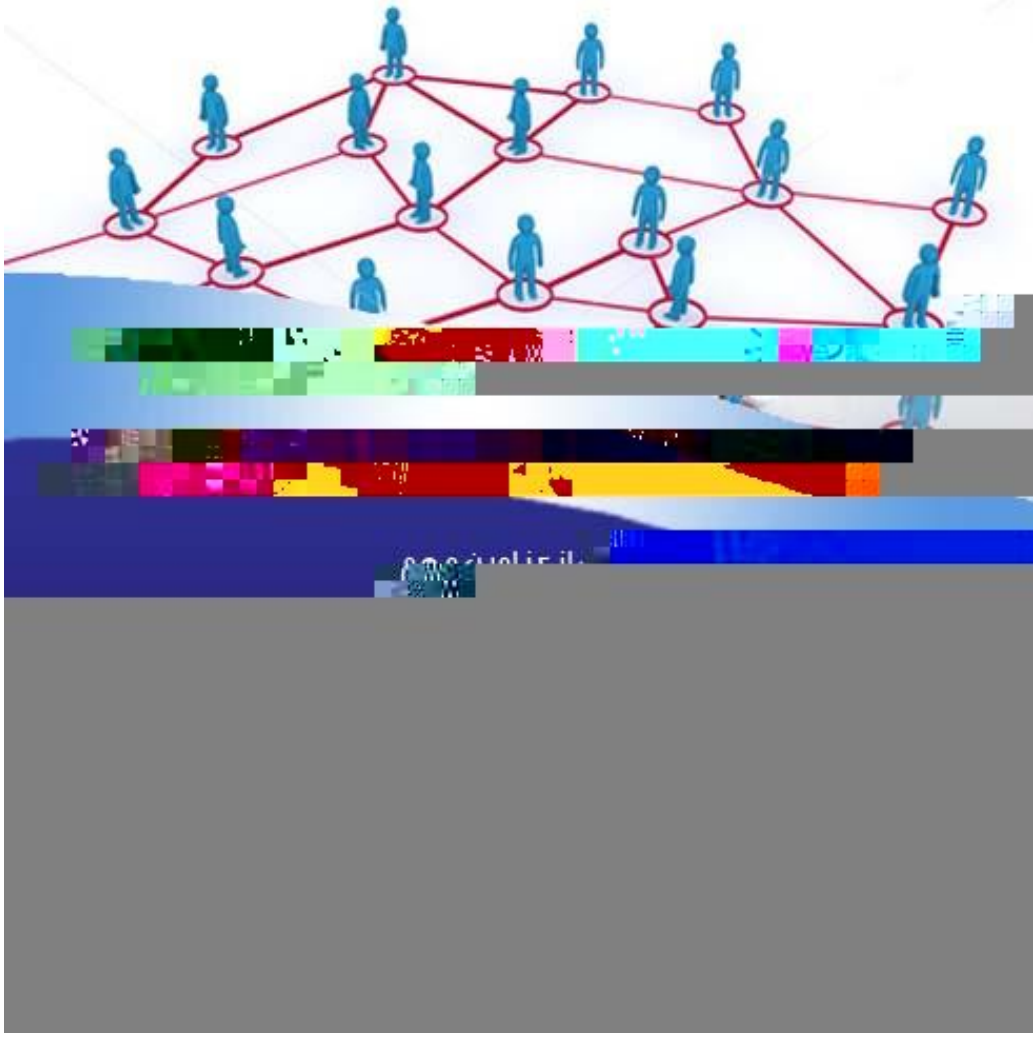
آموزش گام به گام برنامه‌نویسی
بانک اطلاعاتی با ویژوال بیسیک‌نت
(مرجع کامل)

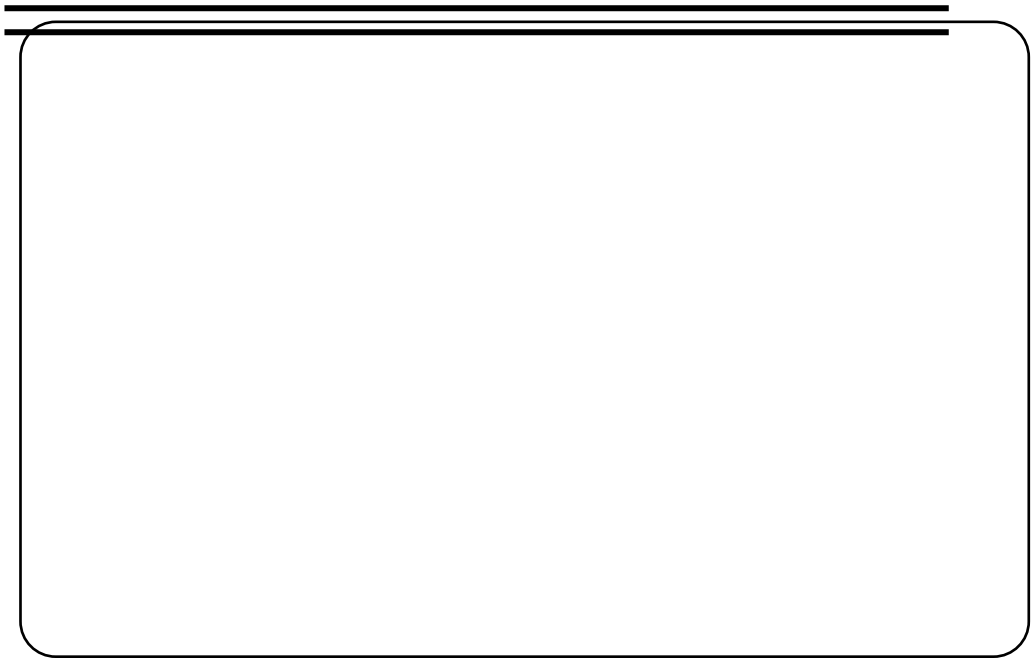


آموزش گام به گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت (مرجع کامل)

تالیف

مهندس رمضان عباس‌نژادورزی

فن‌آوری نوین



سرشناسه : عباس نژاد، رمضان، ۱۳٤۸ –
عنوان و نام پدیدآور : آموزش گام به گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت(مرجع کامل)/تالیف رمضان عباس‌نژادورزی.
مشخصات نشر : بابل: فناوری نوین، ۱۳۸۸.
مشخصات ظاهری : ۲۸۸ص.:مصور،جدول.
شابک : ٦٦۰۰۰ریال:۹۷۸٦۰۰۹۱٤۱۳٥۷
وضعیت فهرست نویسی : فیپا
موضوع : ویژوال بیسیک مایکروسافت
موضوع : ویژوال بیسیک (زبان برنامه‌نویسی کامپیوتر)
موضوع : مایکروسافت دات نت
موضوع : پایگاه‌های اطلاعاتی – مدیریت
رده بندی کنگره : QA۷٦/۷۳/ب۹ع۱٦ ۱۳۸۸
رده بندی دیویی : ۰۰٥/۲٦۸
شماره کتابشناسی ملی : ۸٤۷۷٥۹۱

فن‌اوری نوین
www.fanavarinovin.net
بابل، کدپستی ٤۷۱٦۷-۷۳٤٤۸ تلفن: ۲۲٥٦٦۸۷-۰۱۱۱

آموزش گام به گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت (مرجع کامل)

تألیف: مهندس رمضان عباس‌نژادورزی

ناشر: فن‌آوری نوین

چاپ اول: زمستان ۱۳۸۸

تیراژ :۱۰۰۰ جلد

طراح جلد: احمد فرجی

شابک: ۷- ٥- ۹۱٤۱۳ - ٦۰۰ - ۹۷۸

حروفچینی و صفحه‌آرایی: فن‌آوری نوین **قیمت: ٦٦۰۰ تومان**

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱٤۲ **تلفکس: ٦٦٤۰۰۲۲۰-٦٦٤۰۰۱٤٤**

فهرست مطالب

مقدمه

امروزه حجم زیادی از اطلاعات ذخیره و بازیابی می‌شوند. برای جلوگیری از افزونگی داده (تکرار بی‌مورد داده‌ها)، بی نظمی و ایجاد سازگاری بین گزارش‌ها از بانک اطلاعات استفاده می‌شود. بانک‌های اطلاعات متعددی وجود دارند که پرکاربردترین و بهترین آنها، سیستم بانک اطلاعات رابطه‌ای است. یکی از بانک‌های اطلاعات رابطه‌ای، **SQL Server** است اکنون نسخه ۲۰۰۸ این بانک اطلاعاتی در بازار موجود است. اما، این کتاب طوری طراحی شده است که وابسته به نسخه خاصی از بانک اطلاعاتی نباشد. زیرا در این کتاب سعی شده است تمام کارها از طریق دستورات **SQL** انجام شود. ویژگی‌هایی از قبیل کارایی بالا، سهولت یادگیری و استفاده، کارکردن در محیط شبکه، قابلیت دسترسی و امنیت بالا، آن را به عنوان پرکاربردترین بانک اطلاعات جهان تبدیل کرده است. به طوری که ۷۰ درصد کاربران دنیا از این بانک اطلاعات استفاده می‌کنند.

از طرف دیگر، با بانک اطلاعات **SQLServer** نمی‌توان کارهایی از قبیل ایجاد فرم‌های ورود و ویرایش اطلاعات، تهیه گزارش‌ها و غیره را انجام داد. به همین دلیل، برای انجام کارهای مذکور به زبان برنامه‌نویسی نیاز داریم. از آنجایی زبان برنامه‌نویسی ویژوال بیسیک‌دات‌نت، یک زبان ساده و کار آمد است که افراد زیادی از آن بهره می‌گیرند. از نکات بارز این کتاب آموزش گزارش‌گیری **Microsoft Report** است که در فصل ۹ آن را می‌بینید. این زبان را به عنوان زبان برنامه‌نویسی بانک اطلاعات انتخاب نمودیم.

در این کتاب مطالبی از قبیل ایجاد بانک اطلاعات، جداول، کلاس‌های ویژوال بیسیک‌نت برای کار با بانک اطلاعات، ورود، ویرایش، حذف رکوردها، رویه‌های ذخیره شده، پشتیبان‌گیری و بازیابی فایل‌های پشتیبان به صورت فشرده شده، تراکنش‌ها و گزارش‌گیری از بانک اطلاعات بیان گردید.

کتاب حاضر با بهره‌گیری از سال‌ها تجربه در امر تدریس، تألیف کتب کامپیوتر و مهم‌تر از همه برنامه‌نویسی در زمینه بانک اطلاعات تدوین شده است. از ویژگی‌های جالب و برجسته این کتاب، بیان مثال‌های متنوع کاربردی، حل گام به گام آنها و توضیح کامل مثال‌های بیان شده، می‌باشد.

در پایان امیدوارم این اثر مورد توجه اساتید و دانشجویان عزیز واقع شود.

کتاب حاوی برنامه‌هایی است که کد آنها را می‌توانید به صورت رایگان از سایت انتشارات فن‌آوری نوین به آدرس **www.fanavarinovin.net** بگیرید.

رمضان عباس نژادورزی
fanavarienovin@gmail.com

نام کتاب	لینک خرید چاپی	لینک خرید الکترونیکی	لینک فایل نمونه
مبانی رایانه و برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_380238.html	http://ktbr.ir/b30588	http://ketabesabz.com/dl/52319
آشنایی با مبانی امنیت شبکه (امنیت اطلاعات)	http://daneshnegar.com/book_371137.html	http://ktbr.ir/b30327	
اصول طراحی پایگاه داده‌ها	http://daneshnegar.com/book_371655.html	http://ktbr.ir/b29943	http://ketabesabz.com/dl/52155
آموزش گام‌به‌گام برنامه‌نویسی پایتون	http://daneshnegar.com/book_392582.html	http://ktbr.ir/b29984	http://ketabesabz.com/dl/52181
آزمایشگاه C++ (حل مسائل کامپیوتر مرجع کامل)	http://daneshnegar.com/book_392262.html	http://ktbr.ir/b29982	
آموزش گام‌به‌گام C # با LINQ	http://daneshnegar.com/book_369388.html	http://ktbr.ir/b28451	
ساختمان داده‌ها با C++	http://daneshnegar.com/book_379094.html	http://ktbr.ir/b29676	http://ketabesabz.com/dl/52156
طراحی سیستم‌های شی‌گرا با C# زبان	http://daneshnegar.com/book_374658.html	http://ktbr.ir/b29621	http://ketabesabz.com/dl/52126
مدیریت استراتژیک فناوری اطلاعات	http://daneshnegar.com/book_374659.html	http://ktbr.ir/b29779	http://ketabesabz.com/dl/52125
گرافیک رایانه‌ای با زبان C# برنامه‌نویسی	http://daneshnegar.com/book_376021.html	http://ktbr.ir/b29674	http://ketabesabz.com/dl/51049
درس و کنکور پایگاه داده پیشرفته	http://daneshnegar.com/book_392578.html	http://ktbr.ir/b29644	http://ketabesabz.com/dl/52102
فیزیک الکتریسیته	http://daneshnegar.com/book_379161.html	http://ktbr.ir/b29680	
تجارت الکترونیکی	http://daneshnegar.com/book_379188.html	http://ktbr.ir/b29619	
راهنمای کاربردی کاربری برای شبکه‌های OPNET شبیه‌سازی کامپیوتر	http://daneshnegar.com/book_392583.html	http://ktbr.ir/b28504	
درس و کنکور سیستم عامل پیشرفته	http://daneshnegar.com/book_392580.html	http://ktbr.ir/b28505	http://ketabesabz.com/dl/52180
شبکه‌های کامپیوتری با رویکرد کاربردی، آزمایشگاه شبیه‌سازی	http://daneshnegar.com/book_392254.html	http://ktbr.ir/b28528	

شبکه			
آزمایشگاه پایگاه دادهSQL با Server 2012	http://daneshnegar.com/book_377301.html	http://ktbr.ir/b28503	
کاربرد رایانه در مدیریت و حسابداری	http://daneshnegar.com/book_375892.html	http://ktbr.ir/b28450	
آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت	http://daneshnegar.com/book_368929.html	http://ktbr.ir/b28449	
آموزش گام‌به‌گام برنامه‌نویسی به زبان C++	http://daneshnegar.com/book_380238.html	http://ktbr.ir/b28452	
دانلود کتاب آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با C#	http://daneshnegar.com/book_368486.html	http://ktbr.ir/b28448	
حل مسائل پاسکال		http://ktbr.ir/b28398	
حل مسائل C++	http://daneshnegar.com/book_392262.html	http://ktbr.ir/b28401	http://ketabesabz.com/dl/51048
دانلود کتاب حل مسائل C#	http://daneshnegar.com/book_374657.html	http://ktbr.ir/b28399	http://ketabesabz.com/dl/51011
دانلود کتاب حل مسائل C		http://ktbr.ir/b28397	

فصل ۱ آشنایی با زبان ویژوال بیسیک‌نت

سازمان‌ها برای نگهداری داده‌ها از بانک اطلاعاتی استفاده می‌کنند. یکی از پرکاربردترین نرم‌افزارهای مدیریت بانک‌های اطلاعات، SQL Server است. از طرف دیگر، بانک اطلاعات به تنهایی نمی‌تواند نیازهای سازمان‌ها را برطرف کند. به همین دلیل با یکی از زبان‌های برنامه‌سازی باید بتوان به بانک اطلاعات متصل شده، داده‌های آن را ویرایش، حذف و اضافه نمود. امروزه صدها زبان برنامه‌سازی وجود دارند که از طریق آنها می‌توان به بانک اطلاعات متصل گردید و داده‌های آن را دستکاری نمود. یکی از مهم‌ترین و پرکاربردترین زبان‌های برنامه‌سازی، ویژوال بیسیک نت می‌باشد. به همین دلیل در این فصل به طور خلاصه زبان ویژوال بیسیک‌نت را می‌آموزیم.

۱-۱. زبان ویژوال بیسیک نت

این زبان به همراه نرم‌افزار ویژوال استودیونت ارائه شده است. زبان ویژوال بیسیک‌نت از فناوری شیء[1] و مفهوم شی‌گرایی[2] استفاده می‌کند. هر چیزی که در دنیای واقعی وجود دارد، **شیء** نامیده می‌شود. مثل مردم، ساختمان‌ها، کارخانه‌ها، گیاهان، اتومبیل‌ها، کامپیوترها و غیره. هر شیء از **صفاتی** مانند اندازه، رنگ، وزن و دیگر صفات تشکیل شده است که شکل ظاهری آن را تعیین می‌کنند و رفتارهایی از خودشان نشان می‌دهند، مانند انسان می‌خوابد، گریه می‌کند، می‌خندد، اتومبیل حرکت می‌کند، ترمز می‌نماید و غیره.

از آنجایی که زبان ویژوال بیسیک نت از فناوری شی‌گرایی استفاده می‌نماید، امکاناتی در این زبان اضافه شده است که می‌توانید اشیای دنیای واقعی را مدل‌سازی کنید. برای این که ویژوال بیسیک نت شی‌گرایی را پیاده‌سازی کند از مفهوم **کلاس**[3] استفاده می‌کند. **کلاس**، برای ایجاد انواع جدید به کار می‌رود. هر کلاس شامل داده‌ها و متدهایی است که داده‌ها را دستکاری می‌کنند و سرویس‌هایی را برای مشتریان فراهم می‌نمایند. با مفهوم کلاس و چگونگی پیاده‌سازی آنها در ادامه بیشتر آشنا خواهیم شد.

---

[1]- Object [2]- Object Oriented [3]- Class

۲-۱. فضای نام

همان‌طور که بیان گردید، زبان برنامه‌نویسی ویژوال بیسیک نت از کلاس برای برنامه‌نویسی استفاده می‌کند. کلاس‌ها به دو دسته تقسیم می‌شوند که عبارتند از:

۱. **کلاس‌های آماده**: این کلاس‌ها از قبل نوشته شده‌اند و در کتابخانه FCL ویژوال استودیونت وجود دارند.

۲. **کلاس‌هایی که برنامه‌نویس می‌نویسد**: همه کلاس‌های مورد نیاز برنامه‌نویسان از قبل وجود ندارند. بنابراین برنامه‌نویس نیاز دارد، برخی از کلاس‌ها را بنویسد. در ادامه با این کلاس‌ها آشنا خواهید شد.

کلاس‌هایی که در کتابخانه FCL وجود دارند، در **فضاهای نام**[4] مختلف قرار می‌گیرند. برخی از فضای نام‌ها و وظایف آنها در جدول ۱-۱ آمده است. این فضاهای نام به طور خودکار به برنامه ویژوال بیسیک نت اضافه می‌شوند. علاوه بر این فضاهای نام، فضاهای نام دیگری وجود دارند که می‌توانید آنها را به برنامه اضافه کنید. برای این منظور باید از دستور Imports استفاده نمایید. به عنوان مثال، دستورات زیر را ببینید:

```
Imports System.Data.SqlClient
Imports System.Convert
```

دستور اول، فضای نام System.Data.SqlClient را به برنامه اضافه می‌کند تا بتوانید از کلاس‌هایی که برای اتصال و دستکاری به بانک اطلاعات SQL Server به کارمی‌روند، بهره بگیرید (با این فضای نام در ادامه بیشتر آشنا خواهید شد) و دستور دوم، فضای نام System.Convert را به برنامه اضافه می‌کند تا بتوانید از متدهایی که برای تبدیل انواع داده‌های مختلف به کار می‌روند، استفاده نمایید.

جدول۱-۱ برخی از فضاهای نام موجود در FCL.

فضای نام	هدف
System	حاوی کلاس‌های پایه و انواع داده از قبیل double، int، char و غیره است.
System.Data	دارای کلاس‌هایی است که برای دستیابی به داده‌های بانک اطلاعات استفاده می‌شوند.
System.Io	از کلاس‌هایی تشکیل شده است که برای ورودی- خروجی داده‌ها مانند فایل‌ها به کار می‌روند.
System.Drawing	از کلاس‌هایی تشکیل شده است که برای ترسیم اشکال گرافیکی به کار می‌رود.
System.Linq	از کلاس‌هایی تشکیل شده است که برای کار با LINQ به کار می‌روند.

[4]- Namespaces

۳–۱. انواع داده‌ها

اکثر برنامه‌ها با دریافت داده‌ها، پردازش و استخراج نتایج سر و کار دارند. یعنی، داده‌ها مهم‌ترین بخش برنامه‌نویسی را ایفا می‌نمایند. لذا، باید انواع داده‌هایی که در زبان برنامه‌نویسی وجود دارند، را بیاموزیم. دو نوع داده را می‌توان در زبان ویژوال بیسیک نت تعریف نمود که عبارتند از:

۱. داده‌های مقدار۲. داده‌های مرجع

داده‌های مقدار، همان داده‌هایی هستند که توسط انواع اولیه تعریف می‌شوند (بجز داده‌های Object و String). این انواع در جدول ۲–۱ آمده‌اند و **داده‌های مرجع**، تمام انواعی هستند که توسط برنامه‌نویس تعریف می‌شوند یا کلاس‌های هستند که از قبل تعریف شده‌اند. در ادامه پیاده‌سازی کلاس را خواهید دید.

۴–۱. متغیرها

متغیرها نامی برای کلمات حافظه هستند که دارای ویژگی‌های زیر می‌باشند:

- داده‌ها در آنها ذخیره می‌شوند.
- مقدار آنها در طول اجرای برنامه ممکن است تغییر کند.
- در یک لحظه خاص فقط یک مقدار را دارند.

برای استفاده از متغیرها باید نام، نوع و مقدار آنها را تعیین کرد.

۱–۴–۱. نامگذاری متغیرها

برای نامگذاری متغیرها در ویژوال بیسیک‌نت می‌توان از ترکیبی از حروف، ارقام و خط ربط (_) استفاده کرد. به طوری که اولین آنها رقم نباشد. به عنوان مثال، sum، count1، i_1 می‌توانند نام‌هایی برای متغیرها باشند، ولی 1count و hioh!book نمی‌توانند نام متغیرها باشند. زیرا، اولین نام با رقم 1 شروع شد و در دومین نام کارکتر ! استفاده گردید که کارکترهای مجاز نمی‌باشند.

۲–۴–۱. اعلان متغیرها

بعد از این که متغیرها را نامگذاری کردید باید نوع آنها را تعیین نمایید. یعنی، باید تعیین کنید متغیر چه نوع داده‌ای را ذخیره می‌کند. متغیرها به صورت زیر اعلان می‌گردند:

Dim نام متغیر As نوع داده

در این ساختار Dim و As کلمه کلیدی می‌باشند که برای تعریف متغیر به کار می روند. نوع داده را در جدول ۲-۱ دیدید.

اکنون دستورات زیر را ببینید:

```
Dim x As Integer
Dim  d As Double
Dim s1, s2 As String
Dim obj As Object
Dim y As Boolean
```

دستور اول، متغیر x را از نوع صحیح تعریف می‌کند. دستور دوم، متغیر d را از نوع double و دستور سوم، متغیرهای s1 و s2 را از نوع رشته‌ای تعریف می‌نماید. دستور چهارم، obj را از نوع Object تعریف می‌کند و دستور پنجم y را از نوع Boolean معرفی می‌نماید.

۳-۴-۱. مقدار دادن به متغیرها

برای مقداردهی به متغیرها روش‌های متعددی وجود دارد که عبارتند از:

۱. مقداردهی در زمان اعلان متغیر ۲. پس از تعریف نوع متغیر و با دستور انتساب (=)

۳. دستورات ورودی

به عنوان مثال، دستورات زیرا را ببینید:

```
Dim x As Integer = 15
Dim s As string = "good"
```

دستور اول، متغیر x را با مقدار 15 از نوع Integer تعریف می‌کند و دستور دوم، متغیر رشته‌ای s را از نوع String تعریف می‌نماید و رشته "good" را به آن تخصیص می‌دهد. اکنون دستورات زیر را ببینید:

```
Dim x As Integer
Dim  s As String
x = 10
s = "ویژوال بیسیک نت"
```

دستورات اول و دوم متغیرهای x و s را به ترتیب از نوع Integer و String تعریف می‌کنند. دستور سوم، مقدار متغیر x را برابر ۱۰ قرار می‌دهد و دستور چهارم، مقدار متغیر s را رشته ویژوال بیسیک‌نت تعیین می‌کند.

در ادامه با چگونگی مقداردهی متغیرها با دستورات ورودی آشنا خواهید شد.

۵–۱. ثوابت

ثوابت، مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. برای تعریف ثوابت از واژه const به صورت زیر استفاده می‌شود:

const نام = مقدار ثابت

به عنوان مثال، دستورات زیر را ببینید:

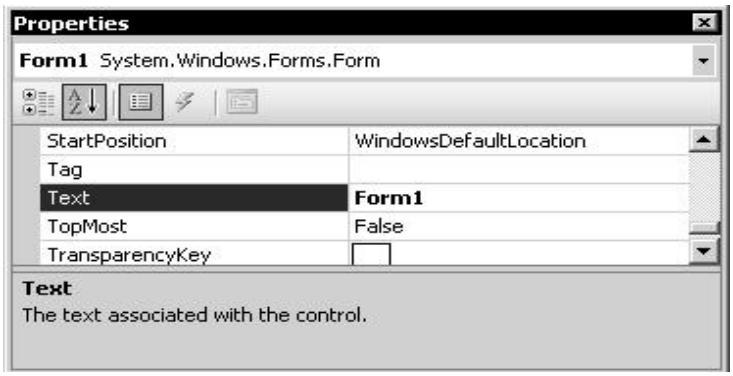
```
const PI = 3.14
const ch = '+'
```

۷–۱. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه بانک اطلاعات در ویژوال بیسیک‌نت حداقل یک فرم دارد. برای استفاده از فرم باید خواص، رویدادها و متدهای آن را بشناسیم. لذا در ادامه به این موضوعات می‌پردازیم.

۱–۷–۱. خواص فرم

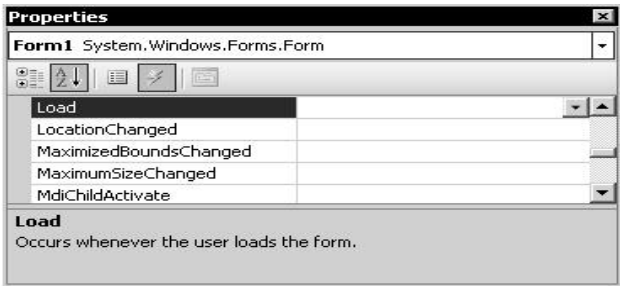
خواص فرم، شکل ظاهری فرم را تعیین می‌کنند. فرم خواص متعددی دارد. بیان همه این خواص از حوصله این کتاب خارج است. لذا، به تشریح خواص مهم فرم و کنترل‌های دیگر می‌پردازیم. برخی از خواص مهم فرم در جدول ۸–۱ آمده است. برای نمایش خواص فرم ، بر روی آن کلیک کنید و سپس کلید F4 را بزنید تا خواص فرم را ببینید (شکل زیر):



۲-۷-۱. رویدادهای فرم

همان‌طور که بیان کردیم، برنامه‌های ویژوال منتظر رخ دادن رویدادهایی هستند که توسط کاربر انتخاب می‌شوند تا به آنها پاسخ دهند. یعنی، اگر برنامه پاسخ‌گو به رویدادی نوشته شده باشد، در صورت انتخاب آن رویداد توسط کاربر، این برنامه اجرا می‌شود. فرم دارای رویدادهای مختلفی است. برخی از مهم‌ترین آنها در جدول ۹-۱ آمده‌اند.

برای مشاهده رویدادهای فرم، بر روی آن کلیک کنید تا فرم انتخاب شود. اکنون گزینه View/ Properties Window را اجرا نمایید تا پنجره Properties ظاهر شود. در این پنجره دکمه Events را کلیک کنید تا لیست رویدادهای فرم را ببینید (شکل زیر):



جدول ۸-۱ خواص مهم فرم.

خاصیت	هدف
Name	نام فرم را تعیین می‌کند. برای اولین فرم، نام فرم Form1 است که می‌توانید آن را تغییر دهید.
ActiveForm	فرم فعال فعلی را تعیین می‌کند.
Enabled	تع
Font	فونت فرم را تعیین می‌کند.
ForeColor	رنگ‌نوشته‌های‌روی‌فرم را تعیین می‌کند.
Language	زبان مورد استفاده در فرم را تعیین می‌کند.
MinimizeBox	تعیین می‌کند آیا دکمه کمینه در عنوان فرم ظاهر شود یا خیر.
RightToLeft	جهت‌نمایش‌اطلاعات‌را تعیین‌می‌کند(این خاصیت برای زبان فارسی مفید است).
Size	اندازه فرم را تعیین می‌کند.

Text	متنی را تعیین می‌کند که باید در عنوان فرم نمایش داده شود.
Location	محل قرار گرفتن فرم را تعیین می‌کند.
Opacity	سطح شفافیت فرم را تعیین می‌کند.
MaximumBox	تعیین می‌کند آیا دکمه بیشینه نمایش داده شود یا خیر.
MinimumBox	تعیین می‌کند آیا دکمه کمینه نمایش داده شود یا خیر.

۳-۷-۱. متدهای فرم

متدها، کارهای خاصی را بر روی فرم انجام می‌دهند. برخی از متدهای مهم فرم در جدول ۱۰-۱ آمده‌اند.

جدول ۱۰-۱ متدهای مهم فرم.

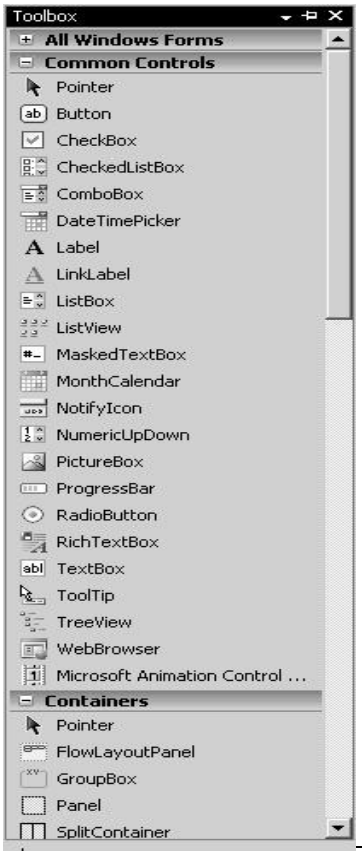
متد	هدف
Active	فرم را فعال می‌کند.
Close	فرم را می‌بندد.
Dispose	فرم را حذف کرده از بین می‌برد.
Focus	مکان‌نما را به فرم مورد نظر منتقل می‌کند.
Hide	فرم را پنهان می‌کند.
ResetText	متن عنوان فرم را پاک می‌کند.
Show	فرم مخفی شده را آشکار می‌کند.
Refresh	فرم را بازسازی کرده، اطلاعات آن را دوباره رسم می‌کند.
ToString	محتویات فرم را به رشته تبدیل می‌نماید.
Validate	موجب اعتبارسنجی فرم می‌شود.

۸-۱. کنترل‌ها

همان‌طور که می‌دانید برای نوشتن برنامه‌ها در ویژوال بیسیک‌نت باید کنترل‌هایی را بر روی فرم قرار دهید (این کنترل‌ها همان قطعات تشکیل دهنده برنامه هستند). خواص آنها را مقداردهی کرده، برنامه پاسخ‌گو به

رویدادهای آنها را بنویسید. بنابراین، کنترل‌ها اجزا اصلی برنامه‌های ویژوال بیسیک‌نت را تشکیل می‌دهند. کنترل‌های زیادی در ویژوال بیسیک‌نت وجود دارند. بحث در مورد همه اینها در این کتاب نمی‌گنجد. لذا، در این کتاب برخی از کنترل‌های مهم را بررسی می‌کنیم.[۵]

این کنترل ما را در ادامه خواهید دید. برای اضافه کردن کنترل جدید بر روی فرم، دکمه Toolbox () را کلیک کنید تا کنترل‌های آماده را مشاهده کنید. اکنون جعبه ابزار ظاهر می‌شود (شکل ۱-۱). در این جعبه ابزار، کنترل مورد نظر را کلیک مضاعف کنید تا به فرم اضافه گردد و آن را به مکان دلخواه از فرم انتقال دهید (با کشیدن و رها کردن). من دکمه button1 را به فرم اضافه کردم و به مکان دلخواه انتقال دادم (شکل زیر):



شکل ۱-۱ پنجره ابزار ویژوال استودیونت

برای حذف کنترلی از فرم، آن را کلیک کرده تا انتخاب شود. اکنون دکمه Delete را فشار دهید تا کنترل از روی فرم حذف گردد.

۱-۸-۱. کنترل Label ()

این کنترل برای نمایش متن‌های غیرقابل ویرایش از قبیل عنوان فیلد و پیام‌های مورد نیاز به کار می‌رود. این کنترل نیز دارای خواص و رویدادهای زیادی است که برخی از مهم‌ترین آنها در زیر آمده است:

- **خاصیت AutoSize**: تعیین می‌کند آیا اندازه کنترل با توجه به مقدار خاصیت Text به طور خودکار تغییر کند یا خیر.
- **خاصیت TextAlign**: مکان قرار گرفتن متن در کنترل Label را تعیین می‌کند. به عنوان مثال، دستور زیر را ببینید.

```
label1.TextAlign = ContextAlignMent.TopLeft
```

این دستور، متن را در بالا سمت چپ نمایش می‌دهد.

[۱]- براي كسب اطلاعات بيشتر در مورد ويژوال بيسيكنت به كتاب آموزش گام به گام ويژوال جعفرنژادقمي و رمضان عباسنژاد مراجعه كنيد.

رویداد AutoSizeChanged: وقتی رخ می‌دهد که مقدار خاصیت AutoSize تغییر کند.

رویداد TextAlignChanged: وقتی رخ می‌دهد که مقدار خاصیت TextAlign تغییر یابد.

جدول ۱۱-۱ خواص، رویدادها و متدهای کنترل TextBox.

خاصیت	هدف
Lines	رشته‌ای از نوع آرایه است که تعداد خط‌های TextBox را تعیین می‌کند.
MaxLength	حداکثر طول متن را تعیین می‌کند. به عنوان مثال، اگر در این خاصیت ۵۰ را وارد کنید، کاربر حداکثر می‌تواند ۵۰ کاراکتر را وارد کند.
MultiLines	تعیین می‌کند آیا TextBox می‌تواند چند سطر اطلاعات را بپذیرد یا خیر.
Readonly	تعیین می‌کند که آیا محتویات TextBox فقط خواندنی باشد یا خیر. اگر اطلاعات TextBox فقط خواندنی باشد، این کنترل مانند Label عمل می‌کند.
PasswordChar	تعیین می‌کند در هنگام ورود اطلاعات در TextBox آیا اطلاعات به صورت کلمه عبور ظاهر شوند (اطلاعات اصلی مخفی گردند یا خیر).
رویداد	**هدف**
MultiLinesChanged	زمانی رخ می‌دهد که خاصیت MultiLines تغییر یابد.
ReadonlyChanged	زمانی رخ می‌دهد که خاصیت Readonly تغییر یابد.
متد	**هدف**
Clear	محتویات TextBox را پاک می‌کند.
ClearUndo	اثر اجرای فرمان Clear انجام شده را خنثی می‌کند.
Copy	متن انتخاب شده در TextBox را در حافظه موقت کپی می‌کند.
Cut	متن انتخاب شده در TextBox را به حافظه موقت انتقال می‌دهد.
Paste	متن حافظه موقت را در مکان فعلی کپی می‌نماید.
Undo	تأثیر آخرین فرمان انجام شده را خنثی می‌کند.
Select	برای انتخاب متن TextBox به کار می‌رود.

SelectAll	کل متن TextBox را انتخاب می‌کند.
DeselectAll	کلیه متن انتخاب شده TextBox را از حالت انتخاب خارج می‌کند.

۲-۸-۱. کنترل TextBox ()

(اطلاعات رشته‌ای طولانی) Memoاین کنترل برای دریافت اطلاعاتی از قبیل مقادیر رشته‌ای، عددی و فیلدها به کار می‌رود. این کنترل نیز مانند کنترل‌های دیگر دارای خواص، رویدادها و متدهای متعددی است. برخی از خواص، رویدادها و متدهای این کنترل در جدول ۱۱-۱ آمده‌اند.

۳-۸-۱. کنترل Button ()

این کنترل‌ها به دکمه فرمان معروف هستند. زیرا با کلیک آن فرمان خاصی (دستورات خاصی) اجرا خواهد شد. خواص، رویدادها و متدهای این کنترل مانند کنترل‌های دیگر است. در ادامه بیشتر با این کنترل آشنا خواهید شد.

فصل ۲

مبانی بانک اطلاعات SQL Server

امروزه سازمان‌ها، مؤسسات، ادارات و شرکت‌ها با حجم عظیمی از داده‌ها سر و کار دارند. به عنوان مثال، فرض کنید بخواهید اطلاعات مربوط به مکالمات شرکت مخابرات یکی از استان‌ها را نگهداری کنید. به طوری که در یک سال حدود ۱۵ میلیارد رکورد جمع‌آوری می‌گردد. نگهداری، پردازش و بازیابی این حجم اطلاعات از طریق فایل‌های معمولی زمان‌بر است. برای جلوگیری از تکرار بی‌مورد داده‌ها (افزونگی داده‌ها)، ایجاد سازگاری بین گزارش‌ها و صرفه‌جویی در میزان حافظه، به کارگیری بانک اطلاعات به صورت یک ضرورت درآمده است. یعنی، بدون استفاده از بانک اطلاعات نمی‌توان اطلاعات مربوط به مکالمات تلفن ثابت یک استان را نگهداری و ذخیره کرد. از طرف دیگر، اکثر برنامه‌های کاربردی که با داده‌ها سر و کار دارند، داده‌ها را در بانک اطلاعات ذخیره می‌نمایند و از طریق بانک اطلاعات آن را پردازش می‌کنند.

بانک‌های اطلاعات متعددی وجود دارند که از جمله می‌توان Access، SQL Server، Oracle، DB2 و My SQL را نام برد. هر یک از این بانک‌های اطلاعات کاربرد خاصی دارند. در بین این بانک‌ها، بانک اطلاعات SQL Server از محبوبیت خاصی برخوردار است. زیرا، حدود ۷۰ درصد از کاربران دنیا از این بانک اطلاعات استفاده می‌کنند. به همین خاطر، در این فصل ابتدا با مفاهیم بانک اطلاعات و دلایل استفاده از آن آشنا خواهیم شد. سپس، یک بانک اطلاعات نمونه را مثال می‌زنیم و در ادامه کتاب، این بانک اطلاعات را ایجاد کرده، اعمال مختلف را بر روی آن انجام می‌دهیم.

۱-۲. تعریف سیستم مدیریت بانک اطلاعات

سیستم مدیریت **بانک اطلاعات**، مکانیزیم نگهداری رکوردها[1] است. یعنی، بانک اطلاعات مخزنی برای نگهداری از داده‌ها است که کاربران آن می‌توانند اعمالی از قبیل:

۱. افزودن جداول خالی به بانک اطلاعات
۲. افزودن رکوردهایی به جداول بانک اطلاعات
۳. تغییر ساختار جداول

1- Records **2** - Query

٤. حذف رکوردهای بانک اطلاعات

٥. تغییر داده‌های بانک اطلاعات

٦. اجرای پرس‌وجو[٧] بر روی جداول

به عبارت ساده‌تر، سیستم مدیریت بانک اطلاعات، سیستمی کامپیوتری است که هدف آن ذخیره و بازیابی داده‌ها می‌باشد. بانک اطلاعات داده را پردازش نموده، به اطلاعات تبدیل کرده، آنها را بازیابی می‌نماید.

۱-۱-۲. دلایل استفاده از بانک اطلاعات

شاید از خودتان بپرسید، بانک اطلاعات چه مزایایی دارد؟ پاسخ به این سوال به مواردی از قبیل اندازه سیستم، تعداد کاربران سیستم و غیره بستگی دارد. هرچه اندازه سیستم بزرگ‌تر شود و تعداد کاربران بیشتر گردند، ضرورت به کارگیری بانک اطلاعات بیشتر خواهد شد. برای بیان مزایای بانک اطلاعات مثال زیر را در نظر بگیرید:

فرض کنید، برای فروشگاه ساده‌ای بانک اطلاعات طراحی می‌کنید. این بانک اطلاعات بسیار کوچک و ساده است و شاید امتیازات استفاده از بانک اطلاعات به چشم نیاید. امّا همین بانک اطلاعات را برای فروشگاه زنجیره‌ای بزرگ در نظر بگیرید که دارای انبارهای زیادی است و موجودی انبارها به سرعت تغییر می‌کند. امتیازات سیستم بانک اطلاعات نسبت به سیستم سنتی که رکوردها بر روی کاغذ نگهداری می‌شوند، در این‌گونه موارد، بیشتر به چشم می‌آیند. برخی از مزایای بانک اطلاعات در زیر آمده‌اند:

- **فشردگی**: چون داده‌های بانک اطلاعات دارای ساختار هستند، نیازی به نگهداری فایل‌های متنی حجیم نیست و از ورود داده‌های نامنظم (بدون ساختار) جلوگیری می‌کند. بنابراین، باعث فشرده‌سازی اطلاعات می‌گردد.
- **سرعت**: سیستم می‌تواند سریع‌تر از انسان داده‌ها را بازیابی و به هنگام کند. مخصوصاً زمانی که سیستم توزیع شده باشد (مانند فروشگاه زنجیره‌ای)، پاسخ‌گویی به درخواست‌های **سراسری** و موردی بسیار سریع‌تر انجام می‌گردد.
- **بودجه کمتر**: خیلی از یکنواختی‌ها در نگهداری فایل‌ها به روش دستی و سنتی که به فضای زیادی نیاز دارند، حذف خواهند شد و دیگر نیازی به ساختمان‌های بزرگ و کارمندان زیادی برای نگهداری و پردازش اطلاعات نمی‌باشد.

- **دسترسی:** در هر زمان اطلاعات دقیق و به هنگام شده، در اختیار قرار می‌گیرند. زیرا، اطلاعات به صورت مجتمع و یک‌پارچه نگهداری می‌شوند.
- **حفاظت:** داده‌ها می‌توانند در مقابل دستیابی غیرقانونی و غیرمجاز حفظ شوند. زیرا، اطلاعات در یک نقطه نگهداری می‌گردند. بنابراین، می‌توان از طریق فایروال، تعریف حساب کاربری و کلمه عبور از ورود افراد غیرمجاز جلوگیری کرد.

البته این مزایا در محیط چند کاربره که بانک‌های اطلاعات بزرگ و پیچیده باشند، چشمگیرتر است. اما، یک امتیاز ویژه در چنین محیطی وجود دارد و آن عبارت است از: **سیستم بانک اطلاعات موجود می‌شود تا مؤسسه** بر روی داده‌هایش **کنترل مرکزی** داشته باشد (این موضوع از اهمیت ویژه‌ای برخوردار است). این وضعیت با وضعیتی که در مؤسسات بدون بانک اطلاعات کار می‌کنند، متفاوت است. در مؤسسات فاقد بانک اطلاعات، هر برنامه کاربردی فایل‌های خاص خودشان را دارند، گاهی نیز نوارها و دیسک‌های مخصوص به خود دارند. بنابراین، داده‌ها پراکنده‌اند و کنترل بر روی داده‌ها با روش سیستماتیک دشوار است.

۲-۱-۲. طراحی بانک اطلاعاتی

بانک‌های اطلاعاتی مختلفی وجود دارند که مهم‌ترین آنها بانک اطلاعات رابطه‌ای است. اطلاعات در بانک اطلاعات رابطه‌ای، بین جداول مختلف توزیع می‌گردند تا ذخیره‌سازی و بازیابی اطلاعات بهینه‌تر شود. یعنی، از افزونگی داده‌ها جلوگیری می‌گردد، بی‌نظمی را کاهش می‌دهد و داده‌های تهی را نیز کاهش خواهد داد. این زمانی اتفاق می‌افتد که بانک اطلاعات خوب طراحی گردد. بنابراین، هرچه بانک اطلاعات بهتر طراحی شود، ابزار مهمی برای مدیریت بر اطلاعات شخصی، تجاری یا اداری است. ولی چنانچه بانک اطلاعات بد طراحی گردد، ارزش چندانی نخواهد داشت. پس، هرچه وقت بیشتری برای طراحی و تحلیل داده‌ها صورت گیرد، نتیجه بهتری بدست می‌آید. زمانی که طراحی کامل بررسی گردید، به راحتی می‌توان بانک اطلاعات را ایجاد نمود.

فرآیند طراحی با تحلیل کارهایی شروع می‌گردد که برای بانک اطلاعات نیاز داریم. در این فرآیند، ابتدا، باید مشخص کنید سیستم چه کاری را باید انجام دهد. با کاربران مصاحبه کرده تا به خواسته‌های آن‌ها پی ببرید. توجه داشته باشید که فرآیند طراحی، فرآیندی تکراری است. وقتی کاربران می‌خواهند از سیستم جدید استفاده کنند، راجع به ویژگی‌های آن فکر می‌کنند، مثل فرم‌های ورود داده‌ها، پرس‌وجوهای خاص، و فیلدهای محاسباتی.

از طرف دیگر، طراحی باید در یک نقطه خاتمه یابد و توسعه بانک اطلاعات شروع گردد. در این صورت، خواسته‌های جدید سیستم را می‌توانید در نسخه‌های بعدی سیستم منظور کنید. فرآیند طراحی بانک اطلاعات را می‌توان به مراحل زیر تقسیم کرد که هر مرحله دارای هدف خاصی است:

۱. **تعیین خواسته‌های کاربران:** در این مرحله، با کاربران مصاحبه می‌گردد. فرم کاربران بررسی می‌شود تا خواسته‌های آنها از بانک اطلاعات مشخص گردد.

۲. **توزیع داده‌ها در جداول:** یکی از مهم‌ترین بخش‌های طراحی توزیع داده‌های جداول است. زیرا، چنانچه طراحی جداول خوب باشد، از تکرار بی‌مورد، بی‌نظمی و ورود داده‌های تهی در جداول جلوگیری می‌کند. این کار با نرمال‌سازی جداول اتفاق می‌افتد که در ادامه نرمال‌سازی را می‌بینید.

۳. فیلدهای هر رکورد را در هر جدول مشخص نمایید.

٤. برای هر جدول کلید اولیه و کلیدهای کاندید را تعیین کنید تا تضمین شود که هیچ دو رکورد یکسان نباشند.

۵. ارتباط بین جداول را تعیین کنید تا بتوانید پرس‌وجوها را از چند جدول تهیه کنید.

٦. طراحی را با کاربران مرور کنید تا مطمئن شوید، بانک اطلاعات طراحی شده نیازهای کاربران را برطرف می‌کند.

۷. جداول بانک اطلاعات را ایجاد کرده داده‌ها را در آنها وارد نمایید.

۸. کارایی بانک اطلاعات را تحلیل کنید و بانک اطلاعات طراحی شده را بهینه نمایید تا بتوانید سریع‌تر نتایج پرس‌وجوها را دریافت کنید.

فصل ۳ ایجاد بانک اطلاعات و جداول آن

در فصل‌های ۱ و ۲ با مفاهیم اولیه ویژوال بیسیک‌نت و بانک اطلاعات آشنا شدیم. در این فصل می‌خواهیم یک بانک اطلاعات را از طریق برنامه ویژوال بیسیک‌نت ایجاد کرده، جداول آن را ایجاد کنیم. قبل از این که بتوانیم بانک اطلاعات را ایجاد کنیم باید مفاهیم زیر را بی‌آموزیم:

۱. ایجاد بانک اطلاعات و جداول آن در SQL Server

۲. مفهوم ADO.NET

۳. اتصال به بانک اطلاعات

٤. آشنایی با شیء Command

در این فصل، ابتدا این مفاهیم را می‌آموزیم و سپس با یک مثال بانک اطلاعاتی که در فصل ۲ بیان گردید را در ویژوال بیسیک‌نت ایجاد کرده و جداول را به آن اضافه خواهیم کرد.

۱-۳. ایجاد بانک اطلاعات و جداول آن

در فصل دوم یک بانک اطلاعات نمونه را دیدید. جداول آن و ساختار آنها را مشاهده کردید. قبل از اینکه بخواهید کاری را بر روی بانک اطلاعات انجام دهید، باید آن را ایجاد نموده، جداول آن را به آن اضافه کنید تا در فصل‌های بعدی بتوانید داده‌ها را در آن وارد نمایید. داده‌های موجود را ویرایش کرده یا حذف کنید. بنابراین، در این بخش به ایجاد، تغییر، حذف بانک اطلاعات و جداول آن می‌پردازیم.

۱-۱-۳. ایجاد بانک اطلاعات

یک سرویس‌دهنده بانک اطلاعات SQL Server می‌تواند چندین بانک اطلاعات را به طور همزمان نگهداری کند. بنابراین باید بتوان بانک اطلاعات جدید ایجاد کرد. برای این منظور از دستور CREATE DATABASE به صورت زیر استفاده می‌شود:

```
CREATE DATABASE database_name      [ ON
```

```
    [ PRIMARY ] [ <filespec> [ ,...n ]
    [ , <filegroup> [ ,...n ] ]
    [ LOG ON { <filespec> [ ,...n ] } ]
    ]
    [ COLLATE collation_name ]
    [ WITH <external_access_option> ] ][;]
    To attach a database
CREATE DATABASE database_name
    ON <filespec> [ ,...n ]
    FOR { ATTACH [ WITH <service_broker_option> ]
    | ATTACH_REBUILD_LOG } [;]
    <filespec> ::=   {
    (
      NAME = logical_file_name ,
      FILENAME = 'os_file_name'
        [ , SIZE = size [ KB | MB | GB | TB ] ]
        [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
        [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
    ) [ ,...n ]      }
    <filegroup> ::=    {
    FILEGROUP filegroup_name [ DEFAULT ]
        <filespec> [ ,...n ]  }
```

شرح پارامترهای این دستور در جداول ۱-۳ آمده است.

جدول ۱-۳ گزینه‌های دستور CREATE DATABASE.

گزینه	هدف
database_name	نام بانک اطلاعات را تعیین می کند که باید در یک سرویس‌دهنده (سرور) یکتا باشد. نام بانک اطلاعات از قانون نام‌گذاری شناسه پیروی می‌کند و حداکثر می‌تواند ۱۲۳ حرف باشد.
ON	فایل داده بانک اطلاعات را تعیین می‌کند. filespec، نام فایل داده و filegroup لیستی از گروه‌های فایل کاربر و فایل‌های آنها را تعیین می‌کند.
LOG ON	فایل کارنامه (Log) را تعیین می‌کند. اگر تعیین نگردد، یک فایل کارنامه به طور خودکار ایجاد می‌شود که اندازه آن ۲۵ درصد اندازه فایل داده است.
FOR ATTACH	برای ایجاد بانک اطلاعات از طریق فایل‌های سیستم عامل به کار می‌رود. filespec نام فایل اصلی بانک اطلاعات را تعیین می‌نماید.
COLLATE	فونت پیش‌فرض بانک اطلاعات را تعیین می‌کند. با این پارامتر می‌توان زبان بانک

	اطلاعات را نیز تعیین نمود.
PRIMARY	لیست filespecهای فایل اصلی بانک اطلاعات را تعریف می‌کند.
NAME = Logical_file_name	نام منطقی فایل را تعیین می‌کند که SQL Server با این نام منطقی آن را می‌شناسد.
FILENAME = os_file_name	نام فایل بانک اطلاعات را در سطح سیستم عامل تعیین می‌کند.
SIZE	اندازه فایل اصلی filespec را تعیین می‌کند.
MAXSIZE	حداکثر اندازه فایل filespec را تعیین می‌کند.
FILEGROWTH = Growth_increment	هنگامی که فضای فایل پر شود، این گزینه حداکثر رشد فایل‌های اصلی(filespec) را تعیین می‌کند. اگر ذکر نشود، فایل آنقدر رشد می‌نماید تا دیسک پر گردد.
FILEGROUP Filegroup_name	نام گروهی را تعیین می‌کند که فایل‌های filespec1، filespec2، ... و filespecn در آن گروه قرار دارند.

به عنوان مثال، دستورات زیر را ببینید:

```
CREATE DATABASE Test
 ON PRIMARY (NAME = Test_data,
   FILENAME = 'E:\Test\Test_data.MDF',
   SIZE = 40,
   MAXSIZE = 100)
 LOG ON (NAME = Test_log,
   FILENAME = 'E:\Test\Test_log.LDF',
   SIZE = 40,
   MAXSIZE = 100)
GO
```

این دستورات بانک اطلاعات Test را در درایو E پوشه Test ایجاد می‌کند. فایل‌های این بانک دارای ویژگی‌هایی از قبیل اندازه 40MB و حداکثر اندازه 100MB است.

۲-۱-۳. تغییر خواص بانک اطلاعات موجود

در بخش قبل آموختید که چگونه بانک اطلاعات را ایجاد کنید. گاهی نیاز است خواص بانک اطلاعات از قبیل اندازه، حداکثر اندازه و غیره را پس از ایجاد بانک اطلاعات تغییر دهید. برای این منظور می‌توانید از دستور ALTER DATABASE استفاده کنید. این دستور به صورت زیر به کار می‌رود:

```
ALTER DATABASE database_name  {
   <add_or_modify_files>
     | <add_or_modify_filegroups>
```

```
    | <set_database_options>
    | MODIFY NAME = New_database_name
    | COLLATE collation_name    }
[;]
<add_or_modify_files>::=  {
    ADD FILE <filespec> [ ,...n ]
        [ TO FILEGROUP { filegroup_name | DEFAULT } ]
  | ADD LOG FILE <filespec> [ ,...n ]
  | REMOVE FILE logical_file_name
  | MODIFY FILE <filespec>   }
<filespec>::=  (
    NAME = logical_file_name
    [ , NEWNAME = New_logical_name ]
    [ , FILENAME = 'os_file_name' ]
    [ , SIZE = size [ KB | MB | GB | TB ] ]
    [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
    [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB| % ] ]
    [ , OFFLINE ] )
<add_or_modify_filegroups>::=  {
    | ADD FILEGROUP filegroup_name
    | REMOVE FILEGROUP filegroup_name
    | MODIFY FILEGROUP filegroup_name
        { <filegroup_updatability_option>
        | DEFAULT
        | NAME = New_filegroup_name  }    }
```

توضیح: برخی از پارامترهای ALTER DATABASE در جدول ۲-۳ آمده است.

به عنوان مثال، دستور زیر را در نظر بگیرید:

```
ALTER DATABASE Test
 MODIFY FILE (NAME = Test_data,
   SIZE = 50,
   MAXSIZE = 110)
GO
```

این دستور اندازه و حداکثر اندازه بانک اطلاعات Test را به 50MB و 110MB تغییر می‌دهد.

جدول ۲-۳ گزینه‌های دستور ALTER DATABASE.

گزینه	هدف
database_name	نام بانک اطلاعات را تعیین می‌کند که باید مشخصات آن تغییر کند.
ADD FILE <filespec>…	فایل‌های filespec1 تا filespecn را به بانک اطلاعات اضافه می‌کند.
To FILEGROUP filegroup_name	فایل‌های اضافه شده را در گروه filegroup_name قرار می‌دهد.

ADD LOG FILE filespec …	فایل‌های کارنامه filespec1 تا filespecn را به بانک اطلاعات اضافه می‌کند.
REMOVE FILE logical_ file_name	فایل logical_file_name را از بانک اطلاعات حذف می‌کند. این فایل قبل از حذف، باید با دستور EMPTYFILE خالی شود.
ADD FILEGROUP filegroup_name	یک گروه جدیدی به نام filegroup_name اضافه می‌کند.
MODIFY FILE filespec	فایل filespec را برای تغییراتی از قبیل نام، اندازه و غیره آماده می‌کند.
MODIFY NAME= New_ dbname	نام بانک اطلاعات را به New_dbname تغییر می‌دهد.
MODIFY FILEGROUP filegroup_name …	گروه فایل را مشخص می‌کند که باید تغییر یابد. این تغییرات شامل فقط خواندنی (READ ONLY) و خواندنی و نوشتنی (READ WRITE) است.
WITH <termination	وقتی تراکنش را از حالتی (مدی) به حالت دیگر می‌برید، این پارامتر تعیین می‌کند تراکنش‌های ناقص چه موقع عقب‌گرد (Rollback) کنند.
COLLATE <collation_name>	نام تلفیق بانک اطلاعات را به collation_name تغییر می‌دهد.

۳-۱-۳. حذف بانک اطلاعات

بانک‌های اطلاعات فضای روی دیسک را اشغال می‌کنند. بنابراین باید بانک‌های اطلاعات اضافی را حذف نمود. برای حذف بانک اطلاعات موجود می‌توانید از دستور DROP DATABASE استفاده کنید. اگر بانک اطلاعات را حذف نمایید، کلیه اشیا آن حذف خواهند شد که قابل بازیابی نمی‌باشند. بنابراین، در هنگام استفاده از این دستور دقت نمایید تا بانک‌های اطلاعات مورد نیازتان را حذف نکنید. دستور DROP DATABASE به صورت زیر به کار می‌رود:

```
DROP DATABASE { database_name } [ ,...n ] [;]
```

database_name[, .. , n]، نام بانک‌های اطلاعات را تعیین می‌کنند که می‌خواهید حذف کنید. اگر بانک اطلاعات را حذف کنید کلیه جداول و اشیا آن حذف خواهند شد.

اکنون دستورات زیر را اجرا کنید.

```
DROP DATABASE Test
GO
```

این دستورات بانک اطلاعات Test را حذف می‌کنند.

در هنگام اجـرای دسـتورات ALTER DATABASE و DROP DATABASE، اگـر بانـک اطلاعـات کـه می‌خواهیم خواص آن را تغییر دهیم یا آن را حذف کنیم، موجود نباشد، پیام خطا ظاهر می‌شود. ولی در هنگـام اجرای دستور CREATE DATABASE چنانچه بانک اطلاعات که می‌خواهید ایجاد کنید از قبل موجود باشـد، پیام خطا ظاهر خواهد شد.

۲-۳. اشیای بانک اطلاعات

بانک اطلاعات برای اهداف تجاری طراحی شده است. با ایجاد بانک اطلاعات اهداف تجـاری آن بـرآورده نمی‌شود. برای اینکه اهداف تجاری بانک اطلاعات برآورده شود، باید اشیای بانـک اطلاعـات را بـه آن اضـافه نمود. بانک اطلاعات دارای اشیای متعددی است که هر یک از اشیا وظیفه خاصی را به عهده دارند. برخی از اشیای بانک اطلاعات در زیر آمده‌اند:

۱. **شیء Database Diagrams**: برای ذخیره کردن ارتباط بین جداول در بانک اطلاعات رابطه‌ای بـه کـار می‌رود. ارتباط بین جداول می‌تواند از افزونگی داده‌ها جلوگیری کند.

۲. **شیء Tables**: یکی از مهم‌ترین اشیای بانک اطلاعات است که برای ذخیره (نگهداری) داده‌هـای بانـک اطلاعات به کار می‌رود.

۳. **شیء Views**: از طریق یک یا چند جدول می‌توان جدول‌های مجازی دیگری را ایجاد کرد تـا بتـوان بـا توجه به نیاز کاربران، فیلدهای خاصی را در اختیار آنها قرار داد. به جداول مجازی که در بانک اطلاعات ایجـاد می‌کنید، **دید**[۸] گویند. این شیء برای نگهداری دیدهایی که از جداول ایجاد شده‌اند، به کار می‌رود.

٤. **شیء Synonyms**: نام دیگری برای هر یک از حوزه اشیا شما می‌باشد.

۵. **شیء Programmability**: برای ذخیره کردن اشیایی از قبیل رویـه‌هـای ذخیـره شـده و توابـع بـه کـار می‌رود. در فصل‌های بعد با مفهوم توابع و رویه‌های ذخیره شده بیشتر آشنا خواهید شد و چگونگی اسـتفاده از آنها را در ویژوال بیسیک‌نت خواهید دید.

٦. **شیء Service Brokers**: سرویس‌های Broker بانک اطلاعات را ذخیره می‌کند.

۷. **شیء Security**: اشیای امنیتی بانک اطلاعات را ذخیره می‌کند.

1- View

در ادامه با نحوه ایجاد اشیا بانک اطلاعات آشنا خواهید شد و چگـونگی بـه کـارگیری آنهـا را در ویـژوال بیسیک‌نت خواهید دید.

۱-۲-۳. ایجاد جدول با دستور SQL

همانطور که می‌دانید جدول از مجموعه‌ای از رکوردها تشکیل می‌شود و رکورد نیز از مجموعه‌ای از فیلدها تشکیل می‌گردد و فیلدها نیز حاوی تعدادی از خواص هستند. بنابراین قبل از اینکه به ایجاد جـدول بپـردازیم، باید خواص فیلدها را بررسی کنیم. این خواص در زیر آمده‌اند:

- **نام فیلد:** هر فیلد در جدول دارای یک نام یکتا[۹] است کـه از قـانون نـام‌گـذاری شناسـه‌هـا در بانـک اطلاعات پیروی می‌کند.
- **نوع فیلد:** برای هر فیلد باید تعیین شود، اولاً چه نوع داده‌ای را مـی‌توانـد ذخیـره کنـد، ثانیـاً تعـداد بایت‌هایی که می‌تواند ذخیره کند، چقدر است. انواع داده‌ها و مقدار فضایی کـه هـر یـک از انـواع در بانک اطلاعات نیاز دارند، در جدول ۳-۳ آمده است.
- **محدودیت‌های فیلد:** این ویژگی تعیین می‌کند هر فیلد دارای چه محدودیت‌هایی (قیـدهای) اسـت که برخی از این محدودیت‌ها در زیر آمده‌اند:

۱. **فیلد کلید اولیه**[۱۰]: فیلدی کلیدی اولیه است که دارای شرایط زیر باشد:

- در یک جدول مقدار هیچ دو رکوردی از این فیلد یکسان نباشد.
- اطلاعات رکوردها براساس این فیلد مرتب باشند.
- مقدر این فیلد نمی‌تواند تهی[۱۱] باشد.

۲. **فیلد کلید خارجی**[۱۲]: برای ارتباط بین دو جدول به کار می‌رود. یعنی، با اسـتفاده از کلیـد اولیـه یـک جدول و کلید خارجی جدول دیگر می‌توان ارتباط بین این دو جدول را برقرار کرد. کلید خارجی در واقع کلید اصلی جدول دیگر در همان بانک است.

۳. **مقدار تهی**: تعیین می‌کند که مقدار فیلد می‌تواند تهی باشد. اگر در فیلـدی مقـدار جـای خـالی[۱۳] وارد کنید، مقدار آن فیلد تهی نیست.

1- Unique **2**- Primary Key **3**- NULL **4**- Foreign Key **5**- Space

٤. **مقدار NOT NULL**: مقدار فیلد نمی‌تواند تهی باشد، یعنی حتماً باید در فیلدهایی که محدودیت NOT NULL دارند، مقدار وارد شود.

٥. **محدودیت UNIQUE**: تعیین می‌کند مقدار فیلدی که دارای این محدودیت است، در جدول منحصر به فرد می‌باشد و برای این فیلد نمی‌توان مقدار تکراری وارد نمود.

٦. **محدودیت References**: این محدودیت برای ایجاد ارتباط بین دو جدول به کار می‌رود . کلید خارجی که ارتباط بین جداول را برقرار می‌کند، در جلوی این واژه به همراه نام جدولی که کلید اولیه در آن جدول وجود دارد، قرار می‌گیرد.

٧. **محدودیت DEFAULT**: این محدودیت مقداری را تعیین می‌کند که اگر کاربر برای فیلد مقداری را وارد نکند، آن مقدار در این فیلد قرار می‌گیرد. یعنی، مقدار پیش‌فرض فیلد را تعیین می‌نماید.

.٨ **محدودیت IDENTITY**: موجب می‌شود تا این فیلد مانند یک فیلد AutoNumber در اکسس عمل کند. در جدول‌هایی که فیلد کلید وجود ندارد، معمولاً فیلدی از این نوع انتخاب می‌کنیم تا به عنوان فیلد کلید عمل کند.

اکنون می‌توانیم به ایجاد جداول بپردازیم. برای ایجاد جدول دو روش وجود دارد:

١. ایجاد جدول با روش طراحی ٢. ایجاد جدول با دستور SQL

اگر بخواهید از بانک اطلاعات به صورت توزیع شده استفاده کنید، بهتر و راحت‌ترین روش ایجاد جداول استفاده از دستورات SQL است. در این روش، به جای این که کاربر ستون‌های (فیلدهای) جدول را با روش Enterprise تغییر دهد، یک اسکریپت (مجموعه‌ای از دستورات SQL است) می‌نویسد و به کاربران در مکان‌های مختلف اجازه اجرای این اسکریپت را خواهد داد. برای ایجاد جدول در SQL می‌توانید از دستور CREATE TABLE به صورت زیر استفاده کنید:

```
CREATE TABLE
   [ database_name . [ schema_name ] .] table_name
    ( { <column_definition> | <computed_column_definition> }
   [ <table_constraint> ] [ ,...n ] )
   [ ; ]
   <column_definition> ::=  column_name <data_type>
     [ COLLATE collation_name ]
     [ NULL | NOT NULL ]
     [
     [ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
```

```
| [ IDENTITY [ ( seed ,increment ) ]
```

در این دستور پارامترهای زیر به کار می‌روند:

- **database_name**: نام بانک اطلاعات را تعیین می‌کند که جدول در آن ایجاد می‌گردد.
- **schema_name**: نام شمای بانک اطلاعات را تعیین می‌کند که جدول در آن ایجاد می‌شود.
- **table_name**: نام جدولی را تعیین می‌کند که باید ایجاد شود.
- **column_definition**: تعریف ستون‌های جدول قرار می‌گیرد.
- **computed_column_definition**: فیلدهای محاسباتی را تعیین می‌کند.
- **table_constraint**: محدودیت‌های جدول را تعریف می‌کند.
- **column_name**: نام ستون‌های جدول را تعیین می‌کند.
- **data_type**: نوع ستون‌های جدول را مشخص می‌نماید.
- **COLLATE**: زبان دریافت اطلاعات ستون جدول را تعیین می‌کند.
- **NULL**: محدودیتی است که تعیین می‌کند محتوای ستون جدول می‌تواند تهی باشد.
- **NOT NULL**: محدودیتی است که تعیین می‌نماید محتوای ستون جدول نمی‌تواند تهی باشد.
- **CONSTRAINT**: محدودیت فیلد را تعیین می‌کند.
- **IDENTIIY**: فیلدهای افزایشی را تعریف می‌کند که seed مقدار شروع فیلد افزایشی و increment مقدار افزایش را تعیین می‌کند.

به عنوان مثال، دستورات زیر را ببینید:

```
USE Test
GO
CREATE TABLE [dbo].[T1]
 (
   [x] [numeric](18,0) PRIMARY KEY,
   [y] [numeric](18,0) NOT NULL,
   [Comment] [varchar](250) COLLATE Arabic_CI_AS NULL
 ) ON [PRIMARY]
GO
```

این دستورات جدولی به نام T1 در بانک اطلاعات Test ایجاد می‌کنند که دارای ۳ فیلد x و y و Comment است (فیلد کلید اولیه x است).

۳-۳-۳. کلاس‌های Connection

اولین قدم در عملیات بانک اطلاعاتی، برقراری ارتباط بین برنامه کاربردی و بانک اطلاعاتی (منبع داده) است. این کلاس‌ها (کلاس‌های Connection)، برای ایجاد اتصال برنامه کاربردی با بانک اطلاعاتی به کار می‌روند. این کلاس‌ها علاوه بر برقراری اتصال به بانک اطلاعاتی، **شروع تراکنش**[۱۴] را امکان‌پذیر می‌نمایند. کلاس Conntection دارای خواصی است که در آن خواص می‌توان نام سرویس‌دهنده (محل بانک اطلاعاتی)، نام بانک اطلاعاتی، اندازه بسته‌ها در شبکه، نام کاربر، کلمه عبور کاربر، اطلاعات احراز هویت کاربر و اطلاعات دیگر را در هنگام اتصال به بانک اطلاعاتی تعیین کرد. از آنجایی که منابع داده متعددی وجود دارند، بنابراین کلاس‌های متفاوتی برای اتصال به بانک اطلاعاتی ارائه شدند که برخی از آنها عبارتند از:

- **کلاس DB2Connection**: برای برقراری اتصال با بانک اطلاعاتی DB2 به کار می‌رود.
- **کلاس OdbcConnection**: برای برقراری اتصال با منبع داده ODBC به کار می‌رود.
- **کلاس OleDbConnection**: برای برقراری اتصال با منابع داده‌ای استفاده می‌شود که OLEDB را پشتیبانی می‌کنند.
- **کلاس SqlConnection**: برای اتصال با بانک اطلاعاتی SQL Server نسخه۷ و بالاتر از آن مورد استفاده قرار می‌گیرد.
- **کلاس OracleConnection**: برای برقراری اتصال با بانک اطلاعاتی اوراکل به کار می‌رود.
- **کلاس SqlCeConnection**: برای برقراری اتصال با بانک اطلاعاتی SQL ServerCE به کار می‌رود.

از آنجایی که در این کتاب بانک اطلاعاتی SQL Server مورد بررسی قرار می‌گیرد، در این بخش به توضیح کلاس SqlConnection می‌پردازیم. این کلاس در فضای نام System.Data.SqlClient قرار دارد. بنابراین، برای کار با این کلاس باید اعمال زیر را انجام دهید:

۱. فضای نام System.Data.SqlClient را به برنامه اضافه کنید تا بتوانید از کلاس SqlConnection استفاده کنید. برای این منظور، دستور زیر را در بخش Imports اضافه کنید.

```
Imports System.Data.SqlClient
```

۲. یک شیء SqlConnection تعریف کرده، خواص و متدهای آن را مقداردهی کنید. برخی از خواص و متدهای کلاس SqlConnection در جداول ۳-۷ و ۳-۸ آمده‌اند. خواص و متدهای دیگر کلاس‌های

Connection از قبیل DB2Connection، OledbConnection و غیره مانند خواص و متدهای کلاس SqlConnection می‌باشند.

یکی از خواص مهم کلاس SqlConnection خاصیت ConnectionString است. در این خاصیت می‌توانید نام سرویس‌دهنده، نام بانک اطلاعاتی، نام کاربر، کلمه عبور و اطلاعات دیگر را مشخص کنید.

رشته اتصال

اگر بخواهید اطلاعات را بین برنامه کاربردی و بانک اطلاعاتی انتقال دهید باید یک اتصال به بانک اطلاعاتی برقرار کنید. برای برقراری اتصال باید پارامترهای خاصیت ConnectionString شیء SqlConnection را تعیین نمایید. این خاصیت پارامترهای متعددی دارد که برخی از مهم‌ترین آنها در زیر آمده‌اند:

- **پارامتر Server**: اطلاعاتی از قبیل نام و آدرس سرویس‌دهنده‌ای که بانک اطلاعاتی بر روی آن نصب شده است یا نام فایل بانک اطلاعاتی برای بانک‌های اطلاعاتی مبتنی بر تأمین‌کننده OLEDB از قبیل اکسس و فاکس‌پرو را مشخص می‌کند. برخی از مقادیری که این پارامتر می‌تواند بپذیرد، عبارت‌اند از: Localhost، Abbasnezhad\SqlExpress، 192.168.0.1 یا C:\myBank.mdb.

Initial Catalog: نام بانک اطلاعاتی را در SQL Server مشخص می‌کند. این پارامتر برای تأمین‌کننده‌های OLEDB مبتنی بر فایل از قبیل اکسس و فاکس‌پرو استفاده نمی‌شود. برای اوراکل به جای این پارامتر می‌توان خالی وارد نمود.

- **UID (UserID)**: حساب کاربری (نام کاربری) را تعیین می‌کند که می‌خواهد به بانک اطلاعاتی وصل گردد.
- **PWD (Password)**: کلمه عبور کاربری را تعیین می‌کند که می‌خواهد به بانک اطلاعاتی وصل شود.
- **Integrated Security**: چگونگی ایمن بودن ارتباط را مشخص می‌کند که می‌تواند مقادیر true، false و SSPI را بپذیرد (SSPI، معادل true است).
- **ConnectionTimeout**: مدت زمان انتظار به ثانیه برای برقراری ارتباط با بانک اطلاعاتی را مشخص می‌کند (مقدار پیش‌فرض این خاصیت، ۱۵ است). اگر در این مدت زمان تعیین شده نتواند اتصال را با بانک اطلاعاتی برقرار کند، خطای اتصال نشان داده می‌شود.
- **CryPI**: تعیین می‌کند آیا از رمزگذاری SSL استفاده شود یا خیر. مقدار پیش‌فرض این پارامتر false است.

- **PacketSize**: اندازه بسته شبکه را به بایت تعیین می‌کند که مقدار پیش‌فرض آن ۸۱۹۲ است. اگر می‌خواهید اندازه بسته شبکه را تغییر دهید، باید عددی مضرب ۵۱۲ وارد نمایید.
- **AttachDBFileName**: مسیر کامل یک فایل پیوست[۱۵] بانک اطلاعاتی است.
- **WorkStationID**: ایستگاه کاری را تعیین می‌کند که به سرور متصل می‌شود.
- **ApplicationName**: نام برنامه کاربردی را تعیین می کند که مقدار پیش‌فرض آن SqlClientDataProvider.Net است.
- **PerisistSecurityInfo**: زمانی که مقدارش false باشد، اطلاعات حساس امنیتی از قبیل رمز عبور به عنوان بخشی از ارتباط فعال برگردانده نمی‌شود. سعی کنید این خاصیت را با مقدار true مقداردهی نکنید، زیرا ممکن است امنیت سیستم‌تان را به خطر بی‌اندازد.

پارامترهای مختلف ConnectionString در بانک‌های مختلف در جدول ۱۰-۳ آمده‌اند.

جدول۹-۳ مقادیر Connection State.

مقدار	هدف
Broken	اتصال باز است، اما کار نمی‌کند. ممکن است بهتر باشد آن را ببندیم و مجدداً باز نماییم.
Closed	اتصال بسته است.
Connecting	اتصال در حال برقراری (وصل شدن) است، اما هنوز باز نشده است.
Executing	اتصال در حال اجرای یک فرمان است.
Fetching	اتصال در حال بازیابی (Fetch) داده است.
Open	اتصال باز است.

به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
1. conStr = "Integrated Security = SSPI;Initial Catalog = Chek; Server
    =.\\SQLEXPRESS;"
2.conStr= "Provider = Microsoft.Jet.OLED.4.0; Server=
     \Samples\Northwind.mdb;"
```

1- Attachable

دستور اول، به بانک اطلاعات Chek از نوع SQL Server که نام سرویس‌دهنده آن SQLEXPRESS است، متصل می‌شود. این سرویس دهنده بر روی کامپیوتر فعلی قرار دارد و دستور دوم، به بانک اطلاعات اکسس به نام Northwind.mdb متصل می‌شود.

مثال ۱-۳ برنامه‌ای که اعمال زیر را انجام می‌دهد:

۱. بانک اطلاعات Chek را ایجاد می‌کند.

۲. جداول Bank، Chek، tblUser، City، Ostan و Shobe را ایجاد کرده یا حذف می‌نماید.

۳. ساختار جداول ایجاد شده را تغییر می‌دهد.

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام CreateDatabase از نوع Application Windows Form ایجاد کنید.

۲. یک کنترل ComboBox و پنج کنترل Button به پروژه اضافه کنید.

۳. دستور زیر را در بخش Imports اضافه کنید:

```
Imports System.Data.SqlClient
```

٤. ناحیه خالی فرم را کلیک مضاعف کرده دستورات زیر را قبل از رویداد Form1_Load اضافه کنید:

```
Dim conStr As String
Dim Con As SqlConnection
Dim Cmd As SqlCommand
Dim SQL As String
```

دستور اول، شیء conStr را از نوع String تعریف کرده مقدار تهی[۱۶] را در آن قرار می‌دهد. دستور دوم شیء Con را تعریف کرده (از نوع SqlConnection) و مقدار تهی را به آن تخصیص می‌دهد. دستور سوم، شیء

1- Null

cmd را از نوع کلاس SqlCommand تعریف کرده مقدار تهی را به آن نسبت می‌دهد و دستور چهارم، شیء SQL (رشته‌ای که دستور SQLای که شیء cmd اجرا می‌کند، در آن قرار می‌گیرد) را از نوع String با مقدار اولیه تهی تعریف می‌کند.

۵. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form1_Load را به صورت زیر تغییر دهید:

```
Private Sub Form1_Load(ByVal sender As System.Object,ByVal e_
      As System.EventArgs) Handles MyBase.Load
  button1.Text = "ایجاد بانک اطلاعاتی"
  button2.Text = "ایجاد جدول"
  button3.Text = "حذف جدول"
  button4.Text = "تغییر ساختار جدول"
  button5.Text = "خاتمه برنامه"
  comboBox1.Items.Clear()
  comboBox1.Items.Add("Bank")
  comboBox1.Items.Add("Chek")
  comboBox1.Items.Add("tblUser")
  comboBox1.Items.Add("Shobe")
  comboBox1.Items.Add("City")
  comboBox1.Items.Add("Ostan")
  comboBox1.Text = "Bank"
  Me.RightToLeft = RightToLeft.Yes
  Me.Text = "ایجاد بانک اطلاعات و جدول آن "
End Sub
```

این دستورات، ابتدا خاصیت Text کنترل‌های button1 تا button5 را تعیین می‌کنند. سپس، چند گزینه به کنترل comboBox1 اضافه کرده خاصیت Text آن را انتخاب می‌کنند. در پایان، خاصیت RightToLeft فرم را به Yes تغییر می‌دهند تا عنوان فرم از راست به چپ نمایش داده شود و عنوان فرم را تغییر می‌دهند.

٦. دکمه button1 را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button1_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles button1.Click
  Try
    Con = New SqlConnection(conStr)
    If (Con.State =ConnectionState.Open) Then
      Con.Close()
    End If
    conStr = "Integrated Security=SSPI;Initial
        Catalog=master;Server=TCI;"
     Con.ConnectionString = conStr
     Con.Open()
```

```
     SQL = "CREATE DATABASE Chek On Primary
       (Name=Chek_Data,Filename='E:\\DB ویـژوال
بـیسیکنـت\\Chek.Mdf',
        Size = 30 )
    Log on ( Name=Chek_Log,
      FileName='E:\\DBویـژوال بـیسیکنـت\\Chek.Ldf', size = 30)"
    Cmd = New SqlCommand(SQL, Con)
    Cmd.CommandType = CommandType.Text
    Cmd.ExecuteNonQuery()
    Con.Close()
    MessageBox.Show("بـانـک مـورد نـظر ایـجـاد شد")
    Catch
       MessageBox.Show(" خطا در ایـجاد بـانـک اطلاعـات")
    End Try
End Sub
```

این دستورات، ابتدا شیء Con (اتصال) را ایجاد می‌کنند، اگر حالت اتصـال (Con) بـاز شـده باشـد، آن را می‌بندند، در ادامه رشته اتصال (conStr) را مقداردهی کرده و خاصیت ConnectionString مربوط به شیء Con را برابر رشته اتصال (conStr) قرار می‌دهد و اتصال را باز می‌کند (Con.Open()). سپس، متغیر مربوط به دستور SQL را مقدار می‌دهند و یک شیء از نوع SqlCommand تعریف کرده خاصیت CommandType آن را از نوع CommandText انتخاب می‌نمایند. در پایان شیء cmd (از نوع SqlCommand) را بر روی بانک اطلاعاتی فعلی اجرا می‌کنند، اتصال را می‌بندد (Con.Close()) و یک پیام مناسب نمایش می‌دهند. اگر در اجـرای ایـن فرآینـد خطایی رخ دهد، بخش catch یک پیام مناسب نمایش می‌دهد (این دستورات یک بانک اطلاعاتی جدید به نـام Chek ایجاد کرده یا پیام خطایی را نمایش می‌دهند).

۷. دکمه button2 را کلیک مضاعف کرده دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button2_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles button2.Click
  Try
    Con = New SqlConnection(conStr)
    If (Con.State = ConnectionState.Open) Then
      Con.Close()
    End If
    conStr = "Integrated Security=SSPI;Initial Catalog=Chek;
        Server=TCI;"
     Con.ConnectionString = conStr
     Con.Open()
     If (comboBox1.Text = "Bank") Then
        SQL = "CREATE TABLE Bank(" _
          & "BankCode Varchar(6) Primary Key," _
          & "BankName varchar(15) NOT NULL," _
```

```
            & "ShobeCode varchar(6)," _
            & "CityCode varChar(6)," _
            & "Tel varChar(15)," _
            & "Address varChar(255))"
        ElseIf (comboBox1.Text = "Chek") Then
          SQL = "CREATE TABLE Chek("  _
              &"ChekNo varchar(15) Primary Key," +  _
              "BankCode varchar(6),"  _
              &"UserName varchar(10),"  _
              &"ChekType varchar(1),"   _
              &"SodorDate varchar(10),"   _
              &"RasidDate varchar(10),"   _
              &"VosolDate varchar(10)," _
           &"Mablag numeric,"   _
           &"Comment varChar(255))"
        ElseIf (comboBox1.Text = "tblUser") Then
           SQL = "CREATE TABLE tblUser(" _
             & "UserName varchar(10) Primary Key," + _
             "Pass Varchar(10)," + _
             "FUser varChar(50)," + _
             "LUser varChar(50)," + _
             "Access varChar(1)," + _
             "Photo varBinary(Max))"
        ElseIf (comboBox1.Text = "City") Then
           SQL = "CREATE TABLE City(" + _
              "CityCode varchar(6) Primary Key," + _
              "CityName Varchar(30)," + _
              "OstanCode Varchar(6))"
        ElseIf (comboBox1.Text = "Ostan") Then
           SQL = "CREATE TABLE Ostan(" + _
              "OstanCode varchar(6) Primary Key," + _
              "OstanName Varchar(30))"
        Else
           SQL = "CREATE TABLE Shobe(" + _
             "ShobeCode varchar(6) Primary Key," + _
             "ShobeName Varchar(30))"
        End If
        Cmd = New SqlCommand(SQL, Con)
        Cmd.CommandType = CommandType.Text
        Cmd.ExecuteNonQuery()
        Con.Close()
        MessageBox.Show("ایجاد شد " + comboBox1.Text + "  جدول")
    Catch
      MessageBox.Show("خطا رخ داد " +comboBox1.Text+ "در ایجاد جدول")
    End Try
End Sub
```

این دستورات، ابتدا یک شیء از نوع SqlConnection به نام Con ایجاد می‌کنند و اگر اتصال به این شیء باز باشد، آن را می‌بندد. در ادامه شیء conStr (رشته اتصال) را مقداردهی می‌کنند (نام بانک مورد نظر را Chek، سرویس‌دهنده را SqlExpress\\. در نظر می‌گیرد) و دستورات بعدی، اتصال به بانک اطلاعاتی را باز می‌کنند. دستور if چک می‌کند که اگر کاربر از comboBox1 گزینه Bank را انتخاب کرده باشد، دستور SQL را برای ایجاد جدول Bank به متغیر SQL تخصیص می‌دهد، وگرنه اگر کاربر گزینه Chek را از comboBox1 انتخاب نماید، دستور SQLای برای ایجاد جداول Chek به متغیر SQL نسبت می‌دهد. وگرنه، چک می‌کند آیا مقدار خاصیت Text کنترل comboBox1 برابر با tblUser است، دستور ایجاد جدول tblUser را به متغیر SQL تخصیص می‌دهد. اگر مقدار خاصیت Text برابر با tblUser نباشد، با مقدار City مقایسه می‌گردد، اگر برابر این مقدار باشد، دستور ایجاد جدول City در متغیر SQL قرار می‌گیرد، وگرنه مقدار خاصیت Text کنترل comboBox1 با مقدار Ostan مقایسه می‌شود. اگر برابر این مقدار باشد، دستور ایجاد جدول Ostan در متغیر SQL قرار خواهد گرفت. در غیر این صورت، دستور ایجاد جدول Shobe در متغیر SQL قرار می‌گیرد. در پایان، شیء SqlCommand را با توجه به مقدار شیء اتصال (Con) و دستور SQL (مقدار متغیر SQL) ایجاد کرده خاصیت CommandType آن را از نوع CommandType.Text در نظر می‌گیرد و با فراخوانی متد ExecuteNonQuery() بر روی شیء cmd دستور SQL را اجرا می‌کند تا جدول مورد نظر ایجاد گردد. نهایتاً شیء اتصال (Con) را می‌بندد. اگر در فرآیند اجرای دستورات خطایی رخ دهد، بخش catch اجرا شده، پیام مناسبی را نمایش می‌دهد. به طور خلاصه می‌توان گفت، این دستورات با توجه به انتخاب کاربر جدولی را ایجاد کرده یا پیام خطای مناسبی را نمایش می‌دهند.

۸. دکمه button3 را کلیک مضاعف کرده دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button3_Click(ByVal sender As System.Object,_
    ByVal e As System.EventArgs) Handles button3.Click
  Try
    Con = New SqlConnection(conStr)
    If (Con.State = ConnectionState.Open) Then
        Con.Close()
    End If
    conStr = "Integrated Security=SSPI;Initial
      Catalog=Chek; Server=TCI;"
    Con.ConnectionString = conStr
    Con.Open()
    SQL = "DROP TABLE " + comboBox1.Text
    Cmd = New SqlCommand(SQL, Con)
    Cmd.CommandType = CommandType.Text
```

```
    Cmd.ExecuteNonQuery()
    Con.Close()
    MessageBox.Show("حذف شد " + comboBox1.Text + "  جدول")
  Catch
    MessageBox.Show("خطا رخ داد "+ comboBox1.Text + " در حذف جدول ")
  End Try
End Sub
```

این دستورات یکی از جداول Bank، Chek، tblUser، City، Shobe یا Ostan را با توجه به انتخاب کاربر در comboBox1 حذف می‌کنند و اگر خطایی در هنگام حذف جداول یا اتصال به بانک اطلاعاتی رخ دهد، پیام مناسب نمایش داده می‌شود.

۹. دکمه button4 را کلیک مضاعف کرده دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button4_Click(ByVal sender As System.Object,_
     ByVal e As System.EventArgs) Handles button4.Click
  Try
    Con = New SqlConnection(conStr)
    If (Con.State = ConnectionState.Open) Then
       Con.Close()
    End If
    conStr = "Integrated Security=SSPI;Initial
      Catalog=Chek;Server=TCI;"
    Con.ConnectionString = conStr
    Con.Open()
    If (comboBox1.Text = "Bank") Then
     SQL="ALTER TABLE Bank ADD Discription VARCHAR(250) NULL"
    ElseIf (comboBox1.Text = "Chek") Then
     SQL="ALTER TABLE Chek ALTER COLUMN Comment varChar(250)"
   ElseIf (comboBox1.Text = "Shobe") Then
     SQL = "ALTER TABLE Shobe ADD  Comment varChar(250)"
   ElseIf (comboBox1.Text = "City") Then
    SQL = "ALTER TABLE City ADD  Comment varChar(250)"
   ElseIf (comboBox1.Text = "Ostan") Then
    SQL = "ALTER TABLE Ostan ADD  Comment varChar(250)"
   Else
    SQL="ALTER  TABLE tblUser ALTER COLUMN FUser varChar(20)"
   End If
   Cmd = New SqlCommand(SQL, Con)
   Cmd.CommandType = CommandType.Text
   Cmd.ExecuteNonQuery()
   Con.Close()
   MessageBox.Show("تغییر یافت " + comboBox1.Text + "  جدول")
  Catch
   MessageBox.Show("خطا رخ داد "+comboBox1.Text+" در تغییر جدول ")
  End Try
```

```
End Sub
```

این دستورات با توجه به انتخاب کاربر ساختار جـداول City، Shobe، Ostan، Bank، Chek یـا tblUser را تغییر داده یا در صورت بروز خطا، پیام مناسبی چاپ خواهند کرد.

۱۰. دکمه button5 را کلیک مضاعف کرده دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button5_Click(ByVal sender As System.Object,_
   ByVal e As System.EventArgs) Handles button5.Click
  End
End Sub
```

این دستورات، اجرای برنامه را خاتمه می‌دهند.

۱۱. پروژه را ذخیره و اجرا کنید. با اجرای پروژه، شکل۵-۳ ظاهر می‌شود. دکمه «ایجاد بانک اطلاعـاتی» را کلیک کنید تا بانک اطلاعاتی Chek ایجاد شود. سپس، در کنترل comboBox1 جـدولی را کـه مـی‌خواهیـد ایجـاد نمایید را انتخاب کرده، دکمه "ایجاد جدول" را کلیک کنید تا جدول انتخاب شده ایجاد شود (اگر در هنگام ایجاد جدول یا بانک خطایی رخ دهد، پیام مناسب نمایش داده می‌شود). دکمه‌هـای دیگـر را امتحـان نماییـد و در پایـان دکمه «خاتمه برنامه» را کلیک کنید.

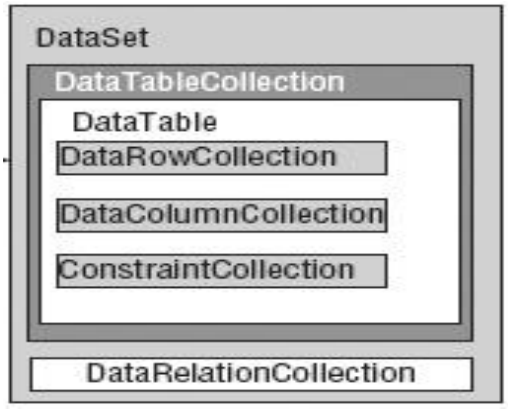
شکل ۵-۳ نمونه خروجی مثال ۱-۳.

فصل ۴ کلاس‌های پایه بانک اطلاعات

در فصل قبل، با برخی از کلاس‌های ADO.NET از قبیل SqlConnection، SqlCommand، رشته اتصال و غیره آشنا شدیم. در این فصل می‌خواهیم برخی از کلاس‌های مهم دیگر از قبیل DataSet، DataTable، DataRow و DataColumn را بی‌آموزیم.

۴-۱. کلاس DataSet

این کلاس هسته ADO.NET است. یک DataSet، از مجموعه‌ای از جداول و اطلاعاتی درباره ارتباط بین آنها تشکیل شده است و حاصل نتایج پرس‌وجو[۱۷] از بانک اطلاعات را نگهداری می‌کند (هر درخواست از بانک اطلاعات، یک **پرس‌وجو** نام دارد). یکی از ویژگی‌های جالب DataSet این است که بدون این که به منبع داده (بانک اطلاعات) متصل باشیم، می‌توانیم از آن استفاده کنیم. یعنی، پس از این که اطلاعات را از بانک اطلاعات بازیابی کردیم و در DataSet ذخیره نمودیم، می‌توانیم بدون اتصال به بانک اطلاعات از داده‌های ذخیره شده در آن استفاده کنیم. همان‌طور که در شکل ۱-٤ می‌بینید، DataSet از دو بخش DataRelationCollection و DataTableCollection تشکیل شده است.



شکل ۱-۴ اجزای DataSet.

1- Query

چون شیء DataTableCollection در بخش DataSet قرار دارد، بنابراین می‌توان چندین جدول یا نتیجه چندین پرس‌وجو را در آن قرار داد. یعنی، نتیجه هر یک از پرس‌وجوها در یک شیء DataSet نگهداری می‌شود. این اشیاء DataTable که نتیجه پرس‌وجوها یا جداول را نگهداری می‌کنند، در بخش DataTableCollection قرار می‌گیرند.

از طرف دیگر، یکی از بخش‌های DataSet، بخش DataRelationCollection است که از اشیای DataRelationCollection تشکیل شده است. اشیای DataRelation، ارتباط بین جداول را نگهداری می‌کند. باید توجه کنید که ارتباط بین جداول به طور خودکار ایجاد نمی‌شود، بلکه باید آنها را صراحتاً با استفاده از کلاس DataRelation ایجاد نمود. برخی از اجزای DataSet در زیر آمده‌اند:

۱. DataTable ۲. DataRelation ۳. DataView ٤. DataAdapter

علاوه براین‌که DataSet می‌تواند مقادیر جداول و پرس‌وجوها را نگهداری کند، داده‌های موجود در فایل XML را نیز می‌تواند بخواند یا داده‌های موجود در DataSet را می‌توان در فایل XML نوشت (ADO.NET از فایل XML برای انتقال داده بین عناصر مختلف برنامه استفاده می‌کند). هر DataSet می‌تواند حاوی یک یا چند شیء DataTable باشد که هر DataTable اطلاعات یک جدول از بانک اطلاعات را نگهداری می‌کند.

برای ایجاد ارتباط بین جداول از کلاس DataRelation استفاده می‌شود. DataView، اطلاعات موجود در DataTable را نمایش می‌دهد. کلاس DataAdapter، برای کار کردن با DataSet ضروری است. این کلاس پلی بین DataSet و منبع داده (بانک اطلاعات) برقرار می‌کند. برای بازیابی اطلاعات از بانک اطلاعات و قرار دادن در DataSet و ثبت تغییرات انجام شده در DataSet از کلاس DataAdapter استفاده می‌گردد. برای استفاده از DataSet باید شیای از آن ایجاد نمایید. برای این منظور، از سازنده DataSet که در جدول ۱-٤ آمده‌اند، استفاده کنید. خواص آن را مقداردهی کرده از متدهای آن نیز استفاده نمایید. خواص و متدهای کلاس DataSet در جداول ۲-٤ و ۳-٤ آمده‌اند.

شیء DataSet در فضای نام System.Data قرار دارد.

جدول ۱-٤ سازنده‌های DataSet.

سازنده	هدف
New()	DataSetای بدون نوع با نام پیش‌فرض NewDataSet ایجاد می‌کند.
New(dsName)	DataSetای بدون نوع با نام مشخص شده dsName ایجاد می‌کند.

جدول ۲-٤ خواص DataSet.	
خاصیت	هدف
Casesenitive	تعیین‌می‌کند که آیا درهنگام‌مقایسه‌بین‌حروف‌بزرگ یا کوچک فرق بگذارد یا خیر.
DataSetName	نام DataSet را تعیین می‌کند.
DefaultViewManager	فیلتر کردن یا مرتب‌سازی پیش‌فرض مربوط به DataSet را تعریف می‌کند.
EnforceConstrains	تعیین می‌کند که آیا در هنگام تغییر رکوردها، محدودیت‌های تعریف شده اعمال شود یا خیر.

جدول ٤-٤ متدهای Add برای اضافه کردن جدول به DataSet.	
متد	هدف
Tables.Add()	DataTable جدیدی در داخل DataSet با نام TableN ایجاد می‌کند که N یک عدد صحیحی است که از ۰ شروع می‌شود و به طور خودکار با افزودن هر جدول به DataSet، اضافه می‌شود.
Table.Add(TableName)	DataTable جدیدی با نام TableName ایجاد کرده به DataSet اضافه می‌کند.
Table.Add(DataTable)	DataTable مشخص شده را به DataSet اضافه می‌کند.
Tables.AddRange(TableArray)	DataTableهایی را که در TableArray قرار دارند به DataSet اضافه می‌کند.

۱-۱-۴. پر کردن DataSet

داده‌های DataSet را می‌توانید با کلاس DataAdapter پر کنید (با کلاس DataAdapter در فصول بعد بیشتر آشنا خواهید شد). برای این منظور باید مراحل زیر را انجام دهید:

۱. شیء اتصال به بانک را تعریف کنید:

```
Dim connStr As String = "رشته اتصال"
Dim conn As SqlConnection = New SqlConnection(connStr)
```

۲. دستور یا دستورات SQLای را تعریف کنید که می‌خواهید داده‌ها را از بانک اطلاعات بازیابی کرده، در DataSet قرار دهید. (مانند دستور زیر):

```
Dim SQL As String = "SELECT * FROM Chek"
```

۳. نمونه‌ای از کلاس SqlDataAdapter تعریف کنید:

```
Dim da As SqlDataAdapter = New SqlDataAdapter (SQL , Conn)
```

٤. شیء DataSet را ایجاد کنید:

```
Dim ds As DataSet = New DataSet()
```

۵. متد Fill را روی نمونه شیء SqlDataAdapter اجرا نمایید تا DataSet را پر کنید. این دستور به صورت زیر به کار می‌رود:

```
da.Fill(ds, "Chek")
```

۲-۴-۱. پر کردن DataSet با چندین جدول

همان‌طور بیان گردید، برای پر کردن DataSet می‌توانید از کلاس DataAdapter استفاده کنید. کلاس DataAdapter دارای کلاسی به نام TableMappings است که با استفاده از آن می‌توانید DataSet را با چندین جدول یا نتیجه پرس‌وجو پر کنید. دستورات زیر را ببینید:

```
Dim SQL As String = "SELECT * FROM Ostan;"
SQL += "SELECT * FROM City;"
SQL += "SELECT * FROM Bank;"
Dim da As SqlDataAdapter = New SqlDataAdapter(SQL, conn)
da.TableMappings.Add("Table" , "Ostan")
da.TableMappings.Add("Table1", "City")
da.TableMappings.Add("Table2" , "Bank")
Dim ds As DataSet = New DataSet()
da.Fill(ds)
```

این دستورات، جدول Ostan، City و Bank را به DataSet اضافه می‌کنند.

۳-۱-۴. تثبیت تغییرات DataSet

پس از این که اطلاعات را از بانک اطلاعات بازیابی کردید و در DataSet قرار داده شد، اطلاعات DataSet را می‌توان بدون این که به بانک اطلاعات متصل باشد، ویرایش کرد. بنابراین باید بتوان تغییرات اعمال شده را به بانک اطلاعات برگرداند. برای این منظور می‌توان از متد ()Update استفاده نمود.

۴-۱-۱. دسترسی به جداول DataSet

بیان گردید که DataSet می‌تواند مجموعه‌ای از جداول را نگهداری کند. برای دسترسی به جداول DataSet می‌توان از خاصیت Table استفاده کرد. به عنوان مثال، دستورات زیر را ببینید:

```
ds.Table(0)
ds.Table(4)
```

دستور اول، اولین جدول DataSet و دستور دوم، پنجمین جدول DataSet را بازیابی می‌کند. زیرا، اندیس جداول DataSet از صفر شروع می‌شوند.

۴-۲. کلاس DataTable

این کلاس یکی از مهم‌ترین اشیاء ADO.NET را تعریف می‌کند که برای ذخیره‌سازی و کار بر روی داده‌ها به کار می‌رود. شیء DataTable در حافظه قرار گرفته داده‌ها را به صورت مجموعه از سطرها و ستون‌ها ذخیره می‌نماید. ستون‌ها، همان فیلدها هستند که نام، نوع و اندازه دارند و سطرها همان رکوردهای جداول هستند.

همان‌طور که دیدید، کلاس DataTable در کلاس DataSet قرار می‌گیرد و هر بار که یک جدول از داده‌ها به DataSet اضافه می‌شود، در یک شیء DataTable قرار می‌گیرد. کلاس DataTable از مجموعه‌ای از DataRow (درخاصیت Rows) تشکیل شده است که این مجموعه حاوی داده‌های جدول هستند و هر DataRow از مجموعه‌ای از DataColumn (در خاصیت Columns) تشکیل شده است که این مجموعه حاوی اطلاعاتی راجع به فیلدهای جدول می‌باشند. در ادامه شرح اشیاء DataRow و DataColumn را می‌بینید. خواص و متدهای شیء DataTable در جداول ٥-٤ و ٦-٤ آمده‌اند و رویدادهای DataTable را در جدول ٧-٤ می‌بینید. برای ایجاد شیء DataTable از متد سازنده آن استفاده می‌گردد. انواع متدهای سازنده DataTable در زیر آمده‌اند:

- **سازنده New()**: یک جدول جدید ایجاد می‌کند.

- **سازنده New(TableName)**: یک جدول با نام TableName ایجاد می‌کند.
- **سازنده New(SerializableInfo, StreamContext)**: برای ایجاد جدول از فریم ورک درون NET. استفاده می‌کند. اکنون دستورات زیر را ببینید:

```
1. Dim dt1 As DataTable = New DataTable()
2. Dim dt2 As DataTable = New DataTable("Chek")
```

دستور اول، یک DataTable به نام dt1 ایجاد می‌کند و دستور دوم، جدولی به نام dt2 ایجاد کرده، جدول Chek را به آن تخصیص می‌دهد.

کلاس DataTable در فضای نام System.Data قرار دارد.

جدول ٥-٤ خواص شیء DataTable.

خاصیت	هدف
Casesensitive	چگونگی مقایسه رشته را تعیین می‌کند. یعنی، تعیین می‌کند آیا بین حروف بزرگ و کوچک فرق قائل شود یا خیر.
ChildRelations	در مجموعه‌ای (کلکسیونی) از اشیاء DataRelation که این DataTable در آن وجود دارد، DataTable مذکور به عنوان جدول پدر است.
Columns	مجموعه ستون‌های (فیلدهای) جدول را مشخص می‌کند.
Constraint	مجموعه‌ای از محدودیت‌های (Constrations) است که برای این جدول تعریف شده‌اند.
DataSet	اگر در برنامه چند DataSet تعریف شده باشد، تعیین می‌کند این جدول متعلق به کدام DataSet است.
DisplayExpression	عبارتی است که برای نمایش نام جدول در واسط کاربر (UI) استفاده می‌شود.
HasErrors	تعیین می‌کند آیا خطایی در سطرهای جدول وجود دارد یا خیر.
ParentRelations	مجموعه‌ای از اشیا DataRelation که این جدول دارد و این جدول، جدول فرزند آنها می‌باشد.
PrimaryKey	مجموعه‌ای(آرایه‌ای) از ستون‌ها هستند که کلید اولیه جدول را تعریف می‌کنند.
Rows	مجموعه سطرهای جدول را مشخص می‌کند. یعنی، کلکسیونی از اشیاء Rows

	می‌باشد که رکوردهای جدول را نگهداری می‌کنند.
TableName	نام جدول (DataTable) را در DataSet تعیین می‌کند.

فصل ۵ ورود و ویرایش داده‌ها

پس از این که جدول‌های بانک اطلاعاتی ایجاد گردید باید بتوان داده‌ها را در آنها اضافه نمود، داده‌های وارد شده را ویرایش یا حذف کرد. در این فصل می‌خواهیم به این مفاهیم بپردازیم. بنابراین، در این فصل مباحث زیر را می‌آموزیم:

۱. دستورات SQL برای ورود، ویرایش و حذف داده‌ها.

۲. کلاس‌های ADO.NET برای ورود، ویرایش و حذف داده‌های جدول.

۵-۱. دستورات SQL برای ورود، ویرایش و حذف داده‌ها

در فصل‌های قبل، با یک مثال ساده نحوه ایجاد بانک اطلاعات و جداول آن را با استفاده از دستور SQL آموختیم. همان طور که بیان گردید، بعد از اضافه کردن جداول به بانک اطلاعات باید بتوان داده‌ها را در آن وارد نمود، را ویرایش کرد و یا داده‌های اضافی را حذف نمود. بنابراین، در این بخش دستورات INSERT، UPDATE و DELETE که برای دستکاری داده‌های جدول به کار می‌روند را می‌آموزیم.

۵-۱-۱. دستور INSERT

این دستور برای اضافه کردن رکورد به جداول بانک اطلاعات به کار می‌رود و به صورت زیر استفاده می‌شود:

```
INSERT [INTO]   {table_name|view_name}
 {[(column_list)] {VALUES({DEFAULT|NULL|expression} [,…]
 |derived_table
 |execute_statement
 } }
 |DEFAULT VALUES
```

پارامترهای این دستور در زیر آمده‌اند:

- **table_name**، نام جدولی را تعیین می‌کند که اطلاعات باید وارد آن شود.
- **view_name**، نام دیدی را تعیین می‌کند که اطلاعات باید وارد آن شود.

- **column_list**، لیست فیلدهایی را تعیین می‌کند که اطلاعات بخش VALUES باید وارد آنها شود.
- **default**، مقدار پیش‌فرض فیلد جدول یا دید را مشخص می‌نماید. اگر چنانچه مقدار فیلد از طریق دستور INSERT وارد نگردد، این مقدار در فیلد قرار می‌گیرد.
- **null**، مقدار تهی را وارد فیلد جدول یا دید می‌نماید.
- **expression**، حاصل یک عبارت را وارد فیلد جدول یا دید می‌نماید (برای مقدار دادن به فیلدهای محاسباتی مفید است).
- **derived_table**، خروجی یک دستور SELECT را وارد جدول می‌نماید. دستور SELECT را در ادامه می‌آموزیم.
- **execute_statement**، خروجی یک دستور SELECT یا READTEXT را به جدول یا دید اضافه می‌کند.

به عنوان مثال، دستورات زیر را ببینید:

```
INSERT INTO City    VALUES ('01','بابل','02')

INSERT INTO City (CityCode, CityName, OstanCode) VALUES ('02','ساری','03')
```

این دستورات، دو رکورد به جدول شهر (City) اضافه می‌کنند.

در هنگام اضافه کردن رکورد با دستور INSERT باید نکات زیر را رعایت کنید:

۱. باید نوع داده‌ها نظیر به نظیر با نوع فیلدها، یکی باشد و طول مقادیر باید کوچک‌تر یا مساوی طول فیلدها باشد.

۲. برای فیلدهایی که کلید اولیه هستند، مقدار تکراری وارد نکنید. اگر کاربر مقادیر تکراری برای این فیلدها وارد کند، نه تنها رکورد اضافه نمی‌شود، بلکه پیام خطا ظاهر خواهد شد.

۳. برای فیلدهایی که محدودیت NOT NULL دارند، نمی‌توانید مقادیر تهی وارد کنید. اگر مقدار تهی برای این فیلدها وارد کنید، پیام خطا ظاهر می‌شود.

٤. اگر دو جدول دارای ارتباط باشند، در موقع اضافه کردن مقدار فیلد ارتباط در جدول فرزند، چنانچه این مقدار برای فیلد ارتباط در جدول پدر (همان جدولی که کلید اولیه در آن تعریف شده است)، وارد نشده باشد، پیام خطا ظاهر می‌شود.

۲-۱-۵. ویرایش رکوردهای جدول

برای ویرایش رکوردهای جدول می‌توانید از دستور UPDATE به صورت زیر استفاده کنید:

```
UPDATE table_name
[TOP(exp) [PERCENT}
SET field_list=value_list|exp_list|NULL
WHERE condition
```

در این دستور، پارامتر table_name، جدولی را تعیین می‌کند که می‌خواهید رکوردهای آن را تغییر دهید. گزینه TOP تعیین می‌کند، چند رکورد اول تغییر کند و تعداد رکوردها با exp تعیین می‌شود. گزینه PERCENT تعیین می‌کند، چند درصد رکوردهای جدول تغییر یابند. field_list، لیست فیلدهایی را تعیین می‌کند که مقادیر آنها باید به یکی از مقادیر value_list، exp_list یا NULL تغییر کند و condition، شرطی را تعیین می‌کند که رکوردهای دارای آن شرط باید مقدار آنها تغییر کند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
UPDATE  City
SET CityName='آمل'
Where  CityCode='02'
```

این دستور، نام شهری که کد شهر آن برابر **02** باشد را به **آمل** تغییر می‌دهد.

در هنگام استفاده از دستور UPDATE به نکات زیر توجه کنید:

۱. اگر در دستور UPDATE بخش WHERE ذکر نگردد، کلیه رکوردهای جدول تغییر می‌یابند. بنابراین، در هنگام استفاده از دستور UPDATE باید نهایت دقت را داشته باشید تا ناخواسته داده‌ها را تغییر ندهید.

۲. اگر با دستور UPDATE مقدار فیلد کلید اولیه یا فیلد یکتا را طوری تغییر دهید که مقدار تکراری برای آن فیلد ایجاد شود، پیام خطا ظاهر خواهد شد. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
UPDATE  City
SET CityCode='01'
Where CityName='آمل'
```

این دستور، موجب نمایش خطا خواهد شد. زیرا، مقدار کد شهر01 برای شهر **بابل** وجود دارد و نمی‌تواند به شهر **آمل** تخصیص یابد. چون کلید اولیه جدول City، فیلد CityCode است.

۳. اگر جدولی با جدول دیگر ارتباط داشته باشد و بخواهیم در جدول دوم، مقدار فیلدی که ارتباط بین دو جدول را برقرار می‌کند، تغییر دهیم (چنانچه این فیلد در جدول پدر دارای همان مقدار باشد که فعلاً در جدول فرزند دارد)، پیام خطا ظاهر خواهد شد.

٤. اگر دو جدول با هم ارتباط داشته باشند و بخواهید مقدار فیلد ارتباط را در جدول اولیه طوری تغییر دهید که این مقدار در جدول دوم موجود نباشد، پیام خطایی ظاهر می‌شود.

۳-۱-۵. حذف رکوردهای جدول

برای حذف رکوردهای اضافی جدول می‌توانید از دستور DELETE به صورت زیر استفاده کنید:

```
DELETE [TOP(exp) [PERCENT]]
FROM table_name
Where condition
```

با این پارامترها در دستور UPDATE آشنا شدید. به عنوان مثال، دستور زیر را ببینید:

```
DELETE  FROM City
Where  CityCode='02'
```

این دستور رکوردی از جدول City (شهر) را حذف می‌کند که کد شهر آن برابر '02' باشد.

در هنگام استفاده از دستور DELETE باید به نکات زیر توجه داشته باشید:

۱. اگر دستور DELETE را بدون شرط استفاده کنید، کلیه رکوردهای جدول حذف خواهند شد.

۲. با دستور DELETE نمی‌توانید جدول را حذف کنید. بلکه، فقط می‌توانید رکوردهای جدول را حذف نمایید.

۳. اگر دو جدول با هم ارتباط داشته باشند و بخواهیم رکوردی را از جدول پدر حذف کنیم که مقدار فیلد ارتباط در جدول فرزند موجود باشد، پیام خطا ظاهر می‌شود (رکورد حذف نمی‌گردد).

۲-۵. انقیاد داده‌ها

کنترل DataGridView بهترین کنترل برای نمایش داده‌ها در فرم برنامه می‌باشد. این کنترل علاوه بر قابلیت نمایش، به راحتی امکان ویرایش، حذف و اضافه کردن داده‌های جدید را برای جداول فراهم می‌نماید. امّا، گاهی اوقات نیاز است که در هر لحظه فقط یک رکورد را بر روی فرم نمایش دهید و بتوانید رکوردهای دیگر را جستجو، پیمایش، ویرایش و حذف نمایید. در این موارد باید کنترل‌هایی از قبیل TextBox، ListBox،

ComboBox و غیره را به فرم اضافه کرده و فیلدهای جدول را به آنها **مقید**[۱۸] کنید. برای مقید کردن فیلدها به کنترل‌های روی فرم دو روش وجود دارد که عبارتند از:

۱. مقید کردن در زمان طراحی **۲**. مقید کردن در زمان اجرا

در این کتاب سعی شده است که انقیاد زمان اجرا مورد بررسی قرار گیرد. برای انقیاد فیلدها به کنترل‌ها از شیء BindingContext فرم استفاده می‌شود. شیء BindingContext از مجموعه‌ای از اشیاء CurrencyManager تشکیل شده است که هر شیء CurrencyManager، بین منابع داده مانند DataSet و کنترل‌های دیگری که به همین منبع داده‌ای متصل می‌باشند، هماهنگی برقرار می‌کند (یعنی، شیء CurrencyManager می‌تواند هماهنگی لازم را بین کنترل‌ها و منابع داده‌ای مختلفی از قبیل DataSet، DataView، DataTable و غیره ایجاد کند). هرگاه یک منبع داده جدیدی به فرم اضافه می‌کنید، به طور خودکار یک شیء CurrencyManager جدید نیز اضافه می‌گردد. به عنوان مثال، دستورات زیر را ببینید:

```
Dim CurrManager As CurrencyManager
CurrManager=(CurrencyManager)(this.BindingContext(ds))
```

دستور اول، یک شیء به نام CurrManager از نوع CurrencyManager ایجاد می‌کند و دستور دوم، مقدار این شیء ایجاد شده را برابر با منبع داده‌ای ds (از نوع DataSet که قبلاً تعریف شده است)، قرار می‌دهد.

پیمایش رکوردها با استفاده از شیء CurrencyManager

بعد از این که شیء CurrencyManager را ایجاد کردید و آن را به یک منبع داده تخصیص داده‌اید، می‌توانید بین رکوردهای منبع داده حرکت کنید. برای نمایش و پیمایش رکوردهای منبع داده با استفاده از شیء CurrencyManager می‌توانید از خاصیت Position این شیء استفاده کنید. خاصیت Position این شیء رکورد فعلی را نمایش می‌دهد که می‌توانید اعمال زیر را از طریق آن انجام دهید:

❖ **حرکت به رکورد بعدی**، برای حرکت به رکورد بعدی باید خاصیت Position شیء CurrencyManager را یک واحد اضافه کنید. به عنوان مثال، دستور زیر را ببینید:

```
CurrManager.Position += 1
```

این دستور، رکورد بعدی را در تمام کنترل‌های ساده (مانند TextBox، CheckBox و ComboBox) که به DataSet مقید شده‌اند، نمایش می‌دهد.

1- Bind

فصل ۷ اشیای پیشرفته SQL Server و ADO.NET

در فصل‌های قبل با بانک اطلاعات آشنا شدیم و توانستیم در بانک اطلاعات جدول ایجاد کرده، از طریق ADO.NET رکوردهایی به آنها اضافه نماییم و رکوردهای موجود را ویرایش، حذف یا بازیابی کنیم. در این فصل می‌خواهیم به اشیای دیگر بانک اطلاعات از قبیل دیدها[۱۹]، توابع[۲۰] و رویه‌های ذخیره شده[۲۱] بپردازیم و چگونگی استفاده از آنها را در ADO.NET ببینیم. در ضمن در این فصل با تعریف متغیرها، مقداردهی به آنها و ساختارهای تصمیم و تکرار در SQL Server آشنا خواهیم شد.

۱-۷. دید

همان‌طور که در فصل‌های قبل مشاهده کردید، دستور SELECT، برای بازیابی اطلاعات از بانک اطلاعات به کار می‌رود. یکی از معایب این دستور این است که در بانک اطلاعات به عنوان یک شیء ذخیره نمی‌شود. یعنی، اگر بخواهید یک دستور SELECT را چندین بار اجرا نمایید باید آن را چندین بار تایپ نمایید و سپس اجرا کنید. این امر موجب صرف وقت زیادی خواهد شد. برای رفع این مشکل، می‌توانید از دید استفاده کنید. چون، دید به عنوان یک شیء در بانک اطلاعات ذخیره می‌گردد.

۱-۱-۷. ایجاد دید

دید نه تنها در بانک اطلاعات ذخیره می‌شود، بلکه امکانات امنیتی را برای کاربران فراهم می‌نماید تا کاربران بتوانند به فیلدها و رکوردهای خاص خودشان دستیابی داشته باشند. دید، جدول مجازی[۲۲] با قالب‌های جدید ایجاد می‌کند. در جدول Chek، چک مربوط به بانک‌های مختلف نگهداری می‌شود. فرض کنید رئیس بانک بخواهد چک‌های مربوط به بانک خودش را ببیند. در حالت عادی چک‌های مربوط به کل بانک‌ها را مشاهده می‌کند. بنابراین، رئیس بانک باید با دستور SELECT که دارای شرط خاصی است، اطلاعات و رکوردهای بانک خودش را بازیابی نماید. اگر تعداد دفعات اجرای این دستور زیاد باشد (به دفعات زیاد بخواهد این کار را انجام

1- Views **2**- Functions **3**- Stored Procedures **4**- Virtual table

دهد)، تایپ دستور نه تنها کسل کننده است، بلکه زمان‌بر نیز خواهد بود. برای رفع این مشکل می‌توانید برای هر یک از کاربران نامبرده (مانند رواسای بانک‌ها) دیدی ایجاد کنید تا این کاربران فقط بتوانند اطلاعات خودشان را بازیابی کنند. از طرف دیگر، چون در دید کاربران به طور مستقیم به جداول بانک اطلاعات دسترسی ندارند، امنیت افزایش می‌یابد. برای ایجاد دید از دستور CREATE VIEW به صورت زیر استفاده می‌شود:

```
CREATE VIEW view_name(fields_list) As SQL دستور
```

در این ساختار، view_name، نام دیدی است که می‌خواهید ایجاد کنید، fields_list، لیست فیلدهای دید را تعیین می‌کند. به عنوان مثال، دستور زیر را ببینید:

```
CREATE  VIEW  Chek1  As
SELECT  *  FROM  Chek   WHERE BankCode = '87'
```

این دستور، دیدی به نام Chek1 ایجاد می‌کند که چک‌های بانک با کد '87' را بازیابی می‌کند.

اکنون کاربر می‌تواند با اجرای دستور زیر چک‌های بانک با کد '87' را ببیند:

```
SELECT  *   FROM  Chek1
```

در هنگام ایجاد دید باید به نکات زیر دقت کنید:

۱. همانند جدول و اشیای دیگر، اگر دیدی را قبلاً ایجاد کرده باشید و بخواهید مجدداً آن را ایجاد نمایید، با پیام خطا مواجه خواهید شد.

۲. در هنگام ایجاد دید نمی‌توانید از عملگر UNION استفاده کنید.

۳. در هنگام ایجاد دید، نمی‌توانید از گزینه ORDER BY در دستور SELECT استفاده کنید. به جای آن می‌توانید از گزینه GROUP BY استفاده نمایید.

٤. در دیدهایی که از ترکیب چند جدول ایجاد شده‌اند، کاربران نمی‌توانند با دستورات UPDATE، INSERT و DELETE جداول پایه را از طریق دیدهای مذکور دستکاری کنند.

۵. در دیدهایی که از عملگر DISTINCT در دستور SELECT استفاده کردید، نمی‌توانید با دستورات INSERT، UPDATE و DELETE جداول پایه آن دیدها را دستکاری کنید.

۲-۱-۷. تغییر دید

هر شیء ایجاد شده را باید بتوانید تغییر دهید. زیرا، اگر شیء مانند دید را قبلاً ایجاد کرده باشید و بخواهید آن را مجدداً ایجاد نمایید، با پیام خطا از طرف SQL Server مواجه خواهید شد. برای رفع این مشکل می‌توانید از دستور ALTER استفاده کنید. برای تغییر دید می‌توانید از دستور ALTER VIEW به صورت زیر استفاده کنید:

دستور ALTER VIEW view_name As SQL

این دستور دید موجود به نام view_name را تغییر می‌دهد. به عنوان مثال، دستور زیر را ببینید:

```
ALTER  VIEW  Chek1    As
 SELECT ChekNo, BankName, SodorDate, Mablag   FROM  Chek c, Bank b
 WHERE c.BankCode = b.BankCode
```

این دستور کد چک، نام بانک، تاریخ صدور و مبلغ چک‌ها را نمایش می‌دهد.

۳-۱-۷. حذف دید

دید از اشیا بانک اطلاعات می‌باشد که فضای بانک اطلاعات اشغال را می‌کند. بنابراین، باید بتوانید دیدهای اضافی را حذف کنید. برای این منظور می‌توانید از دستور DROPVIEW به صورت زیر استفاده کنید:

DROP VIEW view_names

در این دستور view_names، لیست دیدهایی را تعیین می‌کند که می‌خواهید حذف شوند. اگر تعداد دیدهایی که می‌خواهید حذف کنید، بیش از یکی باشد، می‌توانید نام آن‌ها را با کاما از هم جدا کنید. به عنوان مثال، دستور زیر را ببینید:

```
DROP VIEW Chek1
```

این دستور دید Chek1 را از بانک اطلاعات حذف می‌کند.

۲-۷. متغیرها

همان‌طور که بیان گردید، متغیرها، نام‌هایی برای محل‌های از حافظه هستند که در طول اجرای برنامه محتویات آنها تغییر می‌کند. برای استفاده از متغیرها باید آنها را تعریف کرده و محتویات آنها را تغییر دهید.

برای تعریف متغیر در SQL Server می‌توانید از دستور DECLARE به صورت زیر استفاده کنید:

```
DECLARE @ نام متغیر    نوع متغیر
```

به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
DECLARE @SanadCode Decimal(18,0)
DECLARE @BankName Varchar(50)
```

دستور اول، متغیر SanadCode@ را از نوع Decimal و دستور دوم متغیر BankName@ را از نوع Varchar تعریف می‌کند.

اگر متغیری را تعریف کرده به آن مقدار ندهید، به طور پیش‌فرض به آن مقدار NULL تخصیص می‌یابد. برای مقداردهی به متغیرها می‌توانید از دستور SET به صورت زیر استفاده نمایید:

```
SET @نام متغیر = مقدار
```

به عنوان مثال، دستورات زیر را مشاهده کنید:

```
SET @ChekID = 100
SET @BankName = 'ملی مرکزی'
```

دستور اول، مقدار ۱۰۰ را متغیر ChekID@ تخصیص می‌دهد و دستور دوم، به متغیر BankName@ مقدار 'ملی مرکزی' را نسبت می‌دهد. همان‌طور که در نامگذاری متغیرها دیدید، متغیرهایی که برنامه‌نویس تعریف می‌کند، با علامت @ شروع می‌شوند. برخی از متغیرهای دیگر وجود دارند که با دو کارکتر @ (@@) شروع می‌شوند. این متغیرها، متغیرهای سیستمی SQL Server هستند. نمونه‌ای از این متغیرها @@rowcount است. این متغیر تعداد سطرهای (رکوردهای) جدول را نگهداری می‌کند.

۳-۷. توضیحات

توضیحات[۲۳]، یکی از بخش‌های بسیار مهم در برنامه است. زیرا، موجب افزایش خوانایی برنامه می‌شود. فرض کنید برای اداره، مؤسسه یا سازمانی برنامه‌ای را نوشته‌اید و واحد پشتیبانی آن سازمان بخواهد برنامه نوشته شده‌تان را پشتیبانی کند. توضیحات به واحد پشتیبانی کمک می‌کند که برنامه نوشته شده‌تان را راحت‌تر درک کند. به دو روش می‌توانید توضیحات را در SQL Server تعریف کنید که عبارت‌اند از:

۱) -- توضیحات

۲) /* توضیحات */

1- Comments **2-** True **3-** False

روش اول، برای ایجاد توضیحات یک سطری به کار می‌رود. ولی، با روش دوم می‌توانید توضیحات چند سطری را ایجاد نمایید. دستورات زیر را در نظر بگیرید:

```
-- نام بانک تکراری است
/* شروع رویه ذخیره شده
Insert_chek */
```

این دستورات دو توضیح به برنامه اضافه می‌کند که اولی یک سطری، ولی دومی دو سطری است.

۴-۷. ساختارهای تصمیم

در SQL Server می‌توانید تصمیم بگیرید با توجه به شرط خاصی مجموعه‌ای از دستورات اجرا گردند یا مجموعه دیگری از دستورات اجرا نشوند. برای این منظور، می‌توانید از ساختارهای تصمیم SQL Server استفاده کنید. برخی از ساختارهای تصمیم SQL Server عبارت‌اند از:

۱. دستور If … Else

۲. دستور IF تو در تو

۳. دستور Case

۱-۴-۷. دستور IF … Else

این دستور، ابتدا شرطی را تست می‌کند و براساس نتیجه شرط (درستی[۲۴] یا نادرستی[۲۵]) تصمیم می‌گیرد کدام بخش از برنامه اجرا شود. ساختار این دستور به صورت زیر است:

```
If   شرط
     دستور ۱
Else
     دستور ۲
```

در این ساختار، اگر شرط درست باشد، **دستور۱**، وگرنه **دستور۲** اجرا می‌شود. اگر بخواهیم به جای دستور۱ و دستور۲، مجموعه‌ای از دستورات را قرار دهیم، این دستورات را بین BEGIN و END قرار می‌دهیم. به عنوان مثال، دستور زیر را ببینید:

```
if @BankCode IN (SELECT BankCode FROM Bank)
```

```
   SET @Strout = '1' -- کد بانک تکراری است
else    if @BankName IN (SELECT BankName FROM Bank)
   SET @Strout = '2' -- نام بانک تکراری است
```

این دستورات، تعیین می کنند آیا کد یا نام بانک در جدول Bank تکراری است. اگر کد بانک تکراری باشد، مقدار Strout@ را برابر '1' قرار می‌دهد. ولی، اگر نام بانک تکراری باشد، متغیر Strout@ برابر '2' خواهد شد.

۲-۴-۷. دستور IF تو در تو

گاهی نیاز است شرط‌های متعددی را تست کنید. برای این منظور، می‌توانید از IF تو در تو استفاده نمایید:

```
IF     شرط۱
     دستور۱
Else IF شرط۲
      دستور۲
Else IF شرط۳
      دستور۳
      ...
Else IF شرطn
       دستور n
Else
      دستورE
```

در این ساختار، ابتدا **شرط۱** ارزیابی می‌شود، اگر این شرط درست باشد، **دستور۱** اجرا می‌شود و IF تو در تو خاتمه می‌یابد. وگرنه، اگر **شرط۲** درست باشد، **دستور۲** اجرا می‌گردد. در غیر این صورت، این روند ادامه می‌یابد. اگر هیچ یک از شرط‌های ۱ تا n درست نباشد، **دستورe** اجرا خواهد شد. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
If @BankCode IN (SELECT BankCode FROM Bank)
   SET @Strout = '1' -- کد بانک تکراری است
else  if @BankName IN (SELECT BankName FROM Bank)
   SET @Strout = '2' -- نام بانک تکراری است
else
   SET @Strout = '0'
```

این دستور، در جدول Bank جستجو می‌کند. اگر کد بانک تکراری باشد، مقدار '1' را به متغیر Strout@ تخصیص می‌دهد، اگر نام بانک تکراری باشد، مقدار '2' را در متغیر Strout@، وگرنه مقدار '0' را به آن تخصیص خواهد داد.

۵-۷. رویه‌های ذخیره شده

رویه‌های ذخیره شده، تعدادی از دستورات SQL هستند که با هم در یک مجموعه قرار گرفته کامپایل می‌شوند و با یک دستور SQL قابل اجرا می‌باشند. رویه‌های ذخیره شده مزایای متعددی دارند که برخی از آنها عبارت‌اند از:

۱.سرعت اجرای رویه‌های ذخیره شده بالاست. زیرا، رویه‌های ذخیره شده به صورت کامپایل شده در حافظه نهان[۲۶] بانک اطلاعات نگهداری می‌شوند. بنابراین، دستیابی به آنها سریع‌تر انجام خواهد شد و از طرف دیگر، چون کامپایل شده‌اند، در هنگام اجرا نیاز به کامپایل شدن ندارند. پس، سرعت اجرای آنها افزایش می‌یابد.

۲.رویه‌های ذخیره شده برنامه‌نویسی ماژولی[۲۷] را امکان‌پذیر می‌سازند. زیرا، یک بار رویه‌های ذخیره شده را می‌نویسید، کامپایل می‌کنید و چند بار آنها را فراخوانی می‌نمایید تا اجرا شوند.

۳.رویه‌های ذخیره‌شده ترافیک شبکه را کاهش می‌دهند. زیرا، ممکن است رویه‌های ذخیره شده از چندین سطر تشکیل شوند. در حالت عادی، وقتی سرویس‌گیرنده[۲۸] چند خط را به عنوان پرس‌وجو برای سرویس‌دهنده ارسال می‌کند، ترافیک شبکه افزایش می‌یابد. در حالتی که تعداد سرویس‌گیرنده‌ها (کاربران) در محیط شبکه زیاد باشد، این موضوع بیشتر خودش را نشان می‌دهد. ولی، اگر از رویه‌های ذخیره استفاده کنید، جهت اجرا فقط کافی است یک خط را بفرستید تا رویه ذخیره شده اجرا گردد. چون بدنه رویه ذخیره شده در سمت سرویس‌دهنده[۲۹] قرار دارد.

٤.رویه‌های ذخیره شده را می‌توان به طور خودکار پس از راه‌اندازی SQL Server اجرا نمود.

نام رویه‌های ذخیره شده در جدول sys.objects و دستورات آن در جدول sys.Comments ذخیره می‌شوند.

1- Cache Memory **2-** Modular Programming **3 -** Client

1. Server

۱–۵–۷. ایجاد رویه‌های ذخیره شده

رویه‌های ذخیره شده به عنوان یک شیء در بانک اطلاعات ذخیره می‌شوند که می‌توانید با دستور CREATE PROCEDURE آنها را ایجاد کنید. این دستور به صورت زیر به کار می‌رود:

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
 VARYING ] [ = default ] [ OUTPUT ]
 ] [ ,...n ]
[ WITH  { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]
[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

پارامترهای این دستور در جدول ۱–۷ آمده‌اند. به عنوان مثال، دستورات زیر ار ببینید:

```
USE [Chek]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
CREATE PROCEDURE [dbo].[delete_Shobe]
 (@ShobeCodeVarchar(6),
  @strOut varchar(1) output)
AS
 if @ShobeCode in (select ShobeCode from Bank)
   set @strOut='1' -- شماره شعبه در بانک استفاده گردید
   else if @ShobeCode not in (select ShobeCode from Shobe)
      set @strOut='2' -- شماره شعبه در بانک استفاده گردید
   else
   begin
     DELETE [Shobe]   WHERE ( [ShobeCode] = @ShobeCode)
     set @strOut='0' -- بانک حذف گردید
   end
```

این دستورات، رویه ذخیره‌شده delete_Shobe را ایجاد می‌کنند که پارامتر ShobeCode@ را از ورودی گرفته پارامتر Strout@ را به صورت خروجی برمی‌گرداند. این رویه ذخیره شده، اگر کد شعبه در بانک وجود داشته باشد، Strout@ را برابر با '1' وگرنه، اگر کد شعبه وجود نداشته باشد، Strout@ برابر '2' وگرنه، شعبه را حذف می‌کند و مقدار Strout@ برابر '0' خواهد شد. اکنون دستور زیر را در نظر بگیرید:

```
USE [Chek]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER OFF
GO
CREATE PROCEDURE [dbo].[delete_Shobe]
 (@ShobeCodeVarchar(6),QShobename varchar(30),
  @strOut varchar(1) output)
AS
 if @ShobeCode in (select ShobeCode from Shobe)
   set @strOut='1' -- شماره شعبه در بانک استفاده گردید
 ElsE
 begin
   DELETE [Shobe]    VALUES (QShobeCode, QShobename)
   INSERT INTO
   set @strOut='0' -- شعبه اضافه گردید
end
```

فصل ۱۰ پشتیبان‌گیری و بازیابی پشتیبان از بانک اطلاعات پشتیبان‌گیری و بازیابی پشتیبان از بانک اطلاعات

یکی از بخش‌های مهم نرم‌افزار حفظ و نگهداری اطلاعات است. زیرا ممکن است به دلایل مختلف اطلاعات را از دست بدهیم که برخی از این دلایل عبارتند از:

۱. دیسک سخت توسط افراد غیرمجاز مورد دستیابی قرار گیرد و آن‌ها اعمالی مثل فرمت یا حذف اطلاعات را انجام دهند.

۲. دیسک سخت خراب شود و قابل بازیابی نباشد.

۳. اطلاعات قبلی بر روی اطلاعات فعلی کپی گردد (اشتباهاً).

٤. ویروس‌های کامپیوتری، اطلاعات را تخریب کنند.

بنابراین در این فصل به پشتیبان‌گیری و بازیابی اطلاعات از بانک اطلاعات می‌پردازیم. در این فصل دو روش پشتیبان‌گیری و بازیابی اطلاعات را با دو مثال می‌آموزیم. این دو روش عبارت‌اند از:

۱. استفاده از دستورات Backup و Restore

۲. استفاده از کنترل SQLDMO

۱۰-۱. پشتیبان‌گیری با دستور BACKUP DATABASE

روش‌های متعددی برای پشتیبان‌گیری از بانک اطلاعات وجود دارد. کامل‌ترین روش پشتیبان‌گیری استفاده از دستور BACKUP DATABASE است. زیرا، این دستور، از بانک اطلاعات، فایل‌های کارنامه[۳۰] و Filegroupها پشتیبان‌گیری می‌نماید. این دستور، به صورت زیر به کار می‌رود:

```
BACKUP DATABASE { database_name | @database_name_var }
 TO < backup_device > [ ,...n ]
 [ [ MIRROR TO < backup_device > [ ,...n ] ] [ ...next-mirror ] ]
```

1- Log

```
[ WITH
[ BLOCKSIZE = { blocksize | @blocksize_variable } ]
[ [ , ] { CHECKSUM | NO_CHECKSUM } ]
[ [ , ] { STOP_ON_ERROR | CONTINUE_AFTER_ERROR } ]
[ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] DIFFERENTIAL ]
[ [ , ] EXPIREDATE = { date | @date_var }
| RETAINDAYS = { days | @days_var } ]
[ [ , ] PASSWORD = { password | @password_variable } ]
[ [ , ] { FORMAT | NOFORMAT } ]
 [ [ , ] { INIT | NOINIT } ]
 [ [ , ] { NOSKIP | SKIP } ]
 [ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable } ]
 [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
 [ [ , ] MEDIAPASSWORD = {mediapassword | @mediapassword_variable} ]
 [ [ , ] NAME = { backup_set_name | @backup_set_name_var } ]
 [ [ , ] { NOREWIND | REWIND } ]
 [ [ , ] { NOUNLOAD | UNLOAD } ]
 [ [ , ] RESTART ]
 [ [ , ] STATS [ = percentage ] ]
 [ [ , ] COPY_ONLY
```

پارامترهای این دستور در جدول۱-۱۰ آمده است.

۲-۱۰. دستور RESTORE DATABASE

روش‌های مختلفی برای بازیابی پشتیبان وجود دارد. دستور RESTORE DATABASE نسخه کامل بانک اطلاعات را که با دستور BACKUP DATABASE از آن پشتیبان تهیه گردید، بازیابی می‌کند. این دستور به صورت زیر به کار می‌رود:

```
RESTORE DATABASE { database_name | @database_name_var }
[ FROM <backup_device> [ ,...n ] ]
[ WITH
  [ { CHECKSUM | NO_CHECKSUM } ]
  [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
  [ [ , ] FILE = { file_number | @file_number } ]
  [ [ , ] KEEP_REPLICATION ]
  [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
  [ [ , ] MEDIAPASSWORD = { mediapassword |
    @mediapassword_variable } ]
  [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ]
                [ ,...n ]
  [ [ , ] PASSWORD = { password | @password_variable } ]
  [ [ , ] { RECOVERY | NORECOVERY | STANDBY =
    {standby_file_name | @standby_file_name_var }
  } ]
```

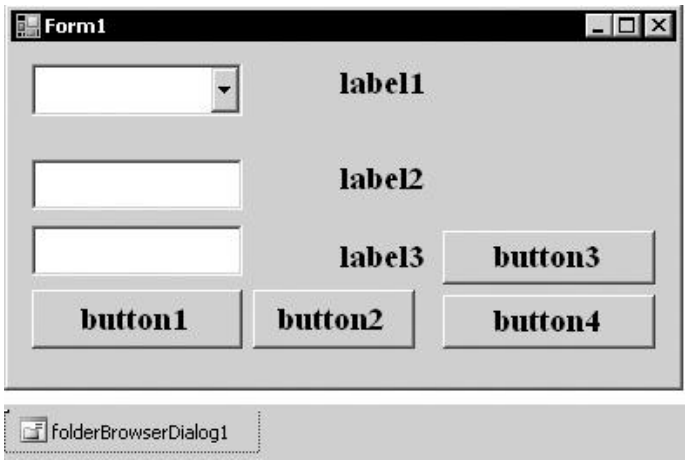
```
  [ [ , ] REPLACE ]
  [ [ , ] RESTART ]
  [ [ , ] RESTRICTED_USER ]
  [ [ , ] { REWIND | NOREWIND } ]
  [ [ , ] STATS [ = percentage ] ]
  [ [ , ] { STOPAT = { date_time | @date_time_var }
  |  STOPATMARK = { 'mark_name' | 'lsn:lsn_number' }
     [ AFTER datetime ]
  |  STOPBEFOREMARK = { 'mark_name' | 'lsn:lsn_number' }
     [ AFTER datetime ]
  } ]
  [ [ , ] { UNLOAD | NOUNLOAD } ]
]
[;]
```

پارامترهای این دستور مانند پارامتر دستور BACKUP DATABASE است. برخی از پارامترهای دیگر این دستور در جدول۲-۱۰ آمده‌اند.

جدول ۲-۱۰ برخی از پارامترهای دستور RESTORE DATABASE.

پارامتر	هدف
RESTRICTED_USER	تعیین می‌کند فقط کاربران db_owner، sysadmin یا dbcreator می‌توانند پس از بازیابی اطلاعات به بانک دستیابی داشته باشند.
MOVE	فایل‌ها را در زمان بازیابی به مکان دیگری منتقل می‌کند.
KEEP_REPLICATION	در هنگام بازیابی بانک اطلاعات آن را بر روی سرویس‌دهنده دیگری تکثیر می‌کند.
PARTIAL	عمل بازیابی را به صورت جزئی انجام می‌دهد.
REPLACE	تعیین می‌کند نام بانک اطلاعات که از آن پشتیبان گرفته‌اید، می‌تواند با نام بانک اطلاعات که در آن بازیابی می‌شوند، یکسان باشد.

مثال ۱-۱۰ برنامه‌ای که با استفاده از دستور Backup و Restore از بانک اطلاعات پشتیبان تهیه کرده یا پشتیبان را بازیابی می‌کند.



مراحل طراحی و اجرا

۱. پروژه جدیدی به نام **BackupRestore** ایجاد کنید.

۲. دستورات زیر را در بخش **Imports** تایپ کنید:

```
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
```

۳. سه کنترل **Label**، یک کنترل **ComboBox** (برای نمایش لیست بانک‌های اطلاعات)، دو کنترل **TextBox**، چهار کنترل **Button** و یک کنترل **FolderBrowserDialog** (برای انتخاب مسیر پشتیبان‌گیری و بازیابی پشتیبان) به برنامه اضافه کنید.

٤. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد **Form1_Load** را به صورت زیر تایپ کنید:

```
Private Sub Form1_Load(ByVal sender As System.Object,ByVal e_
As System.EventArgs) Handles MyBase.Load
  label1.RightToLeft = RightToLeft.Yes
  label2.RightToLeft = RightToLeft.Yes
  button1.Text = "پشتیبان گیری"
  button2.Text = "بازیابی"
  button3.Text = "انتخاب مسیر"
  button4.Text = "خروج"
  label1.Text = "نام بانک اطلاعات جهت پشتیبان گیری"
  label2.Text = "نام بانک اطلاعات بازیابی در سرور"
  label3.Text = "مسیر"
  comboBox1Fill()
```

```
End Sub
```

این دستورات، خواص کنترل‌های روی فرم را مقداردهی می‌کننـد و تـابع comboBox1Fill() را فراخـوانی می‌نمایند تا لیست بانک‌های اطلاعاتی را در کنترل comboBox1 نمایش دهند.

۵. به قبل از رویداد Form1_Load() بروید و دستورات زیر را تایپ کنید:

```
Dim SQL As String = "SELECT name FROM sys.databases"
Dim connStr As String = "Server=.\SQLEXPRESS ;Initial
Catalog= ;Integrated Security=True"
Dim conn As SqlConnection
Dim da As SqlDataAdapter
Dim ds As DataSet = New DataSet()
```

این دستورات، ابتدا دستور SQLی را تعریف می‌کنند که برای بازیابی بانک‌های اطلاعات به کار می‌رود. در ادامـه، رشـته اتصـال را تعریـف کـرده، مقـداردهی مـی‌کننـد و در پایـان، اشـیایی از قبیـل SqlConnection، SqlDataAdapter و DataSet را تعریف می‌نمایند.

٦. در ادامه، تابع comboBox1Fill را به صورت زیر تعریف کنید:

```
Private Sub comboBox1Fill()
  conn = New SqlConnection(connStr)
  da = New SqlDataAdapter(SQL, conn)
  ' Add items to the combo box...
  conn.ConnectionString = connStr
  conn = New SqlConnection(connStr)
  conn.Open()
  da = New SqlDataAdapter(SQL, conn)
  ds.Clear()
  da.Fill(ds, "sys.databases")
  comboBox1.DataSource = ds.Tables(0).DefaultView
  comboBox1.DisplayMember = "name"
End Sub
```

این تابع، ابتدا نمونه‌هایی از اشیا SqlConnection و SqlDataAdapter به نام‌های conn و da را ایجاد کرده، سپس، خواص شیء conn را مقدار داده آن را باز می‌کند. در ادامه، نام بانک‌های اطلاعات متصـل بـه سـرویس دهنده را به da انتقال می‌دهد (با دسـتور SELECT name FROM sys.databases) و در پایـان، نـام بانـک‌هـای اطلاعات را به ds و از طریق آن بر روی comboBox1 انتقال می‌دهد.

۷. دکمه button1 را کلیک مضاعف کرده، و دستورات رویداد Click آن را به صورت زیـر تغییـر دهید:

```
Private Sub button1_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles button1.Click
  Try
    If textBox2.Text = "" Then
       MessageBox.Show(" نشد انتخاب مسیر ")
    Else
     If conn.State = ConnectionState.Open Then
         conn.Close()
     End If
    conn.ConnectionString = connStr
    conn.Open()
    Dim cmdBackup As String = "BACKUP DATABASE " +
      comboBox1.Text + " TO DISK = '" + textBox2.Text +
      comboBox1.Text + "1.bak', " + "DISK='" + textBox2.Text
      + comboBox1.Text + "2.bak' "
    Dim cmd As SqlCommand = New SqlCommand(cmdBackup, conn)
    cmd.CommandType = CommandType.Text
    cmd.ExecuteNonQuery()
    conn.Close()
    MessageBox.Show("بانک اطلاعات مورد نظر پشتیبان گرفته شد")

            End If
        Catch

            MessageBox.Show("خطا در پشتیبان گیری رخ داد")
        End Try
End Sub
```

این دستورات، از بانک اطلاعات انتخاب شده در comboBox1 پشتیبان تهیه می‌کنند (مسیر پشتیبان توسط کاربر انتخاب می‌شود). برای این منظور، ابتدا دستور Try را قرار داده تا در صورت بروز خطا در هنگام پشتیبان‌گیری، برنامه قطع نشود و پیام «**خطا در پشتیبان‌گیری رخ داد**» ظاهر شود. در دستورات، Try ابتدا تست می‌کند textBox2 خالی نباشد (یعنی، مسیر قرارگرفتن پشتیبان توسط کاربر انتخاب شده باشد)، سپس، تست می‌کند آیا اتصال به بانک باز است یا خیر. اگر اتصال باز باشد، آن را می‌بندد. در ادامه، رشته اتصال شیء conn را مقداردهی کرده اتصال را باز می‌نماید. سپس، دستور پشتیبان‌گیری را ایجاد می‌کند (دستور String cmdBackup = ...). در پایان، یک شیء SqlCommand به نام cmd ایجاد کرده، خواص آن را مقداردهی می‌کند و دستور پشتیبان‌گیری را با فراخوانی متد ExecuteNonQuery اجرا می‌کند و اتصال را بسته، پیام مناسب نمایش می‌دهد.

۸. دکمه button2 را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles button2.Click
  Try
   If textBox1.Text = "" Or textBox2.Text = "" Then
     MessageBox.Show("نام بانک اطلاعات یا مسیر انتخاب نشد")
   Else
    connStr = "Integrated Security=SSPI Data
       Source=.\SQLEXPRESS "
     conn = New SqlConnection(connStr)
    If conn.State = ConnectionState.Open Then
      conn.Close()
    End If
    conn.ConnectionString = connStr
    conn.Open()
    Dim cmdRestore As String = "RESTORE DATABASE " +
     textBox1.Text + " FROM DISK = '" + textBox2.Text +
     textBox1.Text + "1.bak', " + "DISK =  '" + textBox2.Text
     + textBox1.Text + "2.bak' "
    Dim cmd As SqlCommand = New SqlCommand(cmdRestore, conn)
    cmd.CommandType = CommandType.Text
    cmd.ExecuteNonQuery()
    MessageBox.Show("بانک اطلاعات مورد نظر بازیابی شد")
    conn.Close()
  End If
 catch
   MessageBox.Show("در بازیابی بانک اطلاعات خطا رخ داد")
 End Try
End Sub
```

این دستورات، بانک اطلاعاتی که نام آن در textBox1 وارد شده است و پشتیبان آن در مسیری که در textBox2 وارد گردید، را بازیابی می‌کنند. برای انجام این کار، از Try استفاده می‌کند تا در صورت بروز خطا در برنامه بازیابی پشتیبان، قطع نگردد. در بلاک Try، ابتدا بررسی می‌کند که محتویات textBox1 یا textBox2 خالی نباشد (اگر خالی باشد، پیام مناسب نمایش می‌دهد). سپس، رشته اتصال را تغییر داده یک شیء اتصال ایجاد کرده، اگر اتصال باز باشد، آن را می‌بندد. در ادامه، رشته اتصال را مقدار داده و اتصال را با متد Open() باز می‌کند. در پایان، دستور SQLایی که برای بازیابی به کار می‌رود را ایجاد کرده با ایجاد شیء SqlCommand و اجرای متد ExecuteNonQuery() دستور بازیابی را اجرا می‌کند و پیام مناسب را نمایش داده اتصال می‌بندد.

۹. دکمه button3 را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
Private Sub button3_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles button3.Click
```

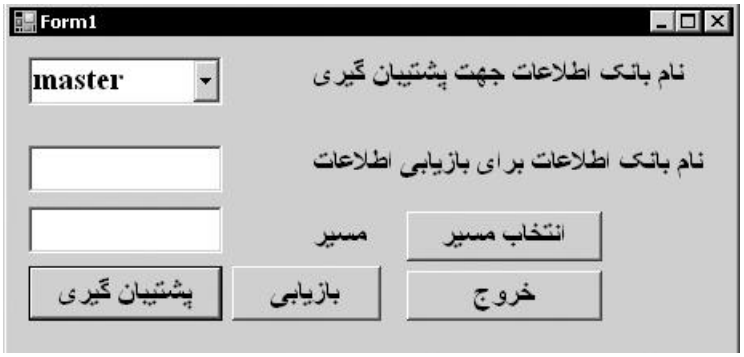
```
    folderBrowserDialog1.Description=" مسیر مورد نظر را انتخاب کنید"
    folderBrowserDialog1.ShowDialog()
    textBox2.Text = folderBrowserDialog1.SelectedPath
End Sub
```

این دستورات، با استفاده از کنترل FolderBrowserDialog مسیر پشتیبان‌گیری و بازیابی پشتیبان را انتخاب می‌کنند.

۱۰. دکمه button4 را کلیک مضاعف کرده، دستورات آن را به صورت زیر تغییر دهید:

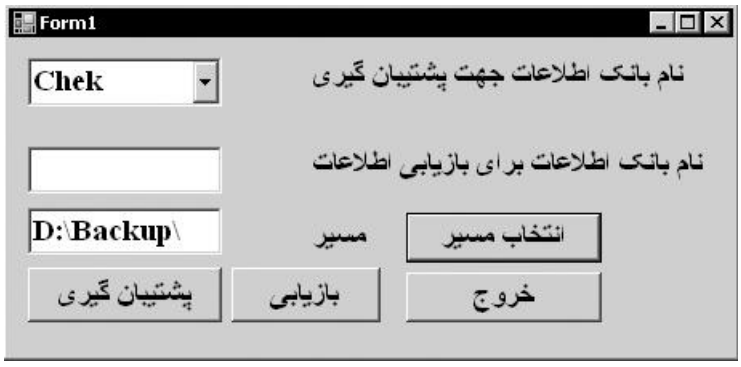
```
Private Sub button4_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles button4.Click
  End
End Sub
```

۱۱. برنامه را ذخیره و اجرا کنید. اکنون شکل۱-۱۰ ظاهر می‌شود.



شکل۱-۱۰ نمونه اولیه خروجی مثال۱-۱۰.

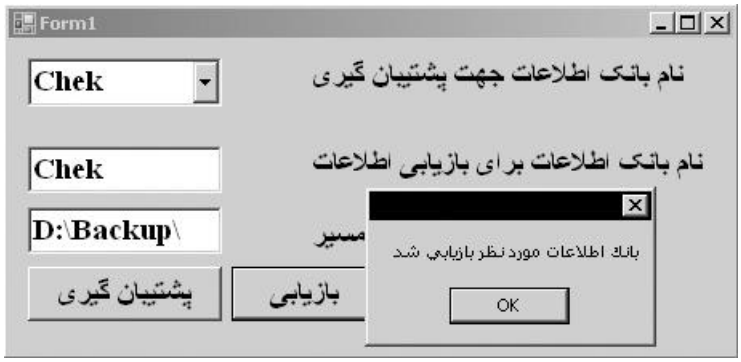
در comboBox1 نام بانک اطلاعات را انتخاب کنید (من بانک اطلاعات Chek را انتخاب نمودم) و دکمه انتخاب مسیر را کلیک کنید تا شکل انتخاب مسیر ظاهر شود. در این شکل مسیر مورد نظر (به عنوان مثال، D:\Backup را انتخاب کنید و دکمه OK را کلیک کرده تا مسیر انتخاب شده را در textBox2 ببینید (شکل۲-۱۰)).



شکل۲-۱۰ نمونه خروجی برای پشتیبان‌گیری.

اکنون دکمه پشتیبان‌گیری را کلیک کنید تا شکل زیر ظاهر شود:

دکمه OK را کلیک کنید. برای بازیابی پشتیبان در textBox1 نام بانک اطلاعات برای بازیابی را وارد کنید (من، Chek را وارد کردم). چون مسیر از قبل مشخص است، دکمه بازیابی را کلیک کنید تا شکل۳-۱۰ ظاهر شود. برای خروج از برنامه دکمه OK سپس، دکمه خروج را کلیک کنید.



شکل۳-۱۰ نمونه خروجی برای بازیابی اطلاعات.

۳-۱۰. مدیریت دایرکتوری و فایل‌ها

در این فصل برای پشتیبان‌گیری و بازیابی پشتیبان نیاز داریم دایرکتوری‌ها و فایل‌هایی را ایجاد کنیم. لـذا در این بخش به مفهوم دایرکتوری‌ها و فایل‌ها می‌پردازیم. برای کار با فایل‌ها و دایرکتوری‌ها ابتدا بایـد فضـای نـام System.IO را به برنامه اضافه کنید (با دستور زیر):

```
Imports System.IO
```

اکنون می‌توانید از کلاس‌های

فایل‌ها)، StreamReader (برای خواندن و نوشتن در فایل‌های متنی) و غیره استفاده کنید.

۱-۳-۱۰. کلاس File

این کلاس برای مدیریت بر فایل به کار می‌رود. اعمالی که با این کلاس می‌توان بـر روی فایـل انجـام داد، عبارت‌اند از: ۱. کپی کردن فایل ۲. تغییرنام فایل ۳. ایجاد فایل ٤. حذف فایل و ۵. غیره.

متدهایی برای انجام این کارها وجود دارند که برخی از این متدها در جدول۳-۱۰ آمده‌اند.

۲-۳-۱۰. کلاس Directory

این کلاس برای انجام اعمالی از قبیل ایجاد، حذف، تغییر نام و اعمال دیگر بر روی دایرکتوری‌هـا بـه کـار می‌رود. برخی از خواص و متدها این کلاس در جدول٤-۱۰ آمده‌اند.

جدول ٤-۱۰ برخی از خواص مهم کلاس Directory.

خاصیت	هدف
Exists	مشخص می‌نماید آیا پوشه وجود دارد یا خیر.
FullName	مسیر کامل دایرکتوری را تعیین می‌کند.
Name	نام دایرکتوری را برمی‌گرداند.
Root	ریشه دایرکتوری تعیین شده را برمی‌گرداند.
متد	**هدف**
CreateDirectory	یک دایرکتوری را ایجاد می‌کند.
Delete	دایرکتوری خاصی را حذف می‌کند.
Move	دایرکتوری خاصی را به مکان جدید منتقل می‌کند.

۳-۳-۱۰. کلاس FileStream

این کلاس برای باز کردن فایل به کار می‌رود تا بتوانید اطلاعات باینری را در آن بنویسید یا از آن بخوانید. برای استفاده از کلاس FileStream باید نمونه‌ای از آن ایجاد کنید و خواص (جدول۵-۱۰) آن را مقداردهی نمایید. برای ایجاد نمونه می‌توانید به صورت زیر عمل کنید:

```
Dim streamName As FileStream = New FileStream(fileName, fileMode, fileAccess, fileShare)
```

fileName، نام فایلی است که باید باز شود. fileMode حالت باز کردن فایل را مشخص می‌کند. مقادیر آن را در جدول۶-۱۰ می‌بینید. fileAccess، شیوه دستیابی به فایل را تعیین می‌کند که مقادیر آن در جدول۷-۱۰ آمده است. FileShare، تعیین می‌کند که فایل چگونه به طور همزمان (معمولاً در شبکه) استفاده شود (جدول ۸-۱۰). به عنوان مثال دستورات زیر را ببینید:

```
Dim inp As FileStream = New FileStream("daftar.bak", FileMode.Create,
   FileAccess.Write, FileShare.None)
```

این دستور، فایل daftar.bak را برای نوشتن ایجاد می‌کند به طوری که کاربران دیگر در شبکه نمی‌توانند به طور همزمان به آن دسترسی داشته باشند.

جدول ۵-۱۰ خواص کلاس FileStream.

خاصیت	هدف
CanRead	تعیین می‌کند آیا فایل قابل خواندن است یا خیر.
CanWrite	تعیین می‌کند آیا فایل امکان نوشتن را دارد یا خیر.
Length	تعداد بایت‌های یک استریم را تعیین می‌کند.
Name	نام استریم را تعیین می‌کند.
Position	مکان فعلی استریم را تعیین می‌نماید.

خواندن از استریم

برای خواندن اطلاعات از استریم می‌توانید از دو متد ReadByte() و Read() استفاده کنید.

- **متد ReadByte()**: جهت خواندن یک کاراکتر از Stream به کار می‌رود و به صورت زیر استفاده می‌شود:

```
Dim کاراکتر خوانده شده AS Integer = نام شیء.ReadByte()
```

به عنوان مثال، دستورات زیر را ببینید:

```
Dim fsSource As FileStream = New FileStream(srcPath, FileMode.Open,_
                                FileAccess.Read, FileShare.Read)
  Dim firstByte AS Integer = fsSource.ReadByte()
```

دستور اول، فایلی که نام و مسیر آن در srcPath قرار دارد را به صورت فقط خواندنی باز می‌کند و دستور دوم، اولین بایت آن را خوانده در متغیر firstByte قرار می‌دهد.

متد ()Read: برای خواندن تعدادی کارکتر از مکان خاصی در فایل به کار می‌رود و به صورت زیر استفاده می‌شود:

```
نام شیء.Read(array, offset, count)
```

array، بایت‌هایی که از فایل خوانده شده است در آن قرار می‌دهد. offset، مکان شروع خواندن (کارکتر شروع خواندن) از فایل را تعیین می‌کند و Count، تعداد کارکترهایی که باید از فایل خوانده شود را تعیین می‌نماید. به عنوان مثال، دستورات زیر را ببینید:

```
Dim bufferWrite As byte()
Dim fsSource As FileStream = New FileStream(srcPath, FileMode.Open,_
      FileAccess.Read, FileShare.Read)
bufferWrite = New byte(fsSource.Length)
fsSource.Read(bufferWrite, 0, bufferWrite.Length)
```

دستور اول، بافری را برای خواندن از فایل به نام bufferWrite تعریف می‌کند. دستور دوم، شیء fsSource را از نوع FileStream تعیین کرده، فایلی را که نام و مسیر آن در srcPath قرار دارد، به صورت فقط خواندنی باز کرده، دستور سوم، آرایه bufferWrite را به اندازه تعداد کارکترهای موجود در فایل در نظر می‌گیرد و دستور چهارم، کارکترهای موجود در استریم fsSource را خوانده در آرایه bufferWrite قرار می‌دهد.

نوشتن در استریم

برای نوشتن در استریم نیز دو متد ()WriteByte و ()Write وجود دارد:

متد ()WriteByte: یک بایت را در استریم می‌نویسد و به صورت زیر به کار می‌رود:

```
نام شیء.WriteByte(byte value)
```

value، مقداری را تعیین می‌کند که در فایل باید نوشته شود. به عنوان مثال، دستورات زیر را ببینید:

```
Dim fsDes As FileStream = New FileStream(srcPath, FileMode.Open,_
   FileAccess.Write, FileShare.Read)
fsDes.WriteByte(12)
```

دستور اول، فایل srcPath را به صورت نوشتنی باز کرده و دستور دوم، مقدار ۱۲ را در آن می‌نویسد.

- **متد ()Write**: برای نوشتن آرایه‌ای از بایت‌ها در مکان خاصی از استریم به کار می‌رود و به صورت زیر استفاده می‌شود:

```
نام شیء.Write(array, offset, count)
```

array، بایت‌هایی را تعیین می‌کند که باید در فایل نوشته شوند. offset، مکان شروع نوشتن را تعیین می‌کند و count، تعداد کارکترهایی که باید در فایل نوشته شوند را مشخص می‌نماید. به عنوان مثال، دستورات زیر را ببینید:

```
Dim fsDes As FileStream = New FileStream(srcPath, FileMode.Open,_
  FileAccess.Write, FileShare.Read)
fsDes.Write(bufferWrite, 10, bufferWrite.Length)
```

دستور اول، فایلی که نام آن در surPath قرار دارد به صورت نوشتنی باز می‌کند و دستور دوم، اطلاعات bufferWrite را از مکان ۱۱ (مکان شروع اشاره‌گر فایل، صفر است) در آن فایل می‌نویسد.

۴-۳-۱۰. کلاس GzipStream

این کلاس، برای فشرده‌سازی فایل و خارج نمودن فایل از حالت فشرده به کار می‌رود. کلاس GzipStream در فضای نام System.IO.Compression قرار دارد. برای استفاده از این کلاس ابتدا باید این فضای نام را با دستور زیر به برنامه اضافه کنید:

```
Imports System.IO.Compression
```

سپس، نمونه‌ای از این کلاس ایجاد کنید. این کار به صورت زیر انجام می‌شود:

```
Dim نام نمونه As GzipStream = New GzipStream("نام فایل", CompressionMode, leaveOpen)
```

در این دستور CompressionMode، تعیین می‌کند که فایل فشرده شود (با مقدار CompressionMode.Compress) یا از حالت فشرده خارج گردد (با مقدار CompressionMode.Decompress) و leaveOpen، تعیین می‌کند که آیا فایل باز شود یا خیر. اگر این مقدار True باشد، فایل را باز می‌کند.

در پایان، باید به خواص آن مقدار دهید و از متدهای آن برای خواندن و نوشتن استفاده کنید. خـواص و متدهای شیء GzipStream در جدول۹–۱۰ آمده است. به عنوان مثال، دستورات زیر را ببینید:

```
Dim gzCompressed As GZipStream = New
 GZipStream(fsDest, CompressionMode.Compress, True)
gzCompressed.Write(bufferWrite, 0, bufferWrite.Length)
gzCompressed.Close()
```

دستور اول، فایلی که نام آن در fsDest قرار دارد را به صورت فشرده باز مـی‌کنـد. دسـتور دوم، اطلاعـات رشته bufferWrite را در آن می‌نویسد و دستور سوم، فایل را می‌بندد.