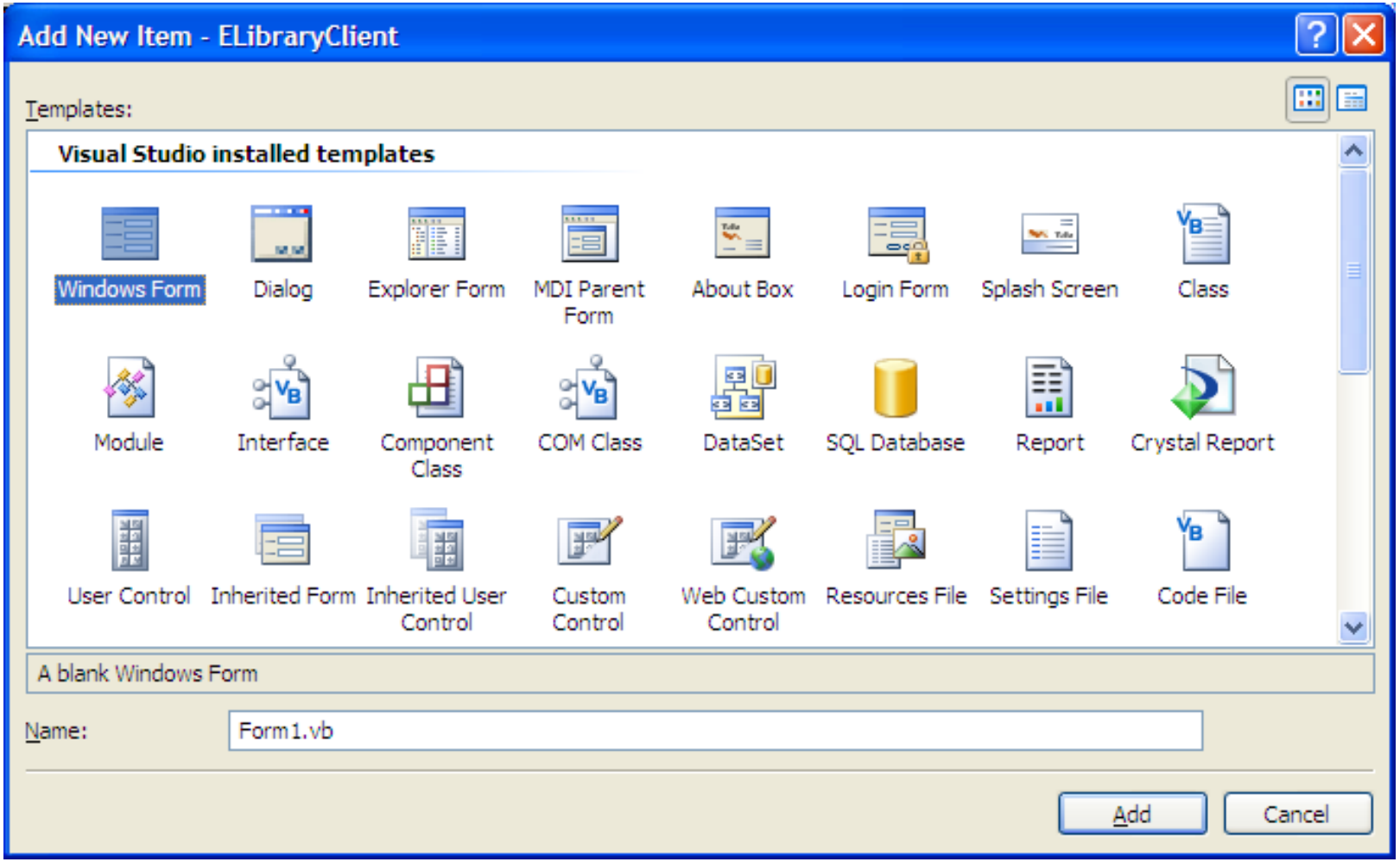
Изграждане на потребителски интерфейс с *Windows Forms*

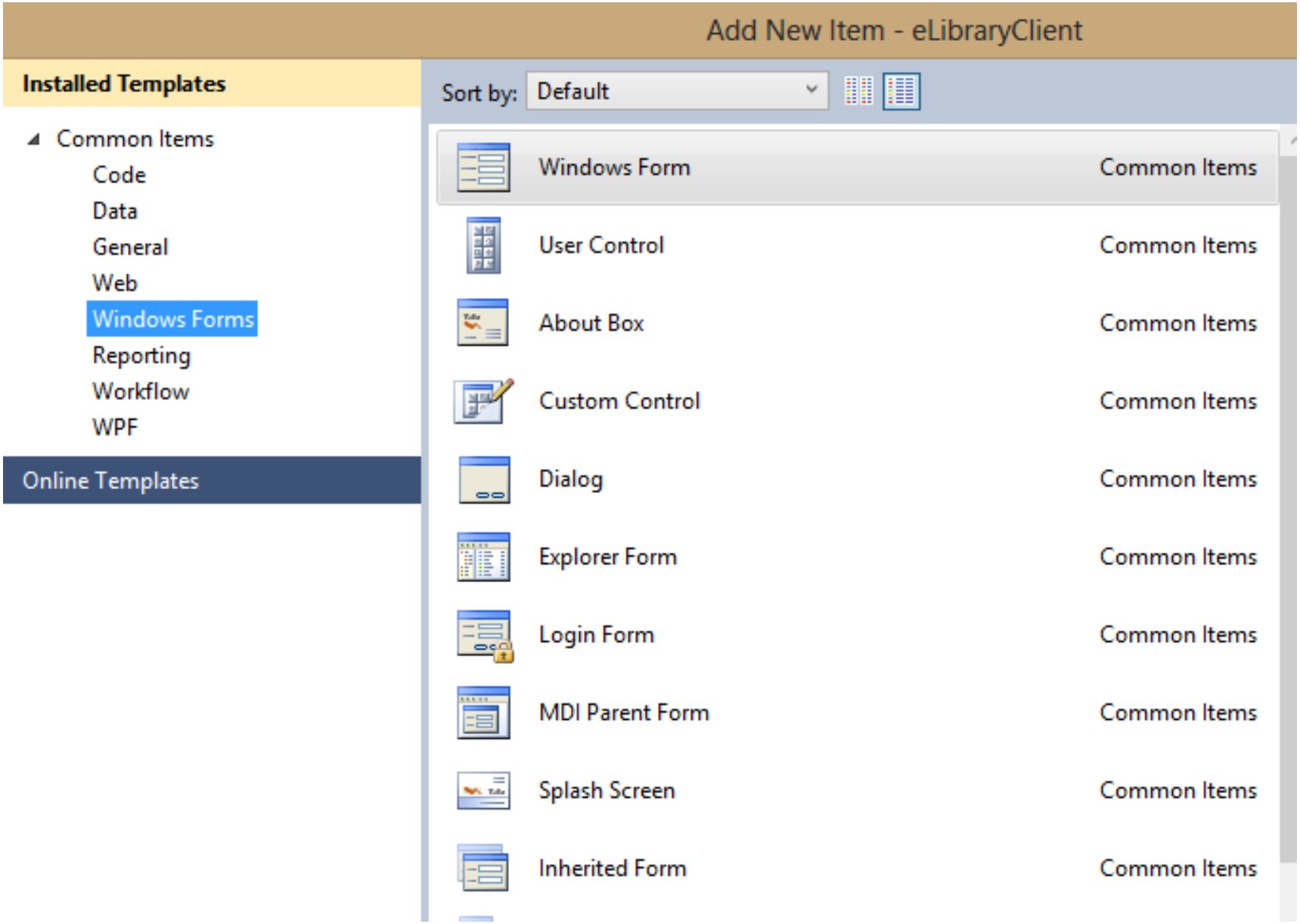
Най-често се използват три основни вида форми:

- Форма от общ вид
- Диалогова форма
- MDI форма (Multi-Document Interface MDI Parent Form)

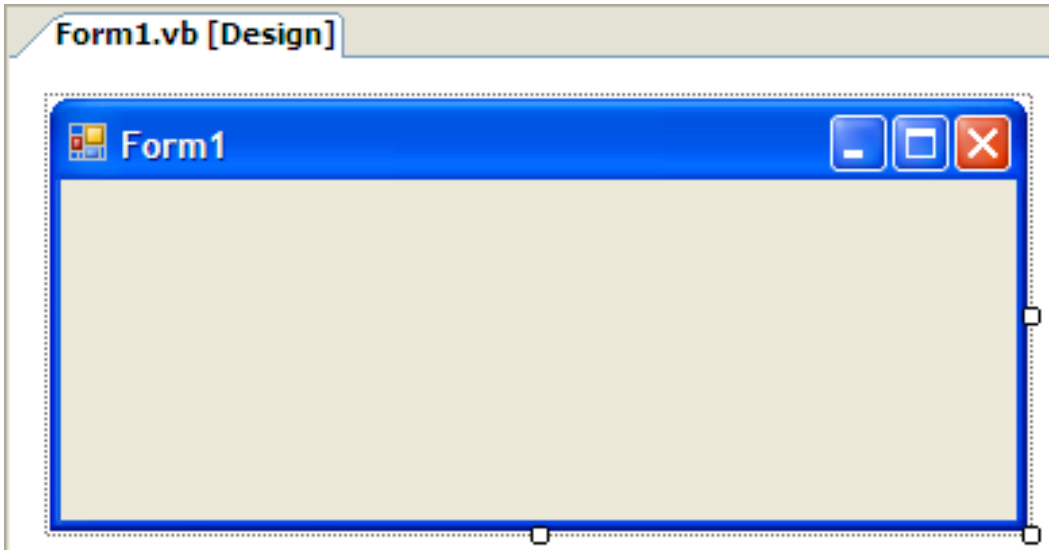
Форма от общ вид

Създава се като от диалоговия прозорец Add New Item избираме ***Windows Form***



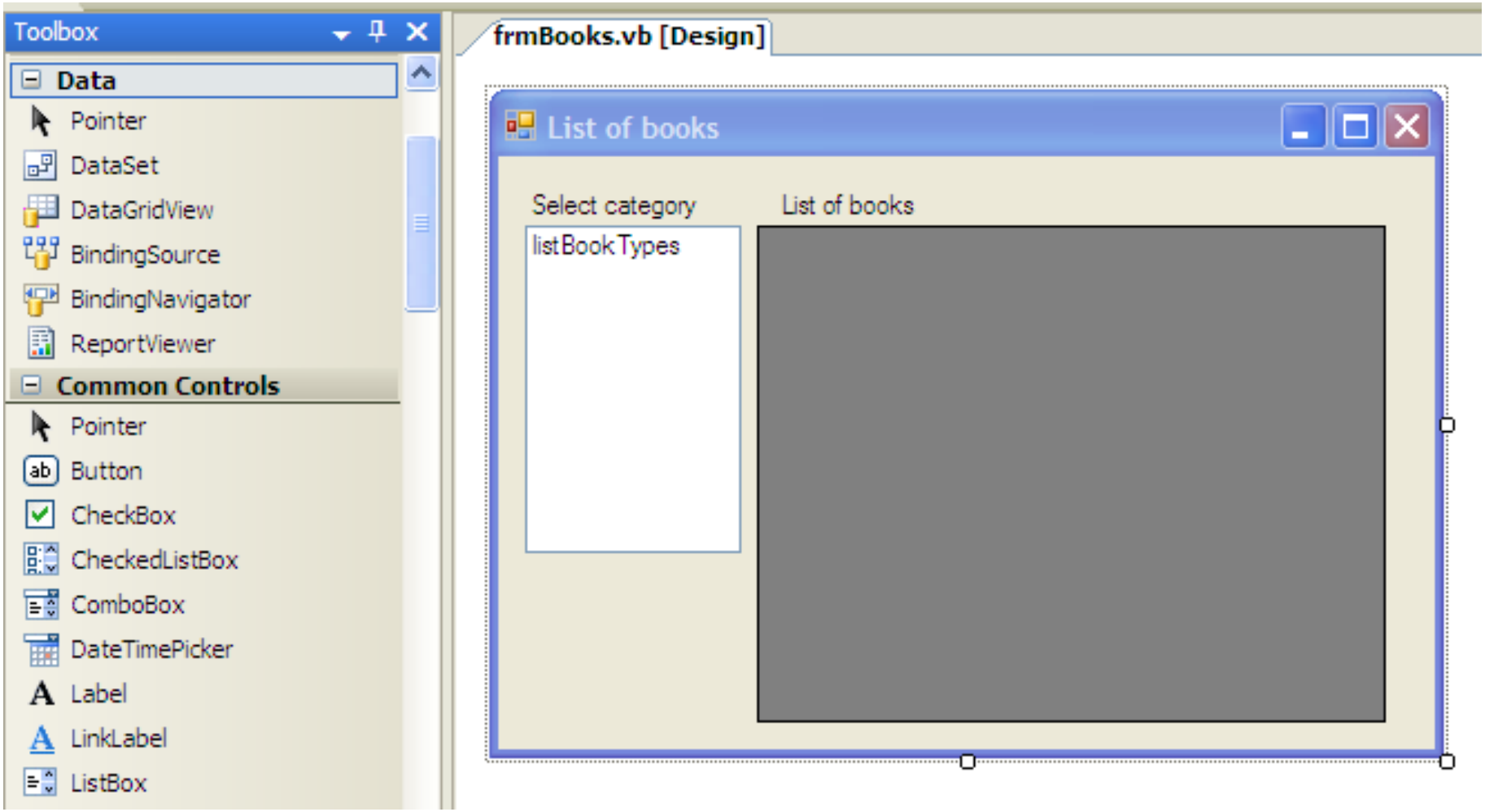

Add New Item - eLibraryClient
Installed Templates
Common Items
Code
Data
General
Web
Windows Forms
Reporting
Workflow
WPF
Online Templates
Sort by:
Default
Windows Form
Common Items
User Control
Common Items
About Box
Common Items
Custom Control
Common Items
Dialog
Common Items
Explorer Form
Common Items
Login Form
Common Items
MDI Parent Form
Common Items
Splash Screen
Common Items
Inherited Form
Common Items

- Формата създадена от темплейта от общ вид има свойство **controlBox=true**, и са активирани трите контрол-бокса в заглавната лента на формата:
  - за минимизиране **MinimizeBox=true**
  - за максимизиране **MaximizeBox=true**
  - за затваряне (той е винаги активиран при *controlBox=true*).



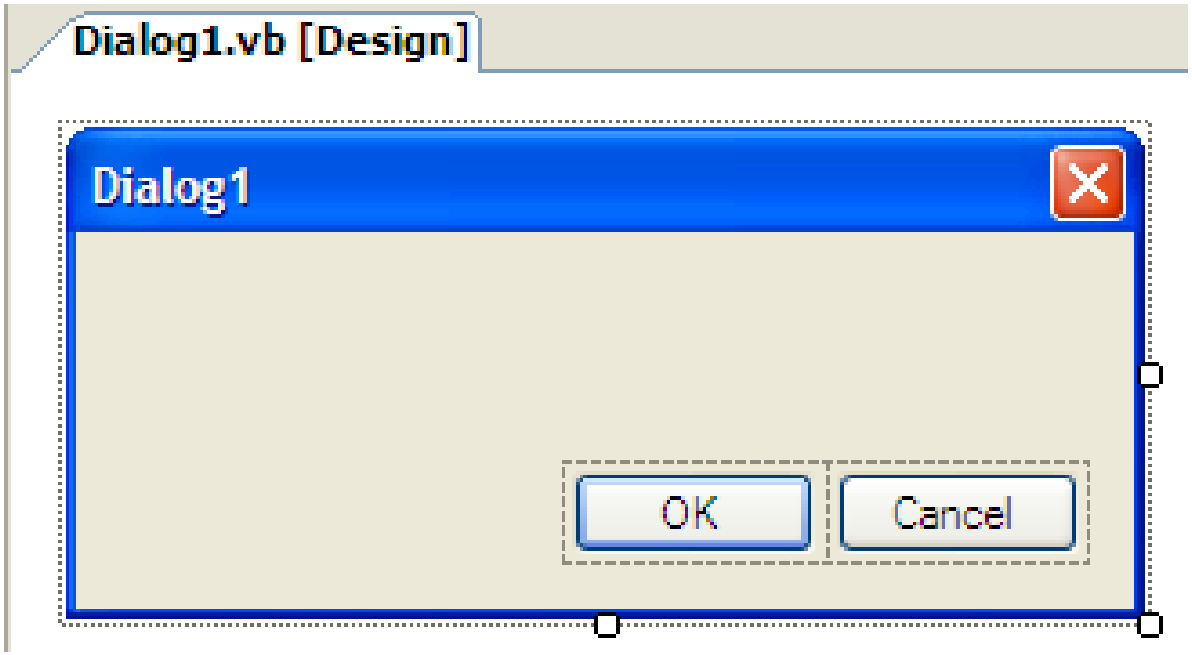
- Компоненти се включват във формата чрез издърпване с мишката (drag-and-drop) от Toolbox.
- В примера ELibrary, формата frmBooks е от основен вид.
- Включени са два етикета (**Label**), един грид (**DataGridView**) и един списъчен контрол (**ListBox**).

  (На следващата фигура се виждат само част от наличните в Toolbox компоненти.)


Toolbox
Data
Pointer
DataSet
DataGridView
BindingSource
BindingNavigator
ReportViewer
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
frmBooks.vb [Design]
List of books
Select category
listBookTypes
List of books

Диалогова форма

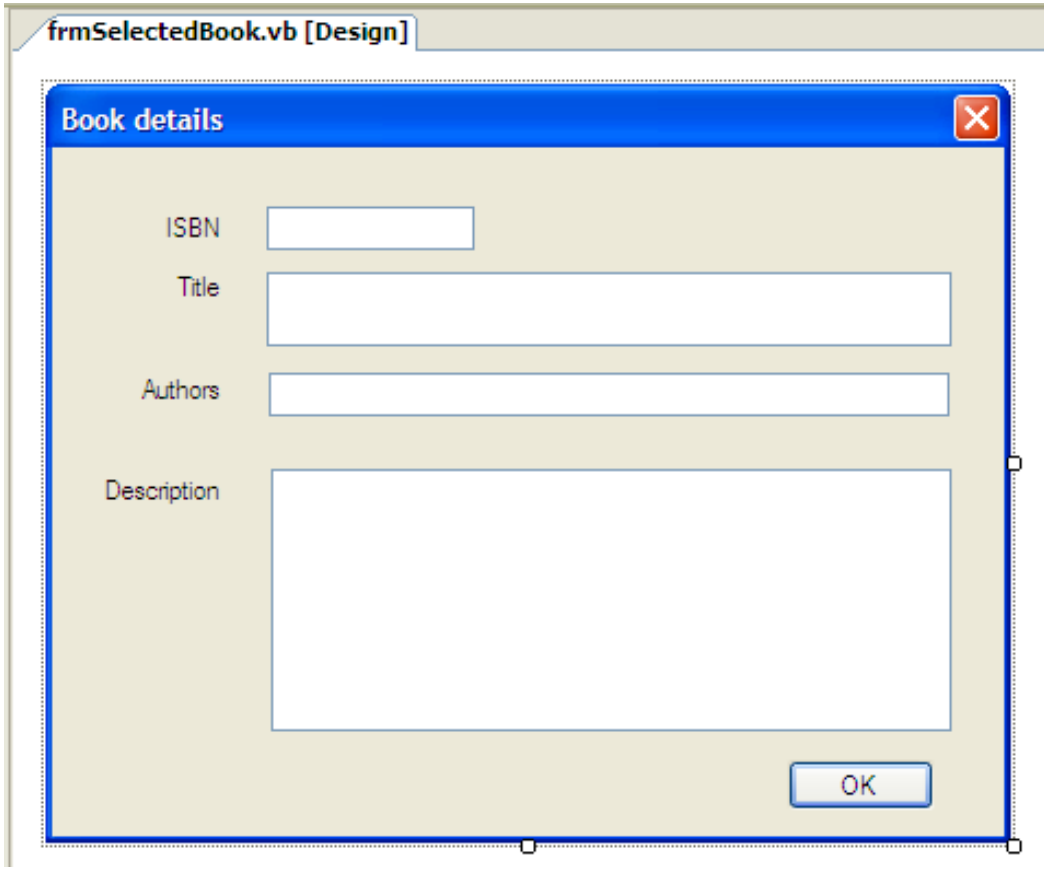
- Форма създадена с темплейт **Dialog** има зададени следните свойства:
- **FormBorderStyle=FixedDialog**
- **controlBox=true** , но е активен само боксът за затваряне, а другите два бокса са дезактивирани MinimizeBox=false ; MaximizeBox= false)
- Включени са два бутона [OK] и [Cancel]



Контрол Button

- Един от най-популярните контроли е Button.
- Използвате се най-често за стартиране на събития включени в кода – например за изпращане на данни попълнени във форма или данни към базата данни и т.н
- Подразбиращото събитие в Button е Click.
- Класът Button е базиран на класа ButtonBase, който пък от своя страна е базира на класа Control.

- В примера ELibrary, формата frmSelectedBook е от диалогов вид. Отстранен е бутонът [Cancel]. Добавени са нови полета.
- Най-често формите от диалогов тип се активират *модално,*
  т.е. до затваряне на формата,
  само тя е активна
  от формите
  на приложението.



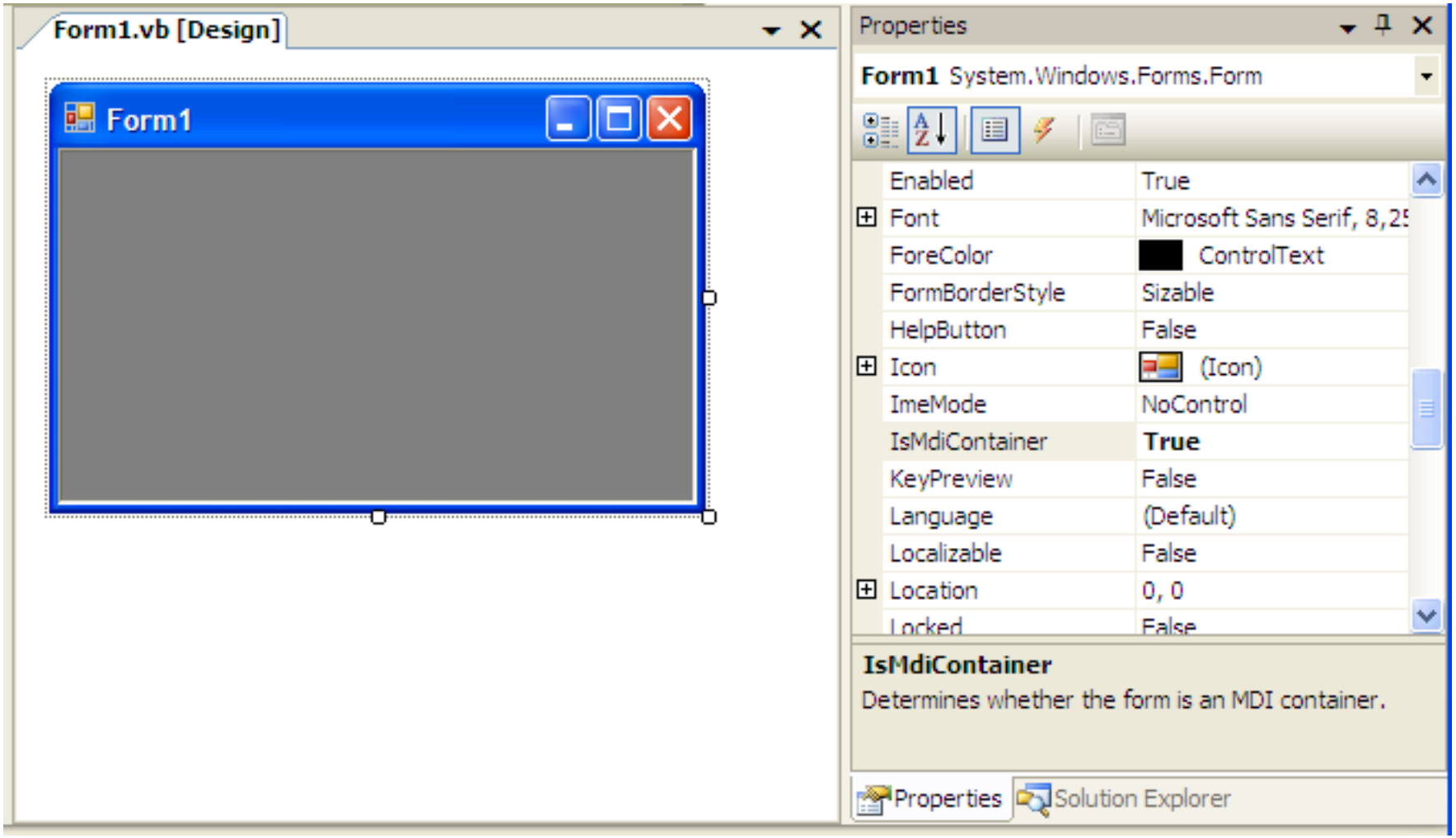
MDI форма (Multi-Document Interface MDI Parent Form)

- Характерното за MDI формата е свойството

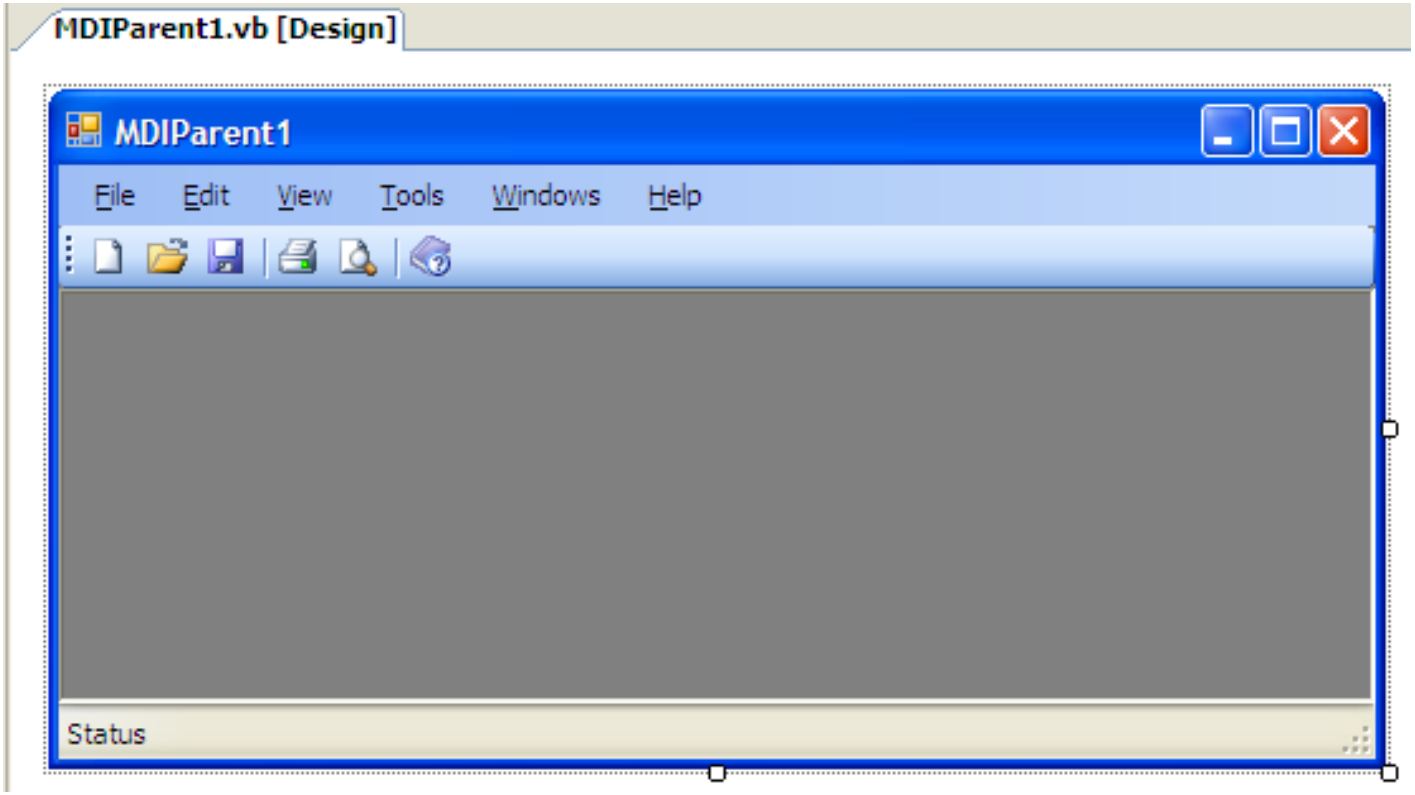
  isMDIContainer=True

- MDI форма може да се създаде от обикновена форма, като се активира горното свойство или пък да се избере специален темплейт **MDI Parent Form.**

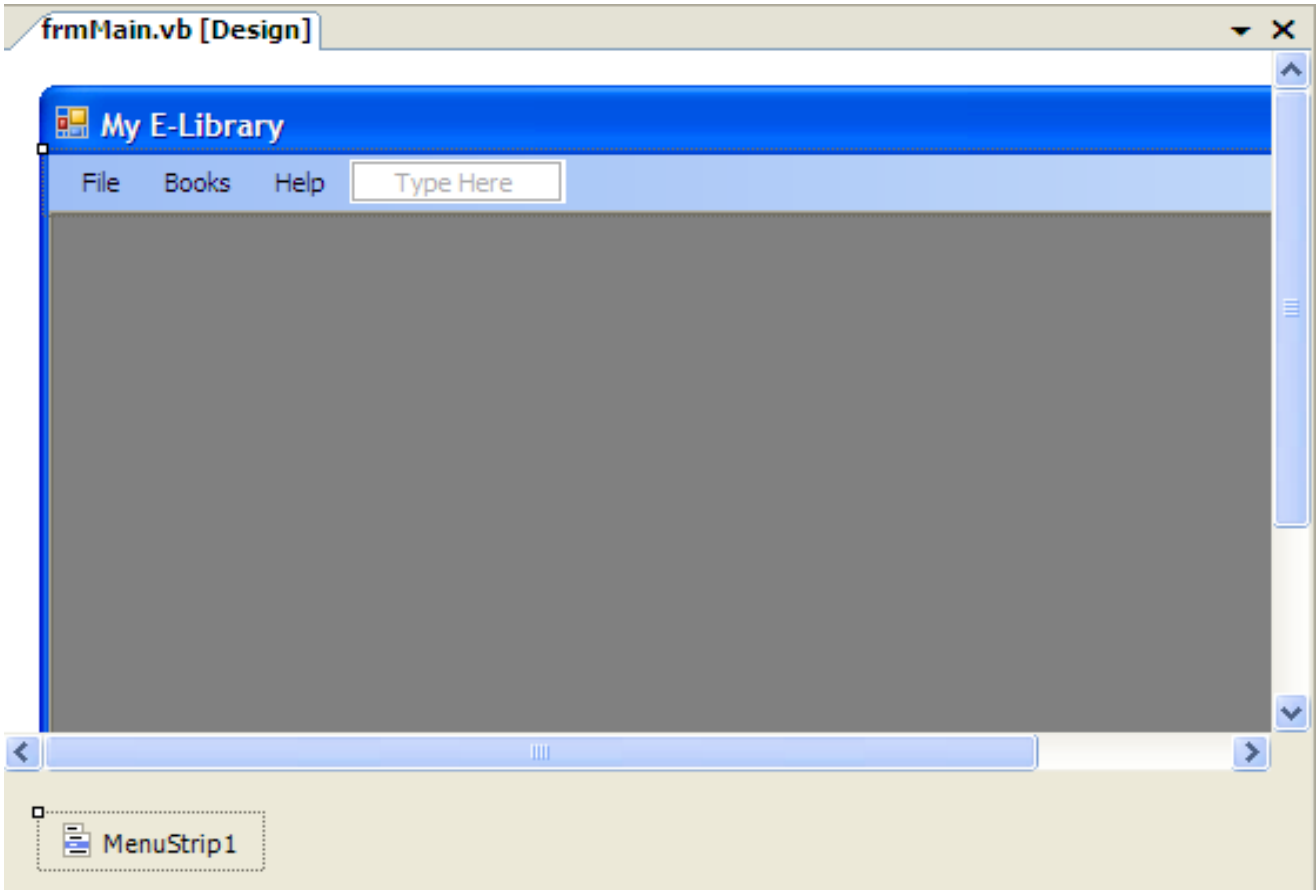
MDI форма


Form1.vb [Design]
Form1
Properties
Form1 System.Windows.Forms.Form
Enabled True
Font Microsoft Sans Serif, 8,25
ForeColor ControlText
FormBorderStyle Sizable
HelpButton False
Icon (Icon)
ImeMode NoControl
IsMdiContainer True
KeyPreview False
Language (Default)
Localizable False
Location 0, 0
Locked False
IsMdiContainer
Determines whether the form is an MDI container.
Properties
Solution Explorer

- MDI форма създадена с темплейта ***MDI Parent Form*** включва ***меню*** и ***инструментална лента*** с набор от стандартни функции. Формата изглежда по следния начин.



- В примера ELibrary, главната форма ***frmMain*** е MDI форма, с меню ***MenuStrip1***
- В MDI форма по принцип могат да се влагат компоненти (етикети, бутони и т.н.), но те ще се скриват от дъщерните форми.



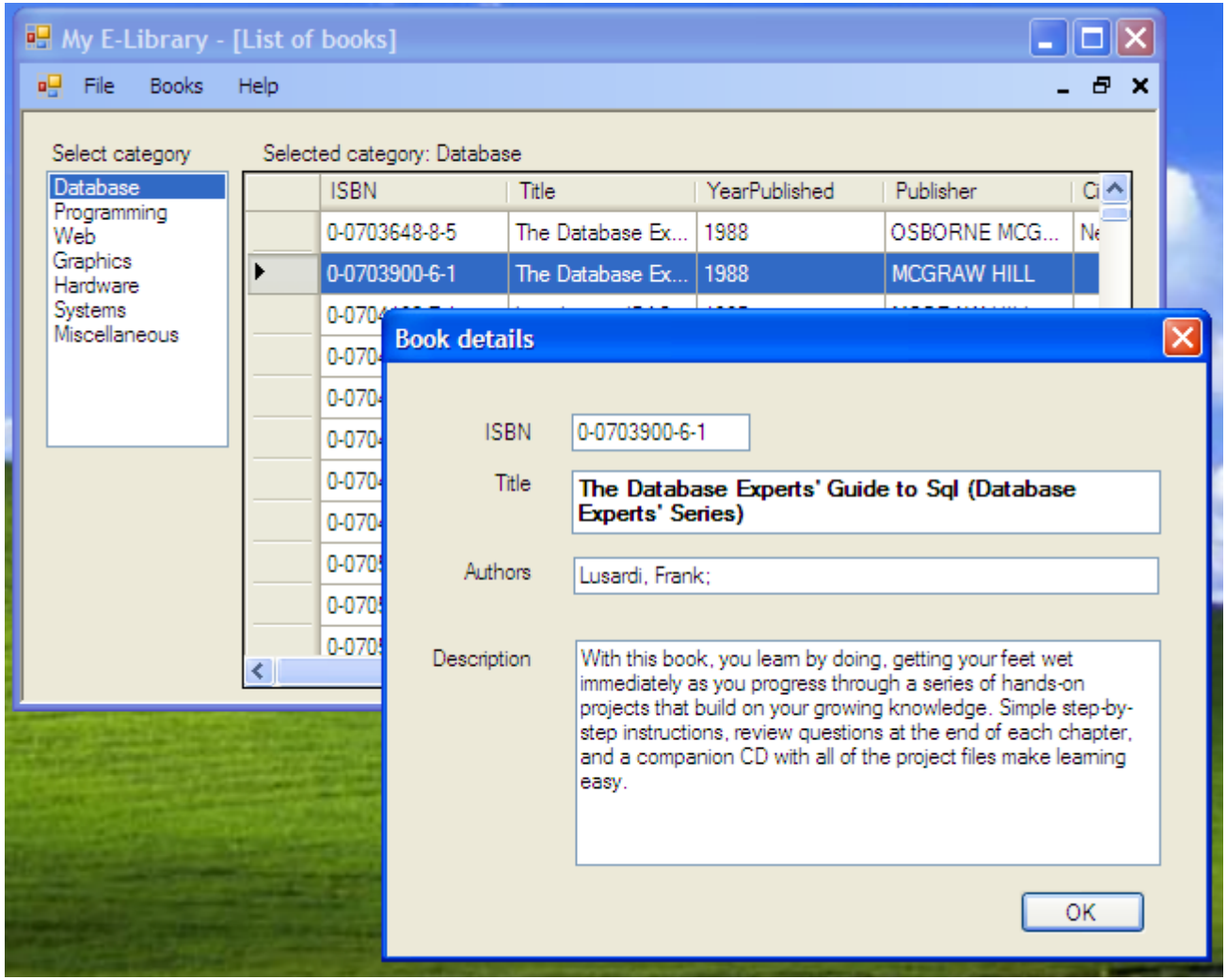
ДЪЩЕРНИ ФОРМИ

- Когато една форма се активира като дъщерна, то тя се отваря само в рамките на родителската MDI форма.

В примера:

- Формата ***frmBooks*** е дъщерна спрямо frmMain и при активиране се показва в нейната рамка.
- Формата ***frmSelectedBook*** не е дъщерна на frmMain. Нейното разположение върху екрана е независимо от frmMain.

Форми frmBooks, frmMain и frmSelectedBook

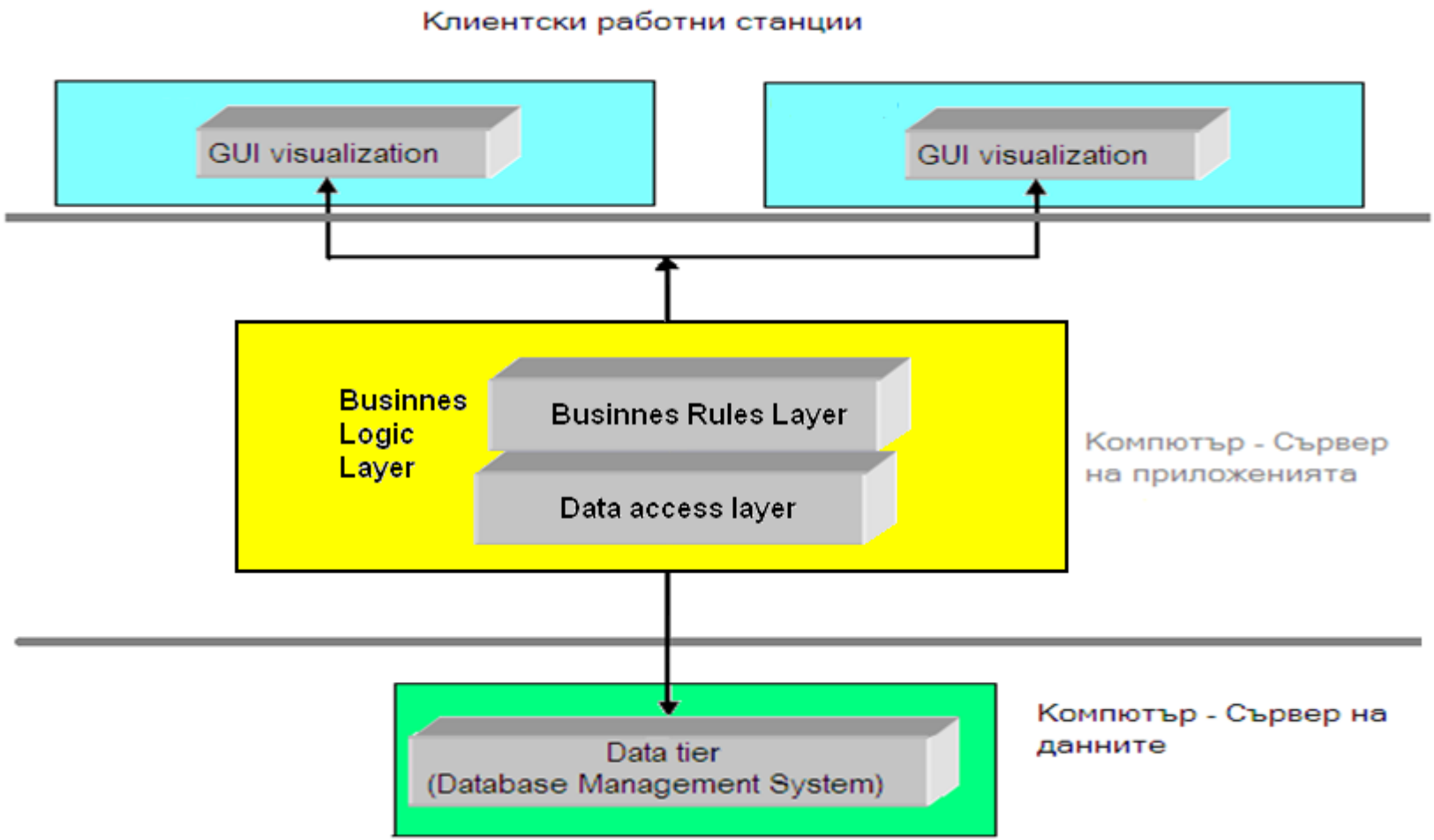


- Стандартните функции, които не се използват в приложението, трябват да се изтрият.
- Забележка: *Дизайнерът не изтрива функциите, при изтриване на съответния компонент от менюто. Функциите трябва да се изтрият ръчно от кода на формата.*

Разработване на приложението на нивото на бизнес логиката

В нивото на бизнес логиката се разграничават две поднива

- Ниво на достъп до данните (Data access layer DAL)
- Ниво на бизнес правилата (Business rules layer BRL)

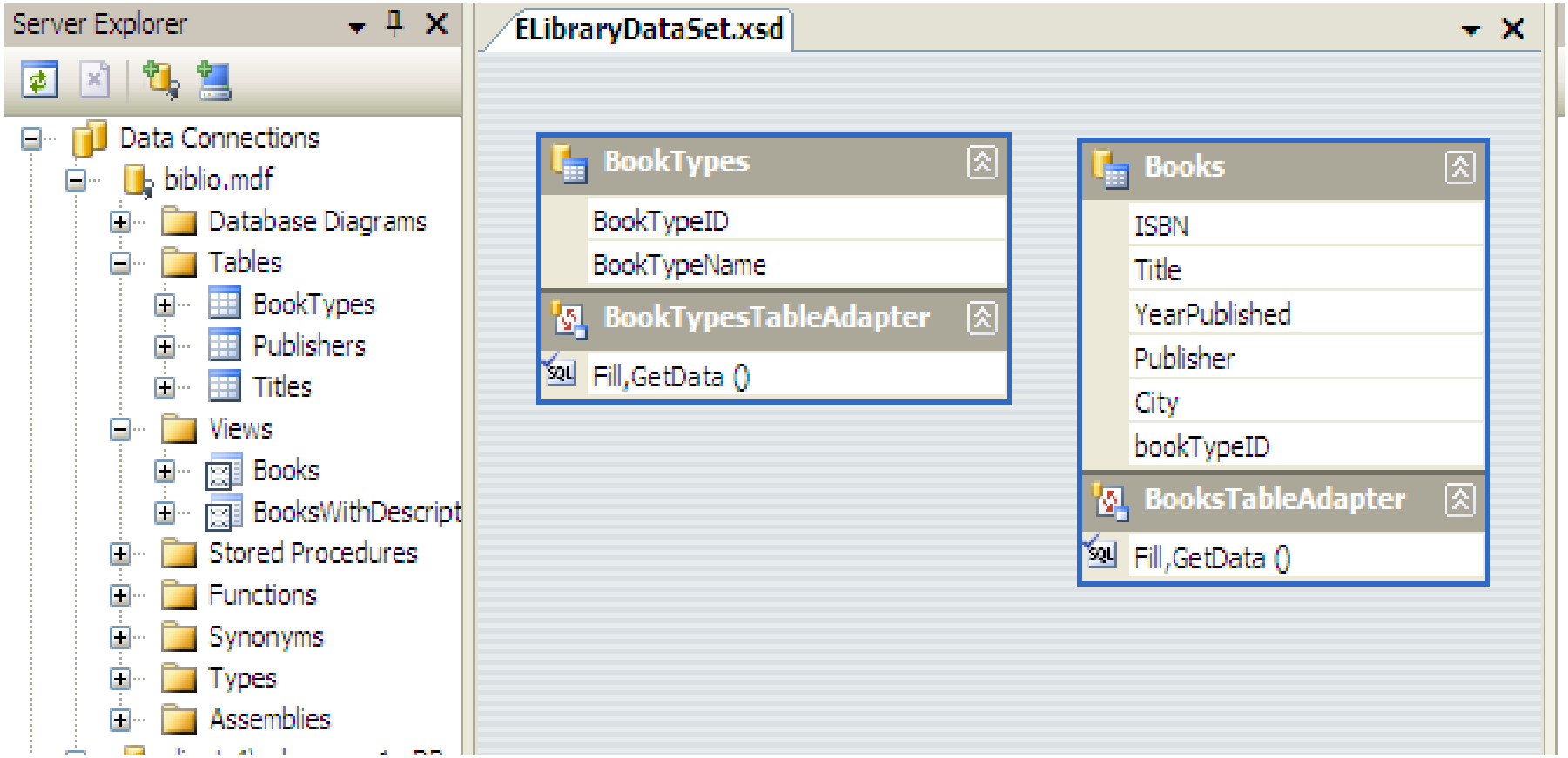

Клиентски работни станции
GUI visualization
GUI visualization
Businnes
Logic
Layer
Businnes Rules Layer
Data access layer
Компютър - Сървер
на приложенията
Data tier
(Database Management System)
Компютър - Сървер на
данните

Подниво на достъп до данните

- Включва обръщенията към нивото на данните (към таблици, заявки, съхранени процедури в базата данни).
- Най-характерния .NET компонент използван на това ниво е **DataSet**.
- Visual studio има графичен дизайнер за разработване на DataSet.
- Таблици и изгледи от SQL база данни, включени в Server Explorer, могат да се включват в DataSet като се издърпват с мишката (drag-and-drop).

В ***ELibraryDataSet*** чрез издърпване от базата данни ***biblio*** са включени
- една таблица BookTypes
- един изглед Books

Дизайнерът автоматично добавя към таблицата компонент ***TableAdapter*** с една команда със стандартно име ***Fill()***, за запълване на таблицата в DataSet с данни от таблицата в БД.



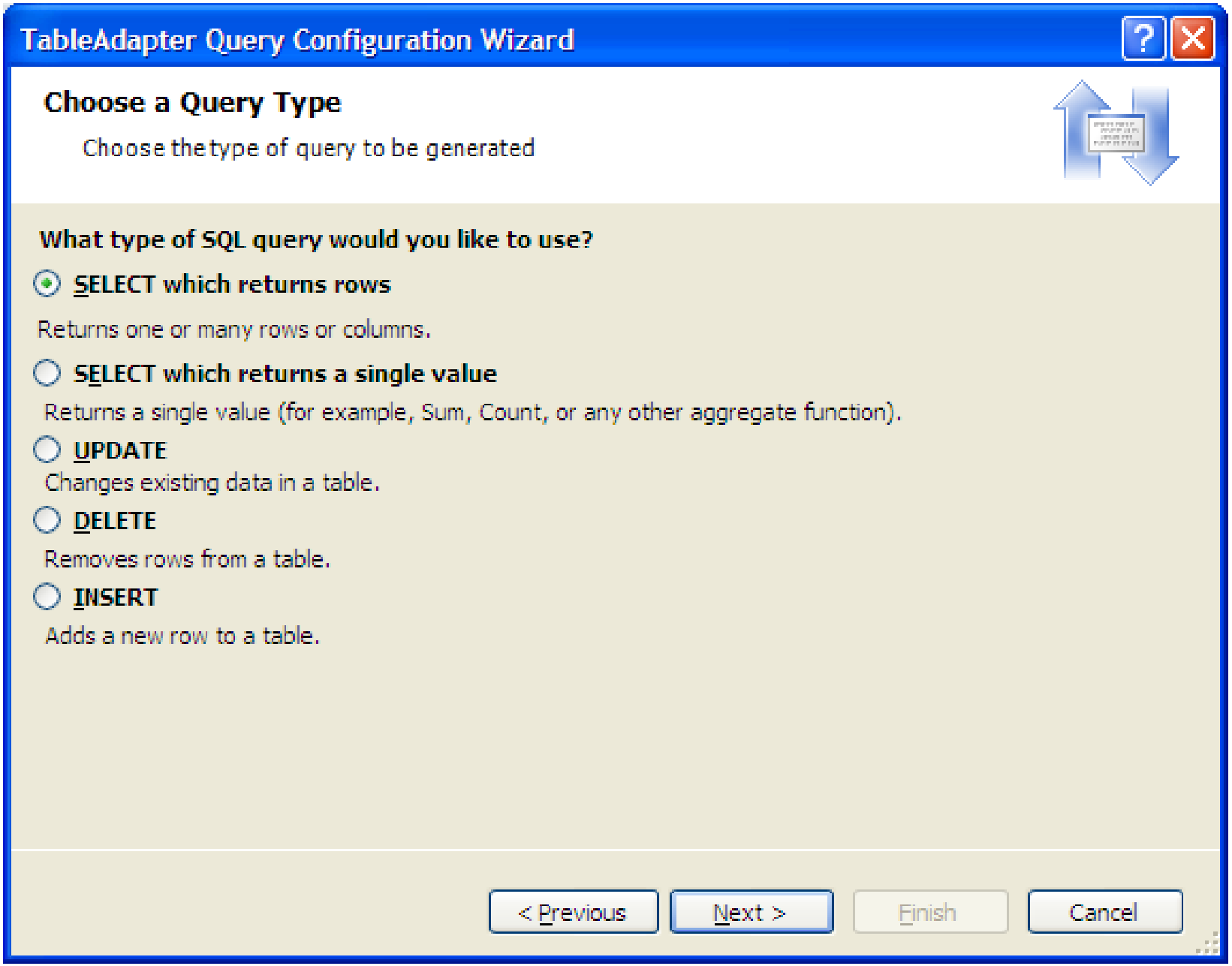
Добавяне на команди към *TableAdapter*

За всяка таблица в DataSet, дизайнерът предлага създаването на 5 типа команди (SQL заявки) които могат да се добавят към ***TableAdapter*** :

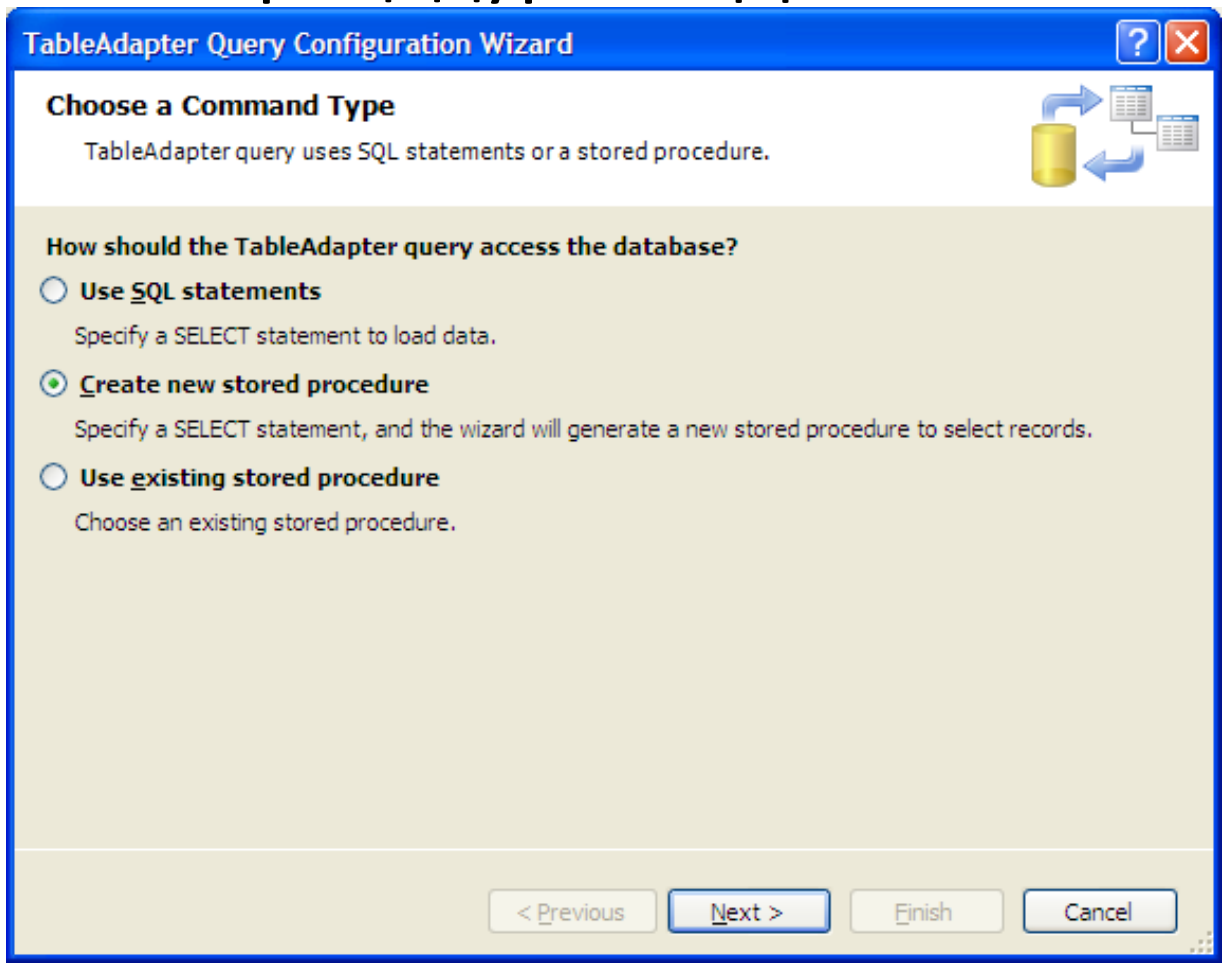
- Команда за запълване на таблицата в DataSet с данни от таблицата в БД (SELECT …)
- Команда връщаща единична стойност – сума или брой записи (SELECT …)

И три типа за отразяване на промените от таблицата в DataSet в таблицата в БД *Промените в таблицата в DataSet не влияят върху данните в съответната таблица от БД докато не се изпълни някоя от дефинираните в DataSet команди*:

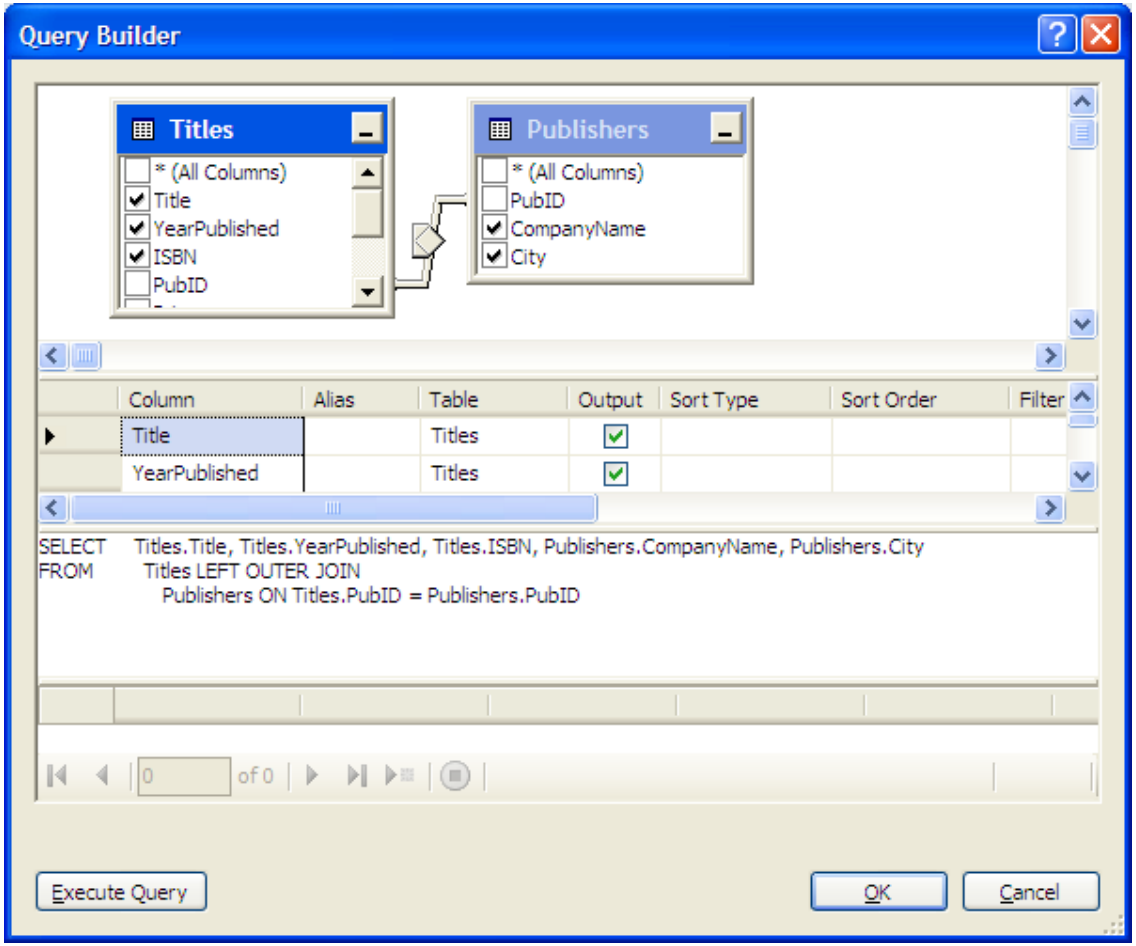
- за актуализация на данните в БД с данни от DataSet (UPDATE …)
- за добавяне на данните в БД (INSERT …)
- за изтриване на данните в БД (DELETE …)


TableAdapter Query Configuration Wizard
Choose a Query Type
Choose the type of query to be generated
What type of SQL query would you like to use?
SELECT which returns rows
Returns one or many rows or columns.
SELECT which returns a single value
Returns a single value (for example, Sum, Count, or any other aggregate function).
UPDATE
Changes existing data in a table.
DELETE
Removes rows from a table.
INSERT
Adds a new row to a table.
< Previous
Next >
Finish
Cancel

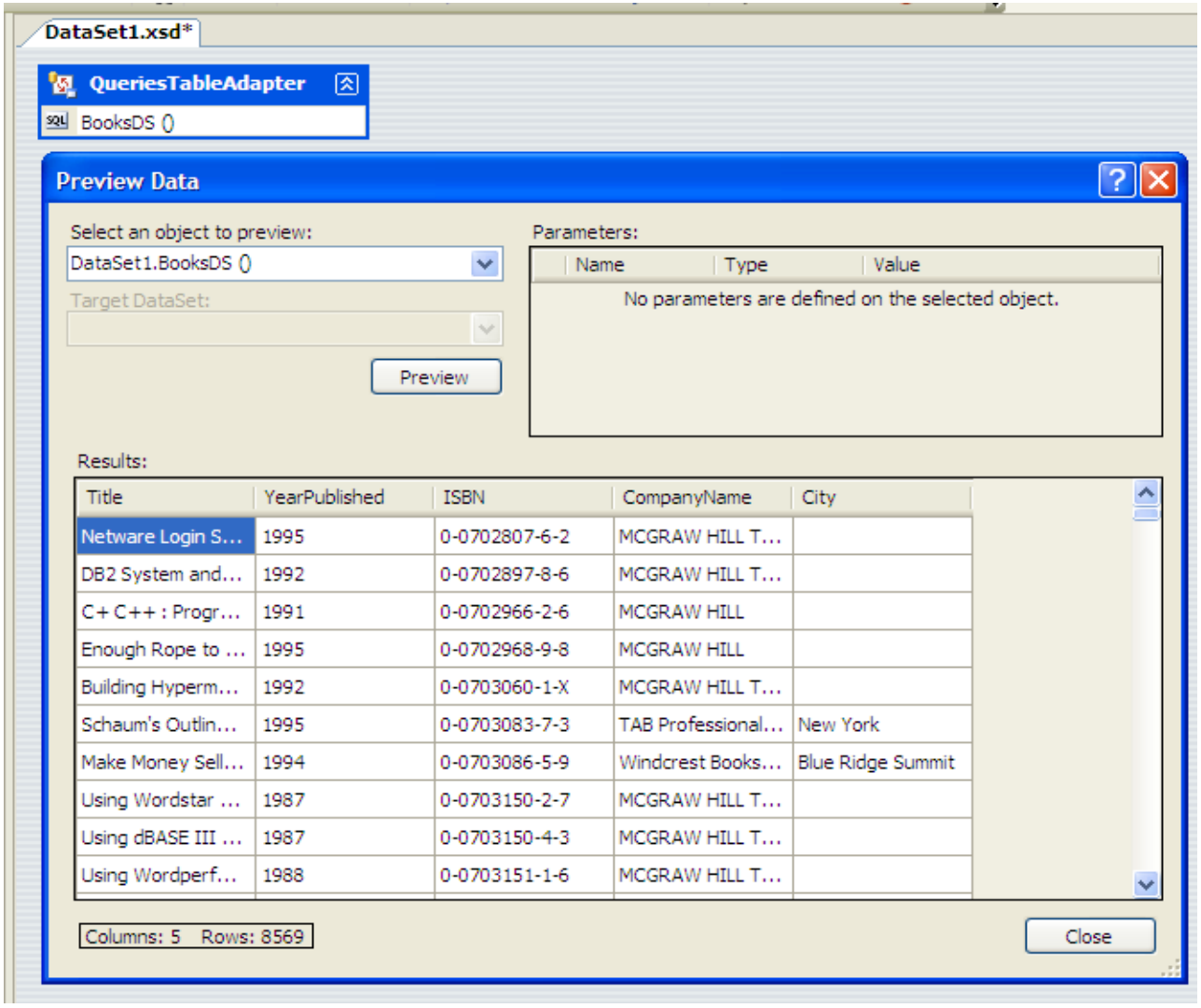
- Разработчикът трябва да избере, дали тези команди да се съхраняват в описанието на DataSet или да бъдат добавени като съхранени процедури в БД.



- В DataSet могат да се дефинират нови изгледи (view) върху таблици от базата данни.
- Например, чрез графичния дизайнер на DataSet можем лесно да дефинираме изглед ***BooksDS***, идентичен с *Books* от базата данни



- В прозореца *Preview Data* ще видим, че изгледа ***BooksDS*** *извежда същите данни,* както изгледа ***Books.***



Извод

- На нивото на достъп на данните мога да се създават SQL заявки (*изгледи в* DataSet и използваните в TableAdapter команди).

Разработващият системата трябва да реши:

- кои да се създават като част от DataSet, като част от кода на приложението,
- кои да се създават и съхраняват в базата данни.

Подниво на бизнес-правилата

- Поднивото на бизнес-правилата включва такива обработки, които не са непосредствено обвързани с структурите на данните в базите данни. Компонентите на това ниво са библиотеки от функции.
- Обособяването на бизнес-правилата в отделни модули, различни от модулите за достъп до данните, се прави когато има съществено преобразуване на данните преди предаването им към потребителското ниво.

От гледна точка на *потребителското ниво*

- Ог гледна точка на потребителското ниво (*клиентските модули)* разликата между двете поднива на бизнес нивото *не е съществена*.

Пример: Клиентска програма извиква от сървера на приложението метод, който дава като резултат средния годишен лихвен процент на една банка за даден вид кредит. Клиентската програма получава в отговор едно число 5.3% . За работата на клиентската програма е напълно безразлично, дали числото 5.3 е взето директно от таблица в БД или е резултат от изчисления на някакъв модул на бизнес-правилата, който е бил активиран при изпълнение на метода.

Разработване на приложение на ниво на данните

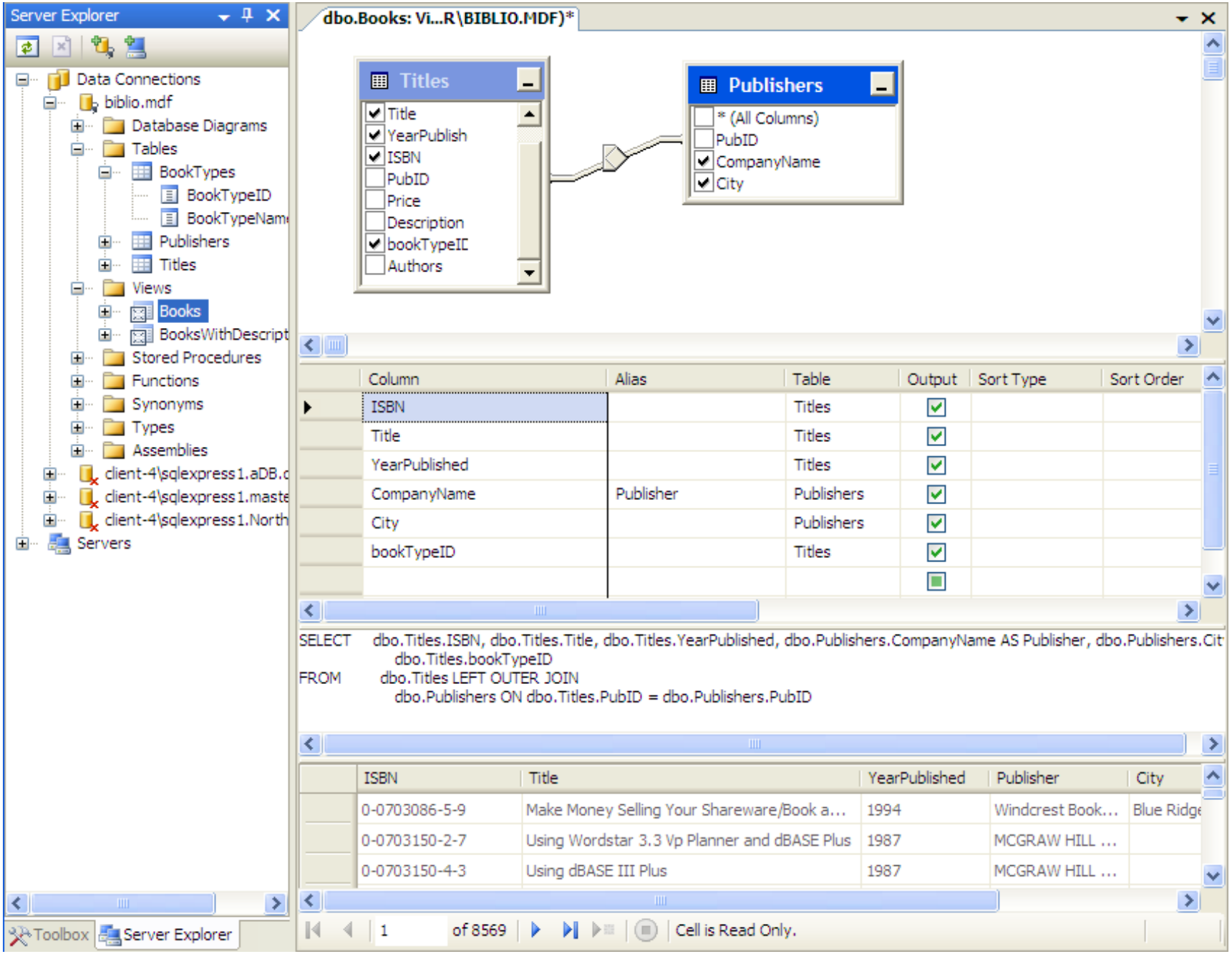
Основни задачи при разработването на приложението на ниво на данните

- Използва се дизайнерът на данни на Visual Studio или SQL Server Manager
- Проектиране на таблиците в базата данни с оптимална логическа схема.
  В някои случаи приложенията се изграждат ползвайки съществуващи бази данни. Тогава задачата се състои в анализ на данните в тези бази данни, с оглед на тяхното използване в новото приложение.
- Изграждане в съответствие с нуждите на приложението на:
  - View (изглед)
  - Съхранени процедури

При създаване на изглед, екранът има четири зони

- графично представяне
- списък на избраните колони и условията към колоните
- Команда на SQL
- Таблица с резултата от изпълнението

(На следващата фигура е показано създаването на view **Books** съдържащо полета от таблиците Titles и Publishers)


Server Explorer
Data Connections
biblio.mdf
Database Diagrams
Tables
BookTypes
BookTypeID
Publishers
Titles
Views
Books
Stored Procedures
Functions
Synonyms
Types
Assemblies
Servers
Toolbox
Server Explorer
dbo.Books: Vi...R\BIBLIO.MDF)*
Titles
Title
YearPublish
ISBN
PubID
Price
Description
bookTypeID
Authors
Publishers
* (All Columns)
PubID
CompanyName
City
Column
Alias
Table
Output
Sort Type
Sort Order
ISBN
Titles
Title
Titles
YearPublished
Titles
CompanyName
Publisher
Publishers
City
Publishers
bookTypeID
Titles
SELECT dbo.Titles.ISBN, dbo.Titles.Title, dbo.Titles.YearPublished, dbo.Publishers.CompanyName AS Publisher, dbo.Publishers.City, dbo.Titles.bookTypeID
FROM dbo.Titles LEFT OUTER JOIN dbo.Publishers ON dbo.Titles.PubID = dbo.Publishers.PubID
ISBN
Title
YearPublished
Publisher
City
0-0703086-5-9
Make Money Selling Your Shareware/Book a...
1994
Windcrest Book...
Blue Ridge
0-0703150-2-7
Using Wordstar 3.3 Vp Planner and dBASE Plus
1987
MCGRAW HILL ...
0-0703150-4-3
Using dBASE III Plus
1987
MCGRAW HILL ...
1
of 8569
Cell is Read Only.