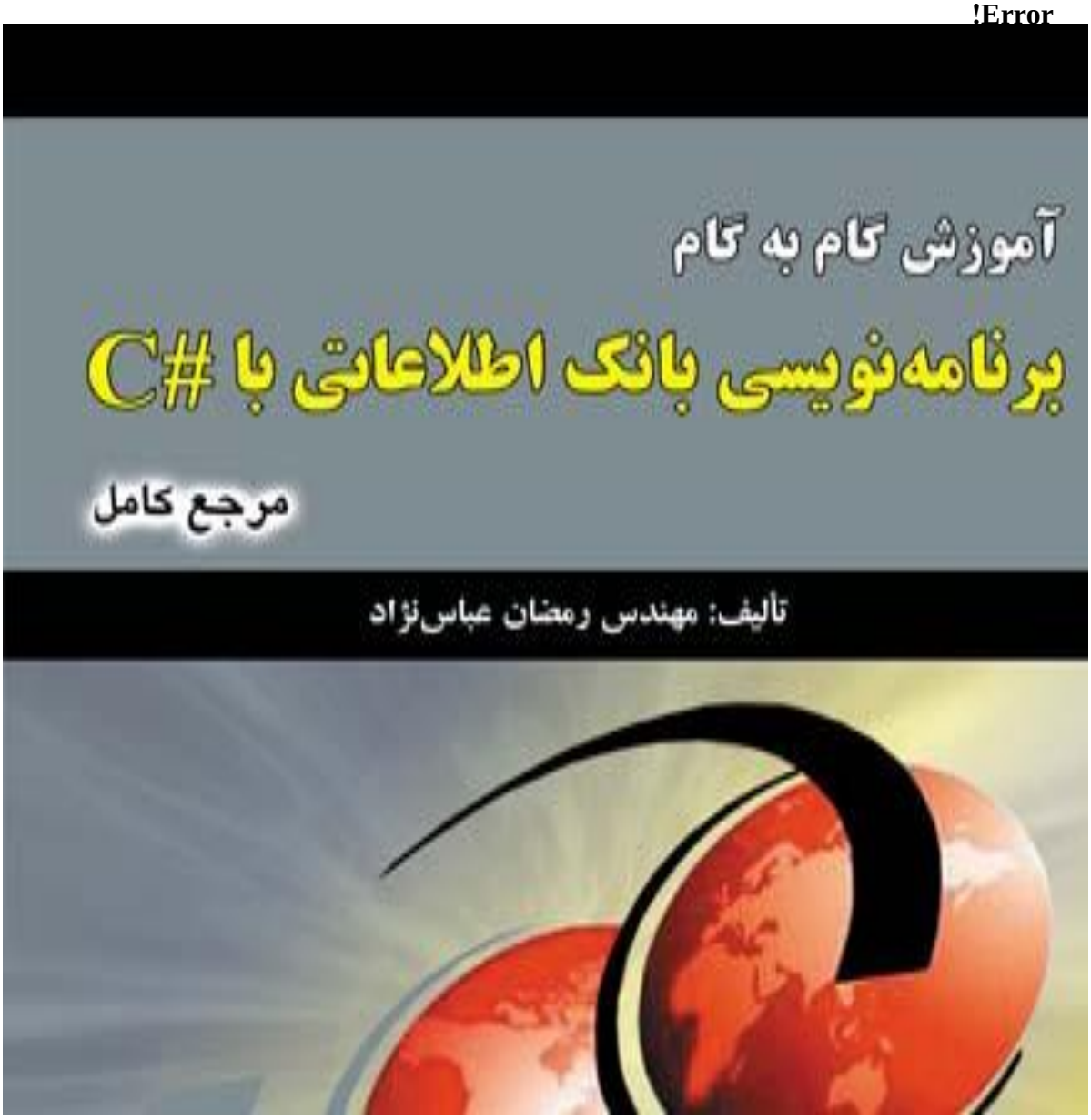
!Error


آموزش گام به گام
برنامه‌نویسی بانک اطلاعاتی با C#
مرجع کامل
تألیف: مهندس رمضان عباس‌نژاد

آموزش گام به گام برنامه نویسی

بانک اطلاعاتی با C#

(مرجع کامل)

تالیف

مهندس رمضان عباس‌نژادورزی

فن‌آوری نوین

سرشناسه	:	عباس‌نژادورزی ، رمضان، ۱۳۴۸ -
عنوان و نام پدیدآور	:	آموزش گام به گام برنامه‌نویسی بانک اطلاعاتی با C# (مرجع کامل)/ تالیف رمضان عباس‌نژادورزی.
مشخصات نشر	:	بابل: فن‌آوری نوین، ۱۳۸۸.
مشخصات ظاهری	:	۳۰۴ص.: مصور، جدول.
شابک	:	۶۵۰۰۰ ریال:978-600-91413-2-6
وضعیت فهرست نویسی	:	فیپا
موضوع	:	سی شارپ (زبان برنامه‌نویسی کامپیوتر)
موضوع	:	پایگاه‌های اطلاعاتی -- مدیریت
رده بندی کنگره	:	QA۷۶/۷۳/س۹۵ع۲ ۱۳۸۸
رده بندی دیویی	:	۰۰۵/۱۳۳
شماره کتابشناسی ملی	:	۴۱۰۸۰۹۱

فن‌آوری نوین

www.fanavarinovin.net

بابل، صندوق پستی ۷۳٤٤۸-٤۷۱٦۷ تلفن: ۲۲۵٦٦۸۷-۰۱۱۱

آموزش گام‌به‌گام برنامه نویسی بانک اطلاعاتی با #c (مرجع کامل)

تألیف: مهندس رمضان عباس‌نژادورزی

ناشر: فن‌آوری نوین

چاپ اول: پاییز ۱۳۸۸

جلد :۲۰۰۰

شابک: ٦- ۲ - ۹۱٤۱۳ - ٦۰۰ - ۹۷۸

حروفچینی و صفحه‌آرایی: فن‌آوری نوین **قیمت:٦۵۰۰ تومان**

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

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 که 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 SQL Server 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 که C# به عنوان یکی از دروس رشته‌های کامپیوتر و IT در دانشگاه‌های کشور تدریس می‌شود، این زبان را به عنوان زبان برنامه‌نویسی بانک اطلاعات انتخاب نمودیم.

در این کتاب مطالبی از قبیل ایجاد بانک اطلاعات، جداول، کلاس‌های C# 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 جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

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

عباس نژادورزی
fanavarienovin@gmail.com

فصل ۱ بخشی از فصل آشنایی با زبان C#

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، C# می‌باشد. به همین دلیل در این فصل به طور خلاصه زبان C# را می‌آموزیم.

۱-۱. زبان C#

این زبان به همراه نرم‌افزار ویژوال استودیونت ارائه شده است. زبان C# 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 C# از فناوری شی‌گرایی استفاده می‌نماید، امکاناتی در این زبان اضافه شده است که می‌توانید اشیای دنیای واقعی را مدل‌سازی کنید. برای این که C# 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.

۱-۲. فضای نام

همان‌طور که بیان گردید، زبان برنامه‌نویسی C# 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.

---

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

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 #C 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 using استفاده نمایید. به عنوان مثال، دستورات زیر را ببینید:

```
using System.Data.SqlClient;
using 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.

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 #C 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 می‌آموزیم.

[4]- Namespaces

جدول۲-۱ انواع داده‌های اولیه در #C.			
نوع در زبان #C	معادلNET.	اندازه	نگهداری می‌کند.
byte	Byte	۱	مقادیر بدون علامت (۰ تا ۲۵۵)
char	Char	۱	کارکترهای یونیکد
bool	Boolean	۱	مقادیر True یا False
sbyte	Sbyte	۱	مقادیر علامت‌دار (۱۲۸- تا ۱۲۷)
short	Int16	۲	مقادیر صحیح از ۳۲۷۶۸- تا ۳۲۷۶۷
Ushort	Uint16	۲	مقادیر صحیح بدون علامت از ۰ تا ۶۵۵۳۵
int	Int32	٤	مقادیر صحیح بین -2^{31} تا $(2^{31}-1)$
uint	Uint32	٤	مقادیر صحیح بدون علامت ۰ تا 2^{32}
float	Single	٤	اعداد اعشاری $\pm 1/5\times10^{-45}$ تا $\pm 3/4\times10^{38}$ با ۷ رقم اعشار
double	Double	۸	اعداد اعشاری $\pm 5/0\times10^{-324}$ تا $\pm 1/7\times10^{308}$ با ۱۵ تا ۱۶ رقم بعد از اعشار
decimal	Decimal	۸	نقطه اعشار ثابت تا ۲۸ رقم
Long	Int64	۸	مقادیر صحیح -2^{63} تا $(2^{63}-1)$
ulong	Uint64	۸	مقادیر ۰ تا $(2^{64}-1)$

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 #C می‌توان از ترکیبی از حروف، ارقام و خط ربط (_) استفاده کرد. به طوری که اولین کارکتر 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:

; نام متغیر نوع داده سطح دستیابی

سطح دستیابی، تعیین می‌کند که در چه محدوده‌های می‌توانید به متغیرها دستیابی داشته باشید و مهم‌ترین آنها public و private هستند. public موجب می‌شود تا متغیر را بتوانید در کل کلاس دستیابی داشته باشید و private موجب می‌شود متغیر را بتوانید در همان بلاک دستیابی داشته باشید. اگر سطح دستیابی ذکر نشود، private در نظر گرفته خواهد شد. در ادامه بیشتر با سطوح دستیابی آشنا خواهید شد.

اکنون دستورات زیر را ببینید:

```
int x;
double d;
string s1, s2;
```

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.

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:

```
int x = 5 , y = 10;
string s = "good";
```

دستور اول، متغیرهای x و y را با مقادیر 5 و 10 از نوع int 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:

```
int x;
string s;
x = 10;
s = "C#.NET";
```

دستورات اول و دوم متغیرهای x و s را به ترتیب از نوع int و string تعریف می‌کنند. دستور سوم، مقدار متغیر x را برابر ۱۰ قرار می‌دهد و دستور چهارم، مقدار متغیر s را رشته C#.NET 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 نوع داده نام = مقدار ثابت ;

به عنوان مثال، دستورات زیر را ببینید:

```
const float PI = 3.14;
const char ch = '+';
```

دستور اول، ثابت PI را با مقدار 3.14 از نوع اعشاری معرفی می‌کند و دستور دوم، ثابت ch را از نوع کارکتری با مقدار '+' تعریف می‌نماید.

۶-۱. عملگرها

عملگرها، نمادهایی هستند که اعمال خاصی را انجام می‌دهند. به عنوان مثال، نماد '-' عملگری است که دو مقدار را از یکدیگر تفریق می‌کند. عملگرها در C# به انواع مختلف تقسیم می‌شوند که در زیر آمده‌اند:

- **عملگرهای محاسباتی**، عملگرهایی هستند که عمل محاسبات را بر روی عملوند انجام می‌دهند. این عملگرها در جدول ۳-۱ آمده‌اند.
- **عملگرهای رابطه‌ای**، عملگرهایی هستند که دو عملوند را با یکدیگر مقایسه می‌کنند (جدول ٤-۱ را ببینید).
- **عملگرهای منطقی**، عملگرهایی هستند که بر روی عبارت منطقی (True یا False) عمل می‌کنند (جدول ٥-۱ را مشاهده کنید).
- **عملگرهای ترکیبی**، از ترکیب عملگرهای محاسباتی و علامت = ایجاد می‌شوند (جدول٦-۱).
- **عملگرهای بیتی**، عملگرهایی هستند که برای تست کردن، مقدار دادن یا شیفت دادن و سایر اعمال بر روی عملوندها به کار می‌روند. عملگرهای بیتی در جدول ۷-۱ آمده‌اند.

جدول ۳-۱ عملگرهای محاسباتی.

عملگر	نام	x	y	مثال	نتیجه
-	تفریق و منهای یکانی	10	12	y-x -y	2 -12
+	جمع	17	5	x +y	22
*	ضرب	10	2	x*y	20
/	تقسیم	10	5	x/y	2
%	باقیمانده تقسیم صحیح	10	3	x%y	1
--	کاهش	10	5	--x , y--	9 , 4
++	افزایش	10	5	x++ , ++y	11 , 6

جدول ٤-۱ عملگرهای رابطه‌ای.

عملگر	نام	x	y	مثال	نتیجه
>	بزرگتر	10	12	x > y	False
=>	بزرگتر یا مساوی	10	7	x > = y	True
<	کوچکتر	10	7	x < y	True
<=	کوچکتر یا مساوی	10	12	x <= y	False
= =	تساوی	10	17	x = = y	False
!=	نامساوی	10	17	x != y	True

جدول ۵-۱ عملگرهای منطقی.

عملگر	نام	x	y	مثال	نتیجه
!	نقیض (not)	10	12	! x	False
&&	و (and)	10	12	x < 12 && y > 10	True
||	یا (or)	12	8	x > 12 || y < 6	False

جدول ٦-۱ عملگرهای ترکیبی.

عملگر	نام	X	y	مثال	معادل	نتیجه
+ =	انتساب جمع	10	12	x += y	x = x + y	x = 22
- =	انتساب تفریق	10	8	x -= y	x = x - y	x = 2
* =	انتساب ضرب	10	12	x *= y	x = x * y	x = 120
/ =	انتساب تقسیم	10	2	x /= y	x = x / y	x = 5
% =	انتساب باقیمانده تقسیم	10	4	x %= y	x = x % y	x = 2

جدول ۷-۱ عملگرهای بیتی.

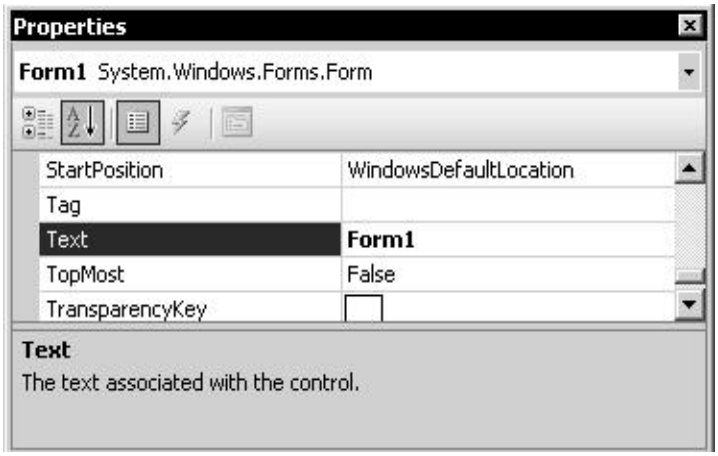
عملگر	نام	x	y	مثال	نتیجه
&	و	10	2	x & y	2
|	یا	10	3	x | y	11
^	یا انحصاری	10	3	x ^ y	9
~	نقیض	126	3	~ x	1
>>	شیفت به راست	7	3	x >> y	56
<<	شیفت به چپ	100	2	x << y	25

۷-۱. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه بانک اطلاعات در C# 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):



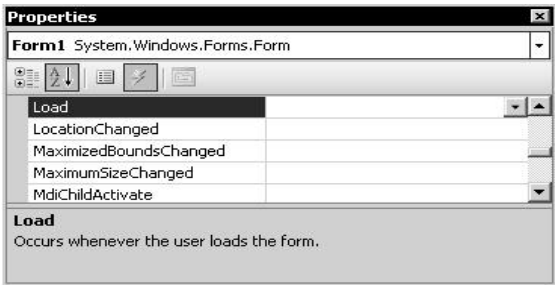
جدول ۸-۱ خواص مهم فرم.

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	تعیین‌می‌کند آیا فرم فعال است یا خیر. اگر فرم غیرفعال گردد، به‌هیچ‌رویدادی‌پاسخ نمی‌دهد.
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.

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 ۹-۱ رویدادهای مهم فرم.

رویداد	هدف
Activated	وقتی که فرم فعال می‌شود، رخ می‌دهد.
Click	وقتی رخ می‌دهد که فرم کلیک گردد.
FormClosed	وقتی رخ می‌دهد که فرم بسته شود.
FormClosing	وقتی رخ می‌دهد که فرم در حال بسته شدن باشد. این رویداد قبل از رویداد FormClosed رخ می‌دهد.
Deactivate	وقتی رخ می‌دهد که فرم غیرفعال گردد.
DoubleClick	وقتی که فرم کلیک مضاعف شود، رخ می‌دهد.
Enter	وقتی مکان‌نما وارد فرم می‌شود، رخ می‌دهد.
KeyDown	وقتی کلیدی فشرده شده پایین می‌رود، رخ می‌دهد.
KeyPress	وقتی کلیدی فشرده می‌شود، رخ می‌دهد. این رویداد قبل از رویداد KeyDown اتفاق می‌افتد.
KeyUP	وقتی کلید فشرده شده رها می‌گردد، رخ می‌دهد.
Leave	وقتی مکان‌نما فرم را ترک می‌کند رخ می‌دهد.
Load	وقتی فرمی باز می‌شود، رخ می‌دهد (قبل از نمایش فرم).
MouseDown	وقتی که کلید ماوس فشرده شود، رخ می‌دهد.
MouseEnter	وقتی مکان‌نمای ماوس وارد فرم شود، رخ می‌دهد.
MouseLeave	وقتی مکان‌نمای ماوس فرم را ترک می‌کند، رخ می‌دهد.
MouseUP	وقتی کلید فشرده شده ماوس رها می‌شود، رخ می‌دهد.
Move	وقتی فرم شروع به حرکت می‌کند، رخ می‌دهد.
Resize	وقتی اندازه فرم تغییر می‌یابد، رخ می‌دهد.
TextChanged	وقتی رخ می‌دهد که متن فرم (کنترل) یا خاصیت Text تغییر کند.
Shown	وقتی که فرم نمایش داده شود، رخ می‌دهد.
SizeChanged	وقتی رخ می‌دهد که مقدار خاصیت Size تغییر می‌یابد.
MouseMove	وقتی رخ می‌دهد که ماوس روی فرم حرکت می‌کند.

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.

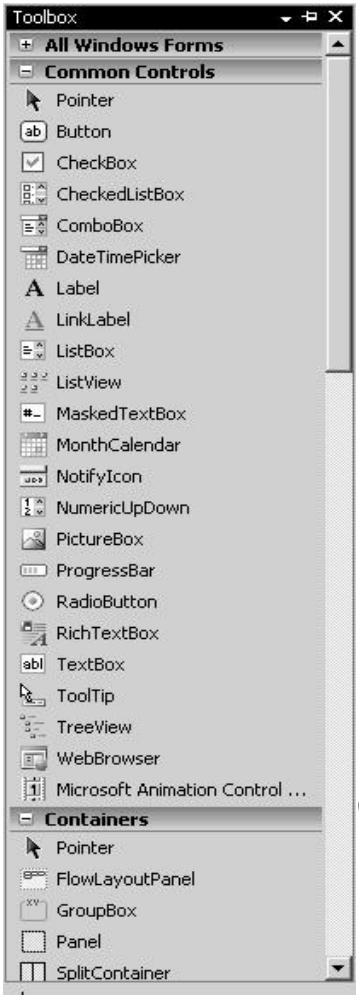
۸-۱. کنترل‌ها

همان‌طور که می‌دانید برای نوشتن برنامه‌ها در C# 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 C# را تشکیل می‌دهند. کنترل‌های زیادی در C# 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 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):

[۱]- براي كسب اطلاعات بيشتر در مورد C# به آموزش گام به گام C# از انتشارات علوم رايانه، تأليف عين‌الله جعفرنژادقمي و رمضان عباس‌نژاد مراجعه كنيد.

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.



شکل ۱-۱ پنجره ابزار ویژوال استودیونت

۱-۸-۱. کنترل Label (A Label)

این کنترل برای نمایش متن‌های غیرقابل ویرایش از قبیل عنوان فیلد، 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.

- **رویداد AutoSizeChanged**: وقتی رخ می‌دهد که مقدار خاصیت AutoSize تغییر کند.
- **رویداد TextAlignChanged**: وقتی رخ می‌دهد که مقدار خاصیت TextAlign تغییر یابد.

۱-۸-۲. کنترل TextBox (abl TextBox)

این کنترل برای دریافت اطلاعاتی از قبیل مقادیر رشته‌ای، عددی و Memo (اطلاعات رشته‌ای طولانی) 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 ۱-۱۱ آمده‌اند.

۱-۸-۳. کنترل Button (ab 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 بخشی از 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، 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[6] 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
٤. حذف رکوردهای بانک اطلاعات
۵. تغییر داده‌های بانک اطلاعات
٦. اجرای پرس‌وجو[7] بر روی جداول

---

[6]- Records [2]- Query

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 گردد، 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 می کنید. 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 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 شود. 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 مخصوصی 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.

۳-۱-۲. نرمال‌سازی داده‌ها

نرمال‌سازی، فرآیند تنظیم ساختار جدول‌های بانک اطلاعاتی است. هدف نهایی نرمال‌سازی این است که داده‌های موجود در بانک اطلاعاتی به ساده‌ترین ساختار آن تبدیل شود و داده‌های زاید به حداقل برسند. به عبارت دیگر، اهداف اولیه نرمال‌سازی، کاهش افزونگی داده، کاهش بی‌نظمی و کاهش داده‌های تهی[۸] است.

نرمال‌سازی، یک منشأ ریاضی پیچیده دارد که شامل مراحل خاصی به نام **فرم‌های نرمال**[۹] است. آقای کاد در مقاله اولیه خود سه فرم نرمال را معرفی کرد که به 1NF، 2NF و 3NF معروف هستند.[۱۰] شخص دیگری به نام «بویس» به همراه کاد فرم نرمال دیگری به نام BCNF[۱۱] را تعریف کرد. بعدها دیگران فرم‌های نرمال‌سازی 4NF و 5NF را معرفی کردند که نادرند. هرچه فرم نرمال بیشتر باشد، محدودیت تست بیشتر است. تا نرمال‌سازی فرم سوم (3NF) برای بانک اطلاعاتی کافی است.

- **در فرم نرمال‌سازی اول**، تست می‌شود که هیچ فیلدی (ستون) بیش از یک قلم داده را شامل نشود. به عنوان مثال، آدرس باید به بخش‌های کشور، استان، شهرستان، خیابان، کوچه و پلاک تقسیم شود. علاوه بر این، در این فرم نرمال‌سازی داده‌های تکراری حذف می‌شوند. به عنوان مثال، از تکرار بعضی از فیلدهای یک رکورد جلوگیری می‌کند. به زبان ساده‌تر، جدولی فرم نرمال اول است که:
  - همه کلیدهای آن تعریف شده باشند.
  - تمام فیلدهای آن به کلید اولیه وابستگی تابعی[۱۲] داشته باشند.
  - فیلدهای آن دارای دامنه تو در تو نباشند. فیلد **نام** باید به فیلدهای **نام** و **نام‌خانوادگی** تقسیم شوند.
- **فرم نرمال دوم**، هرگاه جدولی فرم نرمال اول باشد و فیلدهای آن به بخش‌های کلید اولیه وابسته نباشد، فرم نرمال دوم است.
- **فرم نرمال سوم**، هرگاه جدولی فرم نرمال دوم باشد و فیلدهای غیرکلید آن به فیلدهای غیرکلید دیگر وابسته نباشد، فرم نرمال سوم است. به عنوان مثال، با استفاده از فیلدهای **تعداد واحد** و **نمره**، می‌توان معدل دانشجو را محاسبه کرد. در نتیجه نیازی به فیلد معدل در جدول نیست.

توجه داشته باشید که وقتی بانک اطلاعاتی را طراحی می‌نماید، به نرمال‌سازی جداول آن توجه ویژه‌ای داشته باشید. زیرا، نرمال‌سازی جداول موجب افزایش کارایی خواهد شد.

۲-۲. بانک اطلاعات SQL Server

سیستم بانک اطلاعات SQL Server، یکی از سیستم‌های بانک اطلاعات رابطه‌ای که قدرت فوق‌العاده‌ای دارد. نسخ مختلفی از SQL Server ارائه شده است و آنچه در این کتاب در مورد SQl Server گفته می‌شود، SQL Server 2008 می‌باشد.

---

[8]- NULL [2]- Normal Forms [3]- First Normal Form, Second Normal Form, Thrid Normal Form

[4]- Boyce- codd Normal

[۱۲]- اگر دو فیلد A و B در رابطه R داشته باشیم، آنگاه وابستگي تابعي $A \rightarrow B$ برقرار است اگر براي تمام رابطه‌ها در R، به ازاي هر مقدار A فقط يك مقدار B داشته باشيم.

این سیستم مدیریت بانک اطلاعات (DBMS[13])، در حدود ۷۰ درصد از بانک‌های اطلاعات موجـود را بـه خود اختصاص داده است، به طوری که برای سازمان‌های قدیمی و جدید، کوچک و بزرگ قابل استفاده اسـت. به همین دلیل، در این کتاب بانک اطلاعات SQL Server را بررسی می‌کنیم.

SQL Server ویژگی‌های مختلفی دارد که برخی از آنها عبارت‌اند از:

- **پشتیبانی از سخت‌افزارهای جدید:** این نسخه علاوه بر ایـن کـه CPUهـای قـدیمی، پردازشـگرهای جدیدی ٦٤ بیتی را نیز پشتیبانی می‌نماید. بنابراین، حافظه قابل دسترس و سرعت این نسـخه بانـک اطلاعـات افزایش می‌یابد.
- **پشتیبانی از انواع جدید:** این نسخه علاوه بر داده‌های معمولی، انواع داده از قبیـل XML و Varbinary را پشتیبانی می‌کند که XML برای ذخیره و بازیابی اسناد XML به کار می‌رود و نوع Varbinary، برای ذخیـره و بازیابی تصویر در فیلدهای جداول به کار می‌روند.
- **پشتیبانی از چند نمونه:** بانک اطلاعات SQL Server 2000 حداکثر تا ١٦ نمونه را می‌توانست پشتیبانی کند. ولی، این نسخه از ۵۰ نمونه می‌تواند پشتیبانی نماید.
- **پارتیشن‌بندی داده‌ها:** در این نسخه، جداول و شاخص‌های[14] بزرگ را می‌توان به چند پارتیشن تبـدیل نمود و در چندین پارتیشن مختلف قرار داد تا سریع‌تر بتوان به داده‌های خاصی دستیابی پیدا کرد.
- **بهبود سرویس‌های گزارش‌گیری:** یکی از ابزارهای بسیار مهم بانک اطلاعات، مدیریت گزارش‌گیـری است. این نسخه ابزار قدرت‌مندی برای تولید گزارش دارد.
- **بهبود در کاتالوگ:** کاتالوگ مکانی است که اشیای موجود در بانک اطلاعات را نگهداری مـی‌کنـد. در نسخه SQL Server 2000 و قبل از آن، کاتالوگ به عنوان بخشی از داده‌های اولیه ذخیره می‌شـدند، در حـالی که در این نسخه حدود ۲۵۰ دید[15] جهت نگهداری کاتالوگ در نظر گرفته شده است.
- **یک‌پارچگی با فریم‌ورک دات‌نت:** مهم‌ترین تحول نسخه‌های 2005 و 2008 یک‌پارچگی با فریم‌ورک دات‌نت است. این یک‌پارچگی امکان تولید رویه‌های ذخیره شده[16]، توابعی که توسط کاربر تعریف می‌شـوند، تریگرها، کرسرها و غیره را در یکی از زبان‌هـای برنامـه‌نویسـی از قبیـل Vb.NET، C#.NET، VC++.NET و J#.NET می‌دهد.

۳-۲. معرفی بانک اطلاعاتی نمونه

در این بخش یک بانک اطلاعاتی نمونه را معرفی می‌نمایم. سپس مراحل طراحی و پیاده‌سـازی ایـن بانـک اطلاعاتی را با هم مرور می‌کنیم. بانک اطلاعاتی که برای این کتاب در نظر گرفته شده است، اطلاعـات مربـوط به چک‌های صادر و چک‌های وصولی‌تان یا یـک سـازمان را نگهـداری مـی‌نمایـد. بـرای طراحـی ایـن بانـک اطلاعاتی، مراحل زیر باید انجام شود:

---

[13]- Database Management System [2]- Index [3]- View
[16]- Stored Procedures

۱. **تعیین اهداف بانک اطلاعاتی**: در این مرحله باید فرم‌ها، نمودارها، گزارشات و موارد دیگر مورد نیاز کاربران تعیین شود. در این مرحله، همچنین باید فرم‌ها، گزارشات و غیره طراحی، پیاده‌سازی و با کاربران مرور شود.

۲. **توزیع داده‌ها**: مهم‌ترین مرحله طراحی بانک اطلاعاتی، تعیین چگونگی توزیع داده است. در این مرحله باید تعیین گردد، چه داده‌هایی در چه جداولی توزیع شود. در این بخش باید تعداد جداول بانک اطلاعاتی، فیلدهای هر یک از جداول و نوع فیلدها تعیین شود. در این بانک اطلاعاتی نمونه چند جدول در نظر گرفته شده است که برخی از آنها عبارت‌اند از:

- **جدول Bank**: اطلاعات مربوط به بانک‌ها از قبیل شماره بانک، نام بانک، کد شهر بانک و غیره را نگهداری می‌کند (جدول۱-۲ را مشاهده کنید).
- **جدول Chek**: اطلاعاتی مربوط به چک‌های صادره و وارده از قبیل شماره چک، شماره بانک، تاریخ صدور، مبلغ چک، تاریخ وصول و غیره را نگهداری می‌نماید (جدول۱-۲ را ببینید).
- **جدول tblUser**: اطلاعات کاربران بانک اطلاعات را نگهداری می‌کند. این اطلاعات عبارت‌اند از: کد کاربر، نام کاربر، کلمه عبور، نام‌خانوادگی کاربر، سطح دسترسی و تصویر کاربر (جدول۱-۲ را مشاهده کنید).

۳. **تعیین ساختار و فیلدهای جدول**: بعد از این که فیلدهای جدول مشخص گردید، باید تعیین شود نوع هر فیلد چیست، آیا فیلد می‌توانند Null باشد یا خیر، چه فضایی از حافظه را اشغال می‌کنند. در این بخش باید تعیین کنیم هر جدول از چند فیلد تشکیل شده است و هر فیلد دارای چه خواصی (نام، اندازه، نوع و محدودیت‌ها) است. فیلدهای بانک اطلاعاتی نمونه در جدول۱-۲ آمده‌اند.

۴. **تعیین فیلد کلید اولیه[17] جداول**: یکی از بخش‌های مهم طراحی بانک اطلاعاتی تعیین فیلد کلید اولیه هر جدول است. فیلد کلید اولیه دارای سه خاصیت است که عبارت‌اند از:

۱. فیلد کلید اولیه نمی‌تواند تهی (Null) باشد. یعنی اگر فیلدی را به عنوان کلید اولیه انتخاب کنید. به طور خودکار محدودیت NOT NULL به آن تخصیص می‌یابد.

۲. فیلد کلید اولیه (یا کلید) تضمین می‌کند که هیچ دو رکوردی در جدول برای آن فیلد دارای مقدار یکسانی نمی‌باشند. یعنی، فیلد کلید از ورود رکوردهای تکراری در جدول جلوگیری می‌کند.

۳. اطلاعات جدول براساس فیلد کلید اولیه مرتب می‌شوند. به عنوان مثال، در بانک اطلاعاتی نمونه، در جدول tblUser، فیلد UserName، فیلد کلید اولیه است. زیرا، برای هیچ دو کاربری مقدار این فیلد برابر نمی‌باشد، یا به عبارت دیگر، هیچ دو کاربری حساب کاربری یکسانی ندارند. ولی در جدول Bank، فیلد BankCode فیلد کلید اولیه است.

۵. **تعیین ارتباط بین جدول‌ها**: یکی از ویژگی‌های اولیه بانک اطلاعات رابطه‌ای، ارتباطی است که جداول می‌توانند با یکدیگر داشته باشند. ارتباط بین جداول موجب می‌شود تا از افزونگی (تکرار بی‌مورد داده‌ها در جدول) جلوگیری شود. ارتباط بین جدول Chek و Bank با استفاده از فیلد BankCode برقرار می‌شود.

---

[17]- Primary Key

جدول ۱-۲ جداول بانک اطلاعات نمونه.			
نام جدول	نام فیلد	نوع	شرح
Bank	BankCode	Varchar(6)	کد بانک
	BankName	Varchar(50)	نام بانک
	CityCode	Varchar(6)	کد شهر بانک
	ShobeCode	Varchar(6)	کد شعبه بانک
	Tel	Varchar(15)	شماره تلفن بانک
	Address	Varchar(250)	آدرس بانک
Chek	ChekNo	Varchar(15)	شماره چک
	BackCode	Varchar(6)	کد بانک مربوط به چک
	ChekType	Varchar(3)	کد نوع چک
	Username	Varchar(10)	نام کاربر
	SodorDate	Varchar(10)	تاریخ صدور چک
	RasidDate	Varchar(10)	تاریخ سررسید چک
	VosolDate	Varchar(10)	تاریخ وصول
	Mablag	Numberic	مبلغ چک
	Comment	Varchar(250)	توضیحات
TblUser	UserName	Varchar(10)	حساب کاربری
	Pass	Varchar(10)	کلمه عبور
	Fuser	Varchar(15)	نام کاربر
	LUser	Varchar(20)	نام‌خانوادگی
	Access	Varchar(1)	نحوه دستیابی
	Picture	Varbinary	تصویر کاربر
City	CityCode	Varchar(6)	کد شهرستان
	CityName	Varchar(30)	نام شهرستان
	OstanCode	Varchar(6)	کد استان
Ostan	OstanCode	Varchar(6)	کد استان
	OstanName	Varchar(30)	نام استان
Shobe	ShobeCode	Varchar(6)	کد شعبه
	ShobeName	Varchar(30)	نام شعبه

۶. تعیین اشیای دیگر بانک اطلاعات: بانک اطلاعاتی از اشیای متعددی تشکیل شده است. مهم‌ترین شیء بانک اطلاعاتی جدول است. جدول برای نگهداری داده‌های بانک اطلاعاتی به کار می‌رود. اشیای دیگر از قبیل دیدها[۱۸]، توابع[۱۹]، رویه‌های ذخیره شده[۲۰]، کرسرها[۲۱]، تریگرها[۲۲] و غیره در بانک اطلاعاتی وجود دارند.

[1]- Views [2]- Functions [20]- Stored Procedures 4 - Cursors [5]- Triggers

در ادامه کتاب با این اشیاء آشنا خواهید شد و چگونگی ایجـاد، حـذف و ویـرایش ایـن اشـیا را در بانـک اطلاعاتی می‌آموزیم.

فصل ۳۳ بخشی از ایجاد بانک اطلاعات و جداول آن

در فصل‌های ۱ و ۲ با مفاهیم اولیه #C و بانک اطلاعات آشنا شدیم. در این فصل می‌خواهیم یک بانک اطلاعات را از طریق برنامه #C 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
۴. 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 #C ایجاد کرده و جداول آن 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 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، 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 filespace …	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 اجرای دستورات ALTER DATABASE و DROP DATABASE، اگر بانک اطلاعات که می‌خواهیم خواص آن را تغییر دهیم یا آن را حذف کنیم، موجود نباشد، پیام خطا ظاهر می‌شود. ولی در هنگام اجرای دستور CREATE DATABASE چنانچه بانک اطلاعات که می‌خواهید ایجاد کنید از قبل موجود باشد، پیام خطا ظاهر خواهد شد.

۲-۳. اشیای بانک اطلاعات

بانک اطلاعات برای اهداف تجاری طراحی شده است که با ایجاد بانک اطلاعات اهداف تجاری آن برآورده نمی‌شود. برای اینکه اهداف تجاری بانک اطلاعات برآورده شود، باید اشیای بانک اطلاعات را به آن اضافه 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 به کار می‌رود. ارتباط بین جداول می‌تواند از افزونگی داده‌ها جلوگیری کند.

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

۳. **شیء Views**: از طریق یک یا چند جدول می‌توان جدول‌های مجازی دیگری را ایجاد کرد تا بتوان با توجه به نیاز کاربران، فیلدهای خاصی را در اختیار آنها قرار داد. به جداول مجازی که در بانک اطلاعات ایجاد می‌کنید، **دید**[۲۳] گویند. این شیء برای نگهداری دیدهایی که از جداول ایجاد شده‌اند، به کار می‌رود.

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

۵. **شیء Programmability**: برای ذخیره کردن اشیایی از قبیل رویه‌های ذخیره شده و توابع به کار می‌رود. در فصل‌های بعد با مفهوم توابع و رویه‌های ذخیره شده بیشتر آشنا خواهید شد و چگونگی استفاده از آنها را در C#.NET خواهید دید.

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

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

---

[23]- View

در ادامه با نحوه ایجاد اشیا بانک اطلاعات آشنا خواهید شد و چگونگی به کارگیری آنها را در #C 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 جدول بپردازیم، باید خواص فیلدها را بررسی کنیم. این خواص در زیر آمده‌اند:

- **نام فیلد:** هر فیلد در جدول دارای یک نام یکتا[۲۴] است که از قانون نام‌گذاری شناسه‌ها در بانک اطلاعات پیروی می‌کند.
- **نوع فیلد:** برای هر فیلد باید تعیین شود، اولاً چه نوع داده‌ای را می‌تواند ذخیره کند، ثانیاً تعداد بایت‌هایی که می‌تواند ذخیره کند، چقدر است. انواع داده‌ها و مقدار فضایی که هر یک از انواع در بانک اطلاعات نیاز دارند، در جدول ۳-۳ آمده است.
- **محدودیت‌های فیلد:** این ویژگی تعیین می‌کند هر فیلد دارای چه محدودیت‌هایی (قیدهای) است که برخی از این محدودیت‌ها در زیر آمده‌اند:

۱. فیلد کلید اولیه[۲۵]: 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[۲۶] باشد.

۲. فیلد کلید خارجی[۲۷]: برای ارتباط بین دو جدول به کار می‌رود. یعنی، با استفاده از کلید اولیه یک 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 فیلدی مقدار جای خالی[۲۸] وارد کنید، مقدار آن فیلد تهی نیست.

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

۵. 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.

۶. 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.

---

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

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

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

جدول ۳-۳ انواع داده در SQL Server.

نام	تعداد بایت	هدف
Int	4	اعداد صحیح از ۲/۱٤۷/٤۸۳/٦٤۸- تا ۲/۱٤۷/٤۸۳/٦٤۷ است.
ntbigi	8	اعداد صحیح از $۲^{۶۳}$- تا ۱- $۲^{۶۳}$ است.
binary(n)	n	داده‌ها دودویی حداکثر ۲۵۵ بایت را اشغال می‌کنند.
bit	1	مقادیر ۰ یا ۱ را می‌پذیرد.
char(n)	n	حداکثر تا ۸۰۰۰ کارکتر را می‌پذیرد و به ازای هر کارکتر یک بایت را اشغال می‌کند.
datatime	8	تاریخ و زمان را نگهداری می‌کند.
decimal(p,s)	حداکثر17	اعداد اعشاری یا صحیح۱ و $۱۰^{۳۸}$- تا $۱۰^{۳۸}$ را نگهداری می‌کند.
float(n)	8	از مقدار $۱۰۷۹E^{۳۸}$- تا $۱۰۷۹E^{۳۸}$را نگهداری‌می‌کند و دقت‌آن تا ۱۵ رقم است.
real(n)	4	تقریباً از مقدار $۳/۴۰E^{۳۸}$- تا $۳/۴۰E^{۳۸}$ را نگهداری می‌کند (با تقریب ۷ رقم).
Image		داده‌های تصویر تا 2GB را در صفحات ۸ کیلوبایتی ذخیره می‌کند.
money	8	داده‌های‌ارزی از ۹۲۲۳۳۷۲۰۳٦۸۵٤۷۷/۵۸۰۸-تا۹۲۲۳۳۷۲۰۳٦۸۵٤۷۷/۵۸۰۷ است.
varchar(n)	n*2	داده‌های یونیکد از ۱ تا ٤۰۰۰ کارکتر را نگهداری می‌کند.
ntext	n*2	داده‌های یونیکد با طول متغیر که طول آن ۱۰۷۳۷٤۱۸۲۳ است.
numeric	حداکثر17	مترادف Decimal است.
real	4	مترادف Float است.
smalldatetime	4	ترکیب تاریخ و زمان با طول کوتاه را نگهداری می‌کند.
smallint	2	اعداد صحیح از ۳۲,۷٦۸- تا ۳۲,۷٦۷ را نگهداری می‌کند.
smalllong	8	مقادیر پولی با حداکثر ٤ رقم اعشار را نگهداری می‌کند.
sysname(n)	n*2	برای ذخیره کردن اسامی اشیای سیستم به کار می‌رود و به ازای هـر کـارکتر دو بایت را اشغال می‌کند.
timestamp		برای نگهداری مهر زمانی در بانک اطلاعاتی به کار می‌رود.
tinyint(n)	n	اعداد صحیح ۰ تا ۲۵۵ بایت که n تعداد بایت‌ها را مشخص می‌کند.
varbinany(n)	n	الگوی بیتی تا ۲۵۵ بایت را نگهداری می‌کند.
uniquidentifier	16	عدد ۱٦ بایتی هگزا دسیمال که شناسه یکتای عمومی را نشان می‌دهد.

XML	حداکثر2GB	برای نگهداری اسناد XML به کار می‌رود.
varbinary(max)		اطلاعات بایتی را به صورتی در نظر می‌گیرد که می‌توانید تصاویر را در آن ذخیره کنید.

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

۱. ایجاد جدول با روش طراحی ۲. 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:

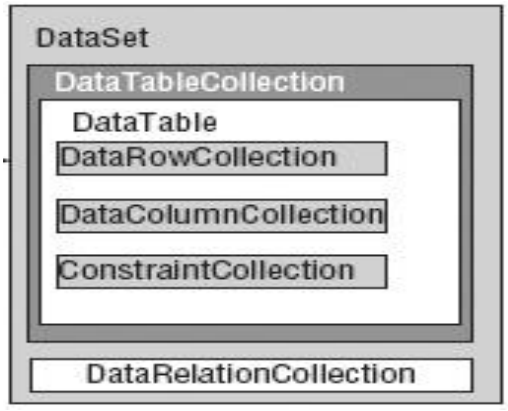
- **database_name**: نام بانک اطلاعات را تعیین می‌کند که جدول در آن ایجاد می‌گردد.
- **schema_name**: نام شمای بانک اطلاعات را تعیین می‌کند که جدول در آن ایجاد می‌شود.
- **table_name**: نام جدولی را تعیین می‌کند که باید ایجاد شود.
- **column_definition**: تعریف ستون‌های جدول قرار می‌گیرد.
- **computed_column_definition**: 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.

فصل ۴ بخشی از 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[29] 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.

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.

---

[29]- Query

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

جدول ۲-٤ خواص DataSet.

خاصیت	هدف
Casesentsive	تعیین‌می‌کند که آیا درهنگام‌مقایسه‌بین‌حروف‌بزرگ یا کوچک فرق بگذارد یا خیر.
DataSetName	نام DataSet را تعیین می‌کند.
DafaultViewManager	فیلتر کردن یا مرتب‌سازی پیش‌فرض مربوط به DataSet را تعریف می‌کند.
EnforceConstrains	تعیین می‌کند که آیا در هنگام تغییر رکوردها، محدودیت‌ها تعریف شده اعمال شود یا خیر.

ادامه جدول ۲-٤ خواص DataSet.

خاصیت	هدف
ExtendedProperties	اطلاعات و خواص سفارشی کاربر را تعیین می‌کند.
HasErrors	تعیین می‌کند که آیا رکوردی از DataRowهای مربوط به DataSet دارای خطاهایی هستند یا خیر.
Locale	اطلاعات محلی (Locale information) را در هنگام مقایسه رشته‌ها تعیین می‌کند (به عنوان مثال، مقایسه از چپ به راست یا از راست به چپ باشد).
NameSpace	فضای نامی که استفاده می‌شود وقتی که در سند XML نوشته یا از آن خوانده می‌شود.
Prefix	یک پیشوند XML که به عنوان نام مستعاری برای فضای نام استفاده می‌شود.
Relations	یک کلکسیون از اشیای DataRelation است که ارتباط بین DataTableها را تعیین می‌کنند.
Tables	کلکسیونی از DataTableهای موجود در DataSet است. نام این Tableها به طور پیش‌فرض Table، Table1، Table2 و ... است. متدهایی برای اضافه کردن جداول به DataSet وجود دارند که در جدول ٤-٤ آمده است.

جدول ۳-٤ متدهای DataSet.

متد	هدف
Clear()	محتویات DataSet را پاک می‌نماید.
Clone()	ساختارهای موجود در DataSet از قبیل ساختار جداول (DataTable)، ارتباط بین جداول و محدودیت‌ها را کپی می‌کند. از داده‌های موجود در جداول کپی نمی‌گیرد.
Copy()	ساختار و داده‌های موجود در جداول DataSet را کپی می‌گیرد.
AcceptChanges()	تمام تغییرات انجام شده در DataSet از زمان بار شدن یا آخرین بار اجرای متد AcceptChanges یا RejectChanges تاکنون را ثبت می‌کند.
GetChanges()	یک‌کپی‌از تمام تغییرات‌انجام شده در DataSet از زمان بار شدن برمی‌گرداند.
HasChanges()	مقداری را برمی‌گرداند که نشان می‌دهد آیا تغییراتی مانند حذف، اضافه و ویرایش سطرها (رکوردها) در DataSet انجام شده است یا خیر.
GetChildRelations()	در ارتباط بین دو جدول فیلد ارتباط جدول فرزند را برمی‌گرداند.
GetParentRelations()	در ارتباط بین دو جدول فیلد ارتباط جدول پدر را برمی‌گرداند.
Merge()	این متد دو DataSet را با هم ادغام می‌کند.
RejectChanges()	تغییرات انجام شده در DataSet را از آخرین زمان اجرای AcceptChanges عقب‌گرد (خنثی) می‌نماید.
ReadXML()	فایل XML را خوانده در DataSet قرار می‌دهد.
ReadXMLSchema()	یک ساختار شِما XML را می‌خواند و در DataSet می‌نویسد.
WriteXML()	داده‌های موجود در DataSet را در فایل XML می‌نویسد.
WriteXMLSchema()	ساختار DataSet را به عنوان یک شِمای XML می‌نویسد.
Reset()	این متد DataSet را به حالت اولیه‌اش برمی‌گرداند.

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 ا ضافه 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.

۴-۱-۱. 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:

```
String connStr = "رشته اتصال";
SqlConnection conn = new SqlConnection(connStr);
```

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

```
String SQL = "SELECT * FROM Chek";
```

۳. نمونه‌ای از کلاس SqlDataAdapter تعریف کنید:

```
SqlDataAdapter da = new SqlDataAdapter (SQL , Conn);
```

۴. شیء DataSet را ایجاد کنید:

```
DataSet ds = 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");
```

۴-۱-۲. 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:

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

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

۲-۲-۴. کلاس DataRow

بیان گردید که هر جدول از چندین سطر تشکیل شده است و هر شیء DataRow سطری از آن جدول را نشان می‌دهد. بنابراین، هر جدول مجموعه‌ای از DataRowها است. با استفاده از نام و اندیس[۳۰] سطر می‌توان به رکورد خاصی دست یافت و با استفاده از اندیس ستون می‌توان محتویات فیلد خاصی را بازیابی یا دستکاری نمود. با این شیء می‌توان سطر جدیدی به جدول اضافه کرد، سطرهای موجود جدول را حذف و ویرایش نمود. خواص و متدهای شیء DataRow در جدول۱۵-٤ و ١٦-٤ آمده‌اند.

جدول ۱۵-٤ خواص DataRow.

خاصیت	هدف
HasError	تعیین می‌کند آیا خطای در سطر وجود دارد یا خیر.
Item	داده‌های ستون (فیلد) خاصی را تعیین می‌کند. خواص مربوط به Item در جدول۱۷-٤ آمده است.
ItemArray	کلیه مقادیر ستون‌های سطر را در یک آرایه قرار می‌دهد یا برعکس.
RowError	توصیف خطای سفارشی برای یک سطر را مشخص می‌کند.
RowState	حالت یک سطر (DataRowState) را مشخص می‌کند.
Table	جدولی را تعیین می‌کند که این سطر (DataRow) در آن قرار دارد.

جدول ١٦-٤ متدهای DataRow.

متد	هدف

[30]- Index

AcceptChanges	تمام تغییرات انجام شده در یک DataRow را تثبیت (اعمال) می‌کند.
BeginEdit	یک عملیات ویرایش را شروع می‌کند.
CancelEdit	یک عملیات ویرایش را کنسل می‌کند.
Delete	سطری را حذف می‌کند (علامت‌گذاری کرده تا DataSet به هنگام گردد).
EndEdit	یک عملیات ویرایش را با اعمال تغییرات خاتمه می‌دهد.
GetChildRows	تمام سطرهای جدول فرزند مربوط به DataRow را می‌خواند.
GetParentRows	تمام سطرهای جدول پدر مربوط به یک DataRow را می‌خواند.
HasVersion	تعیین می‌کند که آیا یک نسخه مشخص شده از DataRow وجود دارد یا خیر.
IsNull	تعیین می‌کند که آیا ستون مشخصی Null است یا خیر.
RejectChanges	تمام تغییرات انجام شده در DataRow را عقب‌گرد می‌کند.
SetParentRow	سطر پدر مربوط به یک DataRow را تعیین می کند.
ClearErrors	خطاهای سطر را پاک می‌کند.
GetColumnError	خطاهای ستون را می‌خواند.

جدول ۱۷-٤ خواص DataRow Item.

خاصیت	هدف
Item(columnName)	مقدار ستونی را برمی‌گرداند که نام ستون آن باcolumnName مشخص شده است.
Item(dataColumn)	مقدار مربوط به dataColumn مشخص شده را برمی‌گرداند.
Item(columnIndex)	مقدار ستونی را برمی‌گرداند که شماره ستون توسط columnIndex مشخص شده است (columnIndex از صفر شروع می‌شود. یعنی، اولین ستون دارای columnIndex صفر است).
Item(columnName, rowVersion)	مقدار مربوط به نسخه rowVersionای را برمی‌گرداند که خاصیت ColumnName آن با رشته columnName مشخص شده است.
Item(dataColumn, rowVersion)	مقدار مربوط به نسخه rowVersionای را برمی‌گرداند که با dataColumn مشخص شده است.
Item(columnIndex, rowVersion)	مقدار نسخه rowVersionای را برمی‌گردارند که ستون آن توسط columnIndex مشخص شده باشد.

مثال۱-٤

برنامه‌ای که اعمال زیر را انجام می‌دهد:

۱. با استفاده از کلاس DataRow مقادیر جدول tblUser را در ListBox نمایش می‌دهد.

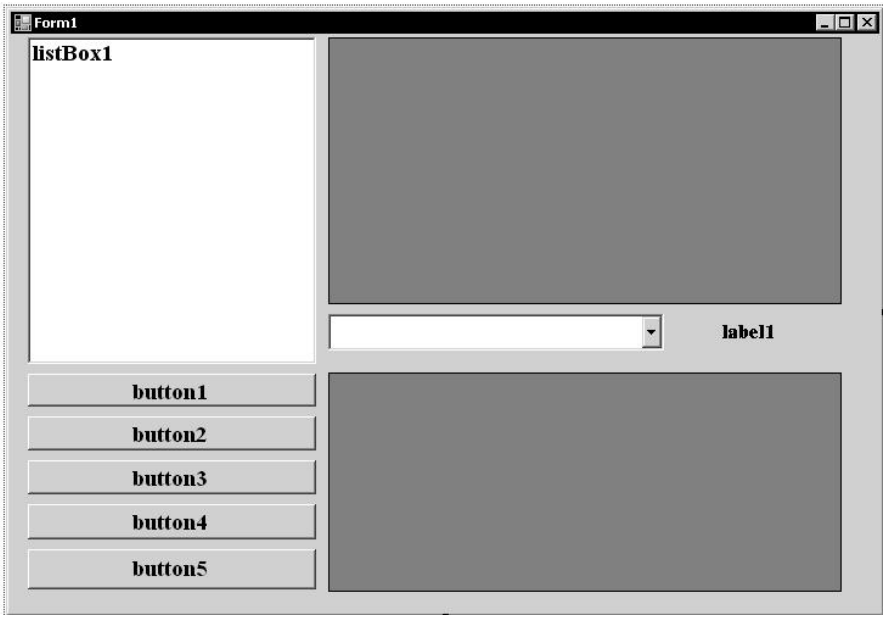
۲. با استفاده از کلاس DataColumn نوع هر فیلد جدول tblUser را نمایش می‌دهد.

۳. با استفاده از کلکسیون Relations ارتباط بین جدول City و Ostan را برقرار می‌کند.

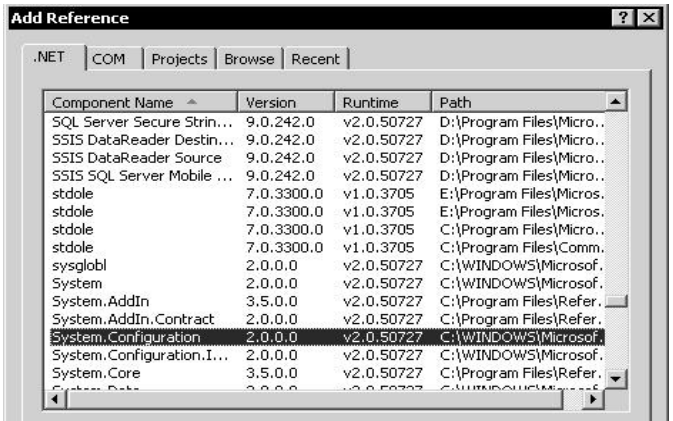
۴. اطلاعات فیلدهای رشته، تصویر و ComboBox را در یک DataGridView نمایش می‌دهد.
۵. نسخه‌های اصلی رکوردهای جدول City را در dataGridView1 و اطلاعات نسخه‌های مختلف آن را در dataGridView2 نمایش می‌دهد.
۶. رشته اتصال را در فایل App.Config قرار می‌دهد.

مراحل طراحی و اجرا

۱. **پروژه جدیدی به نام Field ایجاد کنید.**
۲. **یک کنترل ListBox به فرم اضافه کنید. این کنترل برای نمایش مقادیر جدول، لیست فیلدها و نوع آنها به کار می‌رود.**
۳. **یک کنترل Label و یک کنترل ComboBox به فرم اضافه کنید. کنترل ComboBox برای انتخاب نسخه‌های مختلف جدول City که باید در dataGridView2 نمایش داده شوند، استفاده می‌شود.**
۴. **دو کنترل DataGridView به فرم اضافه کنید. کنترل اول، برای نمایش اطلاعات اصلی جداول و کنترل دوم، برای نمایش نسخه‌های مختلف جدول City به کار می‌رود.**
۵. **پنج کنترل Button به فرم اضافه کنید. هر یک از این کنترل‌ها برای انجام عمل خاصی به کار می‌روند.**

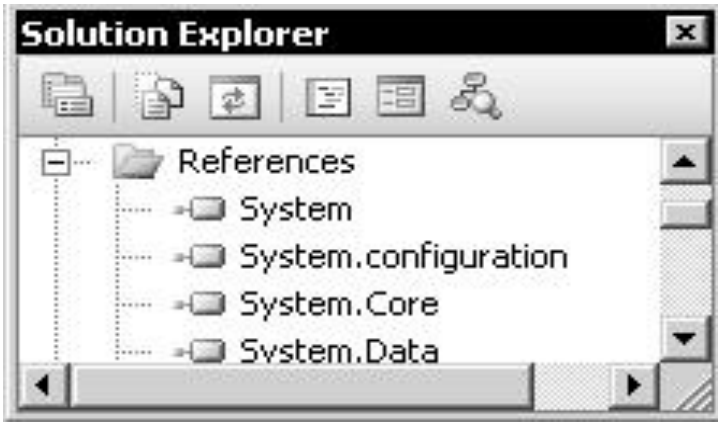


۶. **ارجاع System.Configuration را به پروژه اضافه کنید. برای این منظور گزینه، Project/Add Reference... را اجرا کنید تا پنجره Add Reference ظاهر شود (شکل۲-٤).**



شکل۲-٤ پنجره Add Reference

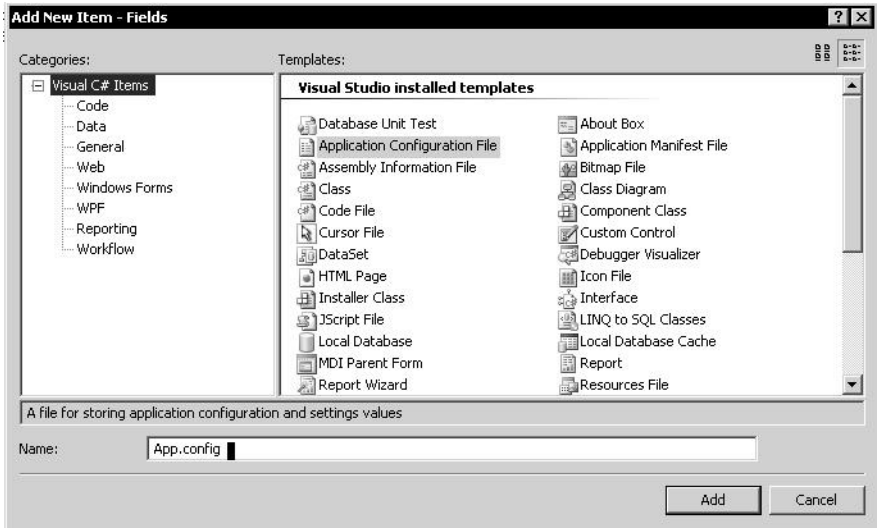
۷. **در این گزینه System.Configuration را انتخاب کرده، دکمه OK را کلیک کنید تا شکل زیر ظاهر شود (پنجره Solution Explorer):**



همان‌طور که در این شکل می‌بینید، ارجاع System.Configuration به پروژه اضافه گردید.

۸. **فایل App.Config را به پروژه اضافه کنید تا بتوانید، رشته اتصال را در آن تعریف کنید. برای این منظور، مراحل زیر را اجرا کنید:**

- گزینه Project/ Add New Items را اجرا کنید تا پنجره Add New Item ظاهر شود (شکل۳-٤).



شکل۳-٤ پنجره Add New Item

- در بخش Templates، گزینه Application Configuration File را انتخاب و دکمه Add را کلیک کنید تا فایل App.Config به پروژه اضافه شود (شکل زیر):

- بر روی این فایل(App.Config) کلیک مضاعف کرده، دستورات آن را به صورت زیر تغییر دهید:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <connectionStrings>
 <add name="Chek"
       connectionString="server=.\sqlexpress;database=Chek;Integrated
      Security=SSPI" />
  </connectionStrings>
</configuration>
```

این دستورات رشته اتصال را تعریف می‌کنند (اتصال به بانک Chek را برقرار می‌کند که بر روی کامپیوتر فعلی قرار دارد).

۹. به بخش using فرم برنامه برگردید و دستورات زیر را به آن اضافه کنید:

```
using System.Data.SqlClient;
using System.Configuration;
```

دستور اول، برای استفاده از کلاس‌های SQL Server اضافه شده است و دستور دوم، برای به کارگیری کلاس‌های Configuration اضافه گردید.

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

```
private void Form1_Load(object sender, EventArgs e)
{
   listBox1.MultiColumn = true;
   label1.Text = "فیلتر کردن";
   button1.Text = " نمایش محتوي فیلدها";
   button2.Text = " نمایش لیست فیلدها";
   button3.Text = "  ایجاد ارتباط ";
   button4.Text = " نمایش انواع فیلدها";
   button5.Text = " فیلتر سازي براساس نسخه";
   comboBox1.Items.Add("Added");
   comboBox1.Items.Add("CurrentRows");
   comboBox1.Items.Add("Deleted");
   comboBox1.Items.Add("ModifiedCurrent");
   comboBox1.Items.Add("ModifiedOriginal");
   comboBox1.Items.Add("None");
   comboBox1.Items.Add("OriginalRows");
   comboBox1.Items.Add("Unchanged");
}
```

دستور اول، کنترل ListBox1 را چند ستونی در نظر می‌گیرد. پنج دستور بعدی، عنوان دکمه‌های روی فرم را تعیین می‌کنند و دستورات بعدی، حالت‌های مربوط به نسخه‌های مختلف رکوردها را به comboBox1 اضافه می‌کنند.

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

```
private void button1_Click(object sender, EventArgs e)
{
   using (SqlConnection conn = new SqlConnection
            (ConfigurationManager.ConnectionStrings["Chek"].
               ConnectionString))
   {
    listBox1.Items.Clear();
    // query
    string sql = @"select * from tblUser";
    try
    {
      conn.Open();
      SqlDataAdapter da = new SqlDataAdapter(sql, conn);
      DataSet ds = new DataSet();
      da.Fill(ds, "tblUser");
      DataTable dt = ds.Tables["tblUser"];
      // display data
      foreach (DataRow row in dt.Rows)
      {
         foreach (DataColumn col in dt.Columns)
```

```
            listBox1.Items.Add(col.ColumnName + ":" + row[col]);
          listBox1.Items.Add("\n");
       }
    }
    catch (Exception e1)
    {
       MessageBox.Show("Error: " + e1.ToString());
    }
    finally
    {
       conn.Close();
    }
  }
}
```

دستور using، رشته اتصال را از فایل App.Config خوانده و یک نمونه از کلاس SqlConnection به نام conn ایجاد می‌نماید. دستورات داخل بلاک try، ابتدا اتصال conn را باز کرده، اطلاعات جدول tblUser را در شیء da از نوع SqlDataAdapter قرار می‌دهد و یک شیء از نوع DataSet به نام ds ایجاد می‌کند و اطلاعات جدول tb1User را در ds پر می‌کند. در ادامه با کلاس DataTable یک جدول به نام dt ایجاد کرده، اطلاعات جدول tblUser که در ds است، در آن قرار می‌دهد. در پایان، حلقه foreach رکوردهای جدول با خاصیت dt.Rows و ستون‌های جدول را با dt.Columns نمایش می‌دهد.

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

```
private void button2_Click(object sender, EventArgs e)
{
   string sql= @"select * from tblUser";
   using (SqlConnection conn = new SqlConnection
           (ConfigurationManager.ConnectionStrings["Chek"].
           ConnectionString)) {
   try {
     conn.Open();
     SqlDataAdapter da = new SqlDataAdapter(sql, conn);
     SqlCommand cmd = new SqlCommand(sql, conn);
     SqlDataReader rdr = cmd.ExecuteReader();
     DataTable schema = rdr.GetSchemaTable();
     listBox1.Items.Clear();
     foreach (DataRow row in schema.Rows) {
       foreach (DataColumn col in schema.Columns) {
          listBox1.Items.Add(col.ColumnName +" :" + row[col]);
      listBox1.Items.Add("");
    }
    listBox1.Items.Add("\n");
    rdr.Close();
    }
   }
   catch (Exception e1)
   {
      MessageBox.Show("Error Occurred: " + e1.ToString());
   }
   finally
```

```
    {
        conn.Close();
    }
  }
}
```

این دستورات، ابتدا با استفاده فایل App.Config یک شیء از نوع SqlConnection به نام conn ایجاد می‌کنند. سپس، اطلاعات جدول tblUser را خوانده (با متد ()ExecuteReader) در rdr شیءای از نوع SqlDataReader می‌باشد، قرار می‌دهند. در ادامه، با فراخوانی متد ()GetSchemaTable، ساختار جدول را خوانده، در شیء Schema از نوع DataTable قرار می‌دهد. سپس، در حلقه foreach برای هر سطر (schema.Rows) که خاصیت Columns آنها خواص فیلدها را نگهداری می‌کنند، به listBox1 اضافه می‌کنند (یعنی، نام فیلدها و خواص آنها را در listBox1 اضافه می‌کنند).

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

```
private void button3_Click(object sender, EventArgs e)
{
   string City = "SELECT * FROM City";
   string Ostan = "SELECT * FROM Ostan";
   dataGridView1.Columns.Clear();
   dataGridView1.DataBindings.Clear();
   using (SqlConnection con = new SqlConnection
               (ConfigurationManager.ConnectionStrings["Chek"].
               ConnectionString))
   {
      DataSet ds = new DataSet();
      SqlDataAdapter da = new SqlDataAdapter(Ostan, con);
      da.Fill(ds, "Ostan");
      da = new SqlDataAdapter(City, con);
      da.Fill(ds, "City");
      ds.Relations.Add("OstanCity",
      ds.Tables["Ostan"].Columns["OstanCode"],
      ds.Tables["City"].Columns["OstanCode"]);
      dataGridView1.DataBindings.Clear();
      dataGridView1.AutoGenerateColumns = true;
      dataGridView1.DataSource = ds.Tables["City"];
   }
}
```

این دستورات. به بانک اطلاعات Chek متصل شده اطلاعات جداول City و Ostan را در ds که از نوع DataSet هستند، پر می‌کنند و با کلاس Relations ارتباط بین این دو جدول را برقرار می‌کنند و اطلاعات جدول City را در کنترل dataGridView1 نمایش می‌دهند.

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

```
private void button4_Click(object sender, EventArgs e)
{
   string select = "SELECT FUser, LUser, Photo, IsNull(Access,0) as
          AccessTO FROM tblUser";
   using (SqlConnection conn = new SqlConnection
          (ConfigurationManager.ConnectionStrings["Chek"].ConnectionString))
   {
      SqlDataAdapter da = new SqlDataAdapter(select, conn);
      DataSet ds = new DataSet();
      da.Fill(ds, "tblUser");
      select = "SELECT  *   FROM Access ";
      da = new SqlDataAdapter(select, conn);
      da.Fill(ds, "Access");
      // Construct the columns in the grid view
      SetupColumns(ds);
      dataGridView1.AutoGenerateColumns = false;
      dataGridView1.DataSource = ds.Tables["tblUser"];
      dataGridView1.AutoResizeRows(DataGridViewAutoSizeRowsMode.
           AllCells);
    }
}
```

ایـن رویـداد، اطلاعـات جـدول tblUser از قبیـل فیلـدهای Text، تصـویر و ComboBox را بـر روی dataGridView نمایش می‌دهد. برای این منظور، اطلاعات جـدول tblUser و اطلاعـات جـدول Access را در شـیء ds از نـوع DataSet پـر مـی‌کنـد. سـپس، بـا فراخـوانی تـابع SetupColumns(ds) سـتون‌هـای کنتـرل dataGridView1 را سفارشی مـی‌کنـد و در پایـان، اطلاعـات جـدول tblUser را در dataGridView1 نمـایش می‌دهد.

۱۵. تابع SetupColumns() را به صورت زیر تعریف کنید:

```
private void SetupColumns(DataSet ds)
{
   dataGridView1.Columns.Clear();
   dataGridView1.DataBindings.Clear();
   // add a string column
    DataGridViewTextBoxColumn forenameColumn = new DataGridViewTextBoxColumn();
   forenameColumn.DataPropertyName = "FUser";
   forenameColumn.HeaderText = "نام کاربر";
   forenameColumn.ValueType = typeof(string);
   forenameColumn.Frozen = true;
   dataGridView1.DataBindings.Clear();
   dataGridView1.Columns.Add(forenameColumn);
   // add a string column
    DataGridViewTextBoxColumn surnameColumn = new DataGridViewTextBoxColumn();
    surnameColumn.DataPropertyName = "LUser";
    surnameColumn.HeaderText = "نام خانوادگي كاربر";
    surnameColumn.Frozen = true;
    surnameColumn.ValueType = typeof(string);
    dataGridView1.Columns.Add(surnameColumn);
    // add a Image column
    DataGridViewImageColumn photoColumn = new DataGridViewImageColumn();
    photoColumn.DataPropertyName = "Photo";
    photoColumn.Width = 200;
```

```
    photoColumn.HeaderText = "تصویر";
    photoColumn.ReadOnly = true;
    photoColumn.ImageLayout = DataGridViewImageCellLayout.Normal;
    dataGridView1.Columns.Add(photoColumn);
    // add a ComboBox column
    DataGridViewComboBoxColumn AccessColumn = new DataGridViewComboBoxColumn();
     AccessColumn.HeaderText = "سطح دسترسي";
    AccessColumn.DataSource = ds.Tables["Access"];
    AccessColumn.DisplayMember = "AccessName";
    AccessColumn.ValueMember = "Access";
    AccessColumn.DataPropertyName = "Access";
    dataGridView1.Columns.Add(AccessColumn);
}
```

این تابع، یک پارامتر از نوع DataSet را به عنوان ورودی گرفته کـه دارای اطلاعـات دو جـدول tblUser و Access است. در بخش اول و دوم، فیلدهای Fuser و Luser از نـوع String را بـه دو سـتون dataGridView1 اضافه می‌کند. در بخش سوم، فیلد Photo از نوع تصاویر را به ستون بعدی dataGridView1 اضافه می‌کند و در بخش آخر، اطلاعات جدول Access (نام دسترسی) را به صورت ComboBox به یک سـتون dataGridView1 اضافه می‌نماید. در این ستون کاربر می‌تواند انواع دسترسی‌ها را ببیند و یکی را انتخاب کند.

۱۶. دکمه button5 را کلیک مضاعف کرده، دستورات رویداد Click آن را به صـورت زیـر تغییـر دهید:

```
private void button5_Click(object sender, EventArgs e)
{
   dataGridView1.Columns.Clear();
   dataGridView1.DataBindings.Clear();
   using (SqlConnection conn = new SqlConnection
          (ConfigurationManager.ConnectionStrings["Chek"].ConnectionString))
   {
       string select = "SELECT * FROM City";
       SqlDataAdapter da = new SqlDataAdapter(select, conn);
       DataSet ds = new DataSet();
       da.Fill(ds, "City");
       dataGridView1.AutoGenerateColumns = true;
       dataGridView1.DataSource = ds.Tables["City"];
       DataView dv = new DataView(ds.Tables["City"]);
       dataGridView2.AutoGenerateColumns = true;
       dataGridView2.DataSource = dv;
       comboBox1.SelectedIndex = 6;
       comboBox1.Enabled = true;
   }
}
```

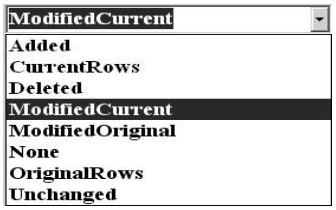
این رویداد، ابتدا یک شیء conn از نوع SqlConnection ایجاد کرده، سـپس اطلاعـات مربـوط بـه نسـخه انتخاب شده جدول City را در کنترل dataGridView2 نمایش می‌دهد. زیرا، اطلاعات نسخه‌های جـدول City در شیء dv از نوع DataView قرار می‌گیرد (با کلاس DataView در فصل بعدی بیشتر آشنا خواهید شد).

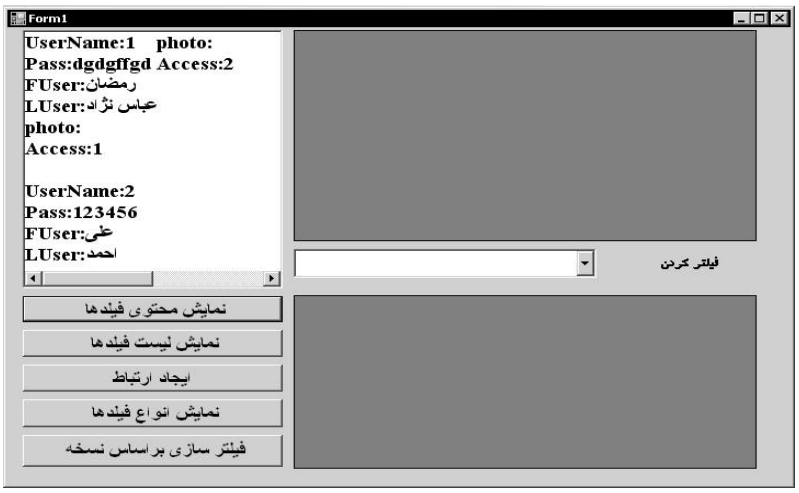
۱۷. بر روی کنترل comboBox1 کلیک مضاعف کرده، دستورات رویـداد SelectedIndexChanged آن را به صورت زیر تغییر دهید:

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
   DataViewRowState state;
   switch (comboBox1.Text)
   {
      case "Added":
         state = DataViewRowState.Added;
         break;
      case "CurrentRows":
         state = DataViewRowState.CurrentRows;
         break;
      case "Deleted":
         state = DataViewRowState.Deleted;
         break;
      case "ModifiedCurrent":
         state = DataViewRowState.ModifiedCurrent;
         break;
      case "ModifiedOriginal":
         state = DataViewRowState.ModifiedOriginal;
         break;
      case "None":
         state = DataViewRowState.None;
          break;
      case "OriginalRows":
          state = DataViewRowState.OriginalRows;
          break;
       case "Unchanged":
          state = DataViewRowState.Unchanged;
          break;
       default:
          state = DataViewRowState.OriginalRows;
          break;
      }
      try
      {
         ((DataView)dataGridView2.DataSource).RowStateFilter = state;
      }
      catch (Exception ex)
      {
         MessageBox.Show(ex.ToString());
      }
}
```

این دستورات، ابتدا یک متغیری به نام state از نوع DataViewRowState تعریف می‌کنند. سپس، بـا توجـه به مقدار خاصیت Text کنترل comboBox1 مقـدار متغیـر state را یکـی مقـادیر DataViewRowState.Added، DataViewRowState.CurrentRows، DataViewRowState.ModifiedOriginal و غیره در نظـر مـی‌گیرنـد. در ادامه، در بلاک try، خاصیت RowStateFilter مربوط به کنترل dataGridView2.DataSource را برابر state قرار می‌دهند تا نسخه خواسته شده جدول City نمایش داده شود.

۱۸. **برنامه را ذخیره و اجرا کنید. اکنون فرم اجرای برنامه ظاهر می‌شود. در این فرم دکمه «نمـایش محتوی فیلدها» را کلیک کنید تا شکل٤-٤ ظاهر گردد.** در این شکل، نام فیلد و مقـدار آن را مـی‌بینیـد. اکنون «**دکمه نمایش لیست فیلدها**» را کلیک کنید تا لیست فیلدها و نوع آنها را در listBox1 ببینید. دکمه «**فیلترسازی براساس نسخه**» را کلیک کنید تا شکل٥-٤ ظاهر گـردد. اکنـون در کنتـرل dataGridView2، شهر بابل را به بابلسر تغییر دهید و comboBox1 را کلیک نموده تا گزینه‌های آن مانند شـکل زیـر ظـاهر شود:

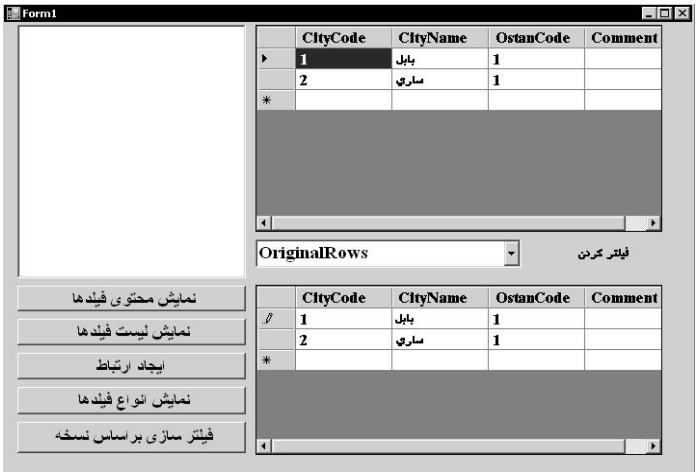




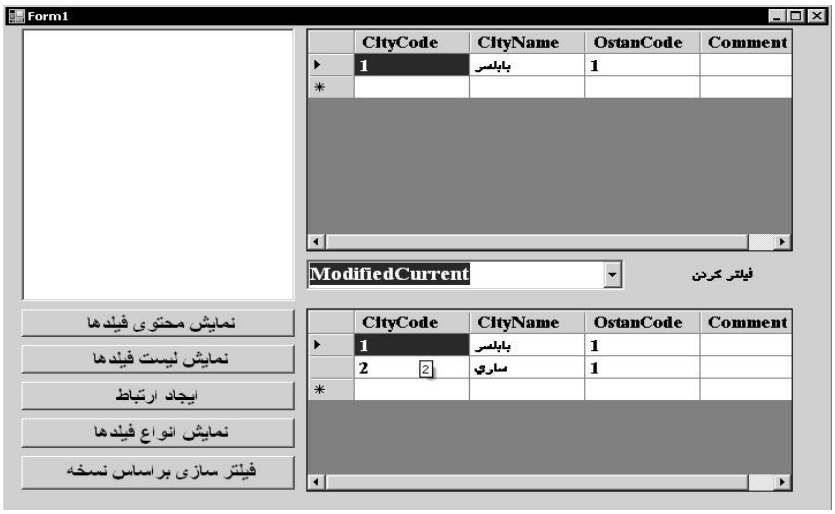
شکل۴-۴ نمایش محتوی فیلدهای جدول tblUser.

در این شکل گزینه ModifiedCurrent را انتخاب کنید تا شکل٦-٤ ظاهر شود. همان طور که در این شـکل می‌بینید، در dataGridView1 فقط شهر بابلسر نمایش داده شده است.

اکنون دکمه «**نمـایش انـواع فیلـدها**» را کلیـک کنیـد تـا فیلـدهای تصـویر و ComboBoxای را بـر روی dataGridView1 ببینید. فیلد سطح دسترسی را کلیک کنید تا شکل زیر ظاهر شود.



شکل۵-۴ نمایش اطلاعات جدول City



شکل۶-۴ نمایش اطلاعات تغییر یافته جدول City با نسخه تغییرات انجام شده.

فصل ۷ بخشی از اشیای پیشرفته SQL Server و ADO.NET

در فصل‌های قبل با بانک اطلاعات آشنا شدیم و توانستیم در بانک اطلاعات جدول ایجاد کرده، رکوردهایی به آنها از طریق ADO.NET اضافه نماییم و رکوردهای موجود را ویرایش، حذف یا بازیابی کنیم. در این فصل می‌خواهیم به اشیای دیگر بانک اطلاعات از قبیل دیدها[۳۱]، توابع[۳۲] و رویه‌های ذخیره شده[۳۳] بپردازیم و چگونگی 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، جدول‌های مجازی[۳۴] 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 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
```

---

[31]- Views [2]- Functions [3]- Stored Procpdures [4]- Virtual tables

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
```

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[۳۵] 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[۳۶] 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 ذخیره شده ترافیک شبکه را کاهش می‌دهند. زیرا، ممکن است رویه‌های ذخیره شده از چندین سطر تشکیل شوند. در حالت عادی، وقتی سرویس‌گیرنده[۳۷] 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 یک خط را بفرستید تا رویه ذخیره شده اجرا گردد. چون بدنه رویه ذخیره شده در سمت سرویس‌دهنده[۳۸] 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.

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 ]
```

---

[35]- Cache Memory
[2]- Modular Programming
[37]- Client
[2]- Server

```
[ { @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، دستور 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 شعبه وجود ندارد
   else
   begin
     DELETE [Shobe]   WHERE ( [ShobeCode] = @ShobeCode)
     set @strOut='0' -- شعبه حذف گردید
   end
```

این دستورات، رویه ذخیره‌شده delete_Shobe را ایجاد می‌کند 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].[insert_Shobe]
 (@ShobeCodeVarchar(6)
  @ShobeNameVarchar(30),
  @strOut varchar(1) output)
AS
 if @ShobeCode in (select ShobeCode from Shobe)
   set @strOut='1' -- شماره شعبه در شعبه وجود دارد
 else if @ShobeName not in (select ShobeName from Shobe)
   set @strOut='2' - نام شعبه در شعبه وجود دارد
 else
 begin
   INSERT INTO [Shobe](ShobeCode, ShobeName) VALUES (@ShobeCode, @ShobeName)
   set @strOut='0' - شعبه اضافه گردید
```

```
end
```

جدول ۱-۷ پارامترهای دستور CREATE PROCEDURE.

پارامتر	هدف
Procedure_name	نام رویه ذخیره شده‌ای را تعیین می‌کند که ایجاد می‌شود.
number	شماره‌ای را مشخص می‌کند که می‌توانید به رویه ذخیره شده تخصیص دهید تا برخی از رویه‌های ذخیره شده که در یک گروه قرار می‌گیرند با این شماره بتوانید فراخوانی کنید.
@Parameter	پارامترهایی را تعیین می‌کنند که می‌توانید مقادیر را از طریق آنها برای رویه ذخیره شده ارسال کنید.
data_type	نوع هر یک از پارامترهای رویه ذخیره شده را تعیین می‌کند.
VARYING	مجموعه نتایج را به عنوان پارامتر خروجی مشخص می‌کند.
default	مقدار پیش‌فرض پارامتر رویه ذخیره شده را تعیین می‌کند. اگر رویه ذخیره شده را بدون مقدار فراخوانی کنید، این مقدار برای پارامتر رویه ذخیره شده در نظر گرفته می‌شود.
OUTPUT	نوع پارامتر را خروجی در نظر می‌گیرد.
WITH RECOMPILE	با هر فراخوانی رویه ذخیره شده دوباره کامپایل می‌گردد.
WITH ENCRYPTION	دستورات رویه ذخیره شده را رمزگذاری می‌کند تا قابل بازیابی نباشد.
FOR REPLICATION	رویه ذخیره شده را برای تکثیر ایجاد می‌کند که توسط سرویس‌گیرنده قابل اجرا نیست.
SQL_Statement	دستوراتی هستند که رویه ذخیره شده را ایجاد می‌کنند.

این دستورات، رویه ذخیره شده insert_Shobe را اضافه می‌کنند. اگر کد شعبه وجود داشته باشد، Strout@ برابر '1' خواهد شد، ولی اگر نام شعبه موجود باشد، strout@ برابر '2' می‌گردد. وگرنه، یک شعبه جدید اضافه کرده، مقدار Strout@ را برابر '0' قرار می‌دهد.

۲-۵-۷. اجرای رویه ذخیره شده

برای اجرای رویه ذخیره شده در محیط SQL Server می‌توانید از دستور EXECUTE استفاده کنید. این دستور به صورت زیر به کار می‌رود.

```
[ { EXEC | EXECUTE } ]
 { [ @return_status = ]
 { module_name [ ;number ] | @module_name_var }
   [ [ @parameter = ] { value | @variable [ OUTPUT ]
      | [ DEFAULT ] }][ ,...n ][ WITH RECOMPILE ]}[;]
```

پارامترهای این دستور عبارت‌اند از:

- **متغیر return_status**: وضعیت را برمی‌گرداند.
- **متغیر @procedure_name**: نام متغیر پارامتر را تعیین می‌کند.
- **پارامتر WITH RECOMPILE**: تعیین می‌کند رویه ذخیره شده قبل از اجرا کامپایل گردد و سپس، اجرا شود. در حالت معمولی، رویه ذخیره شده قبل از اجرا کامپایل نمی‌شود.

به عنوان مثال، دستور زیر را ببینید:

```
EXECUTE delete_Shobe '100'
```

این دستور، با اجرای رویه ذخیره شده delete_Shobe، شعبه شماره '100' را حذف می‌کند.

فصل ۹ بخشی از گزارش‌گیری با Crystal Reports

تاکنون با روش‌های ایجاد بانک اطلاعاتی و اشیا آن، درج، ویرایش، حذف و بازیابی اطلاعات آشنا شدید. همان‌طور که می‌دانید یکی از بخش‌های بسیار مهم برنامه‌های کاربردی داشتن ابزاری مناسب برای تهیه گزارش می‌باشد. ابزارهای متعددی از قبیل Quick Report، Rave Report، Microsoft Report و غیره برای تهیه گزارش از بانک اطلاعات ارائه شده‌اند که رایج‌ترین و مهم‌ترین آنها، ابزار Crystal Report است. برخی از ویژگی‌های این ابزار گزارش‌گیری که آن را از بقیه ابزارهای گزارش‌گیری متمایز نموده است، عبارت‌اند از:

- قوی‌ترین نرم‌افزارهای موجود در بازار جهت گزارش‌گیری است.
- محبوب‌ترین نرم‌افزار طراحی گزارش در بین برنامه‌نویسان می‌باشد.
- قابلیت پاسخ‌گویی به کاربران مبتدی، متوسط و حرفه‌ای برای دریافت انواع گزارش سریع و دقیق را دارد.
- بیش از ۷ میلیون نفر در دنیا از این نرم‌افزار استفاده می‌نمایند.
- داشتن امکانات ویزاردی که تهیه گزارش را سریع و آسان می‌نماید.

۹-۱. امکانات نرم‌افزار Crystal Report

نرم‌افزار Crystal Report دارای امکانات مختلفی است. این امکانات، تهیه گزارش‌های سریع و دقیق را برای انواع کاربران فراهم کرده است. برخی از این امکانات عبارت‌اند از:

- امکان تهیه انواع گزارش‌های مختلف از قبیل لیستی، نموداری، آماری و غیره.
- قابلیت اتصال به انواع بانک‌های اطلاعاتی مختلف از قبیل SQL Server، Access، Oracle، DB2، پاراداکس و غیره.
- قابلیت درج انواع فیلدهای عددی، رشته‌ای، تصاویر، پیوندهای اینترنتی و فیلدهای دیگر در گزارش.
- قابلیت اتصال به منابع داده مختلف از قبیل DataSet، XML و DataReader.
- قابلیت گروه‌بندی گزارشات براساس انواع فیلدها.
- قابلیت طراحی فرمول‌های پیشرفته با استفاده از توابع ریاضی، آماری، رشته‌ای، تاریخ و زمان.
- داشتن محیط برنامه‌نویسی به دو حالت Crystal Syntax و Basic Syntax.
- قابلیت درج و افزودن توابع دلخواه از زبان برنامه‌نویسی دیگر به صورت کتابخانه‌های پیوند پویا (DLL)[۳۹].

[39]- Dynamic Link Library

- قابلیت ارتباط با انواع زبان‌های برنامه‌نویسی از قبیل ویژوال استودیو نسخه۶، ویـژوال اسـتودیو دات‌نـت، دلفی و دلفی‌نت.
- قابلیت طراحی گزارش به صورت پویا یا RunTime.
- قابلیت انجام محاسبات فرمول‌ها و سایر توابع در طرف سرویس‌دهنده (Server Side).

۹-۲. مراحل طراحی گزارش

طراحی هر گزارش مراحل مختلفی دارد که این مراحل در زیر آمده‌اند:

۱.ایجاد ارتباط با بانک اطلاعاتی.

۲.افزودن جداول، فیلدها، فرمول‌های محاسباتی، پارامترها و ... در گزارش.

۳.تعیین نحوه مرتب‌سازی و گروه‌بندی گزارش.

٤.تعیین شرط نمایش رکوردها در گزارش.

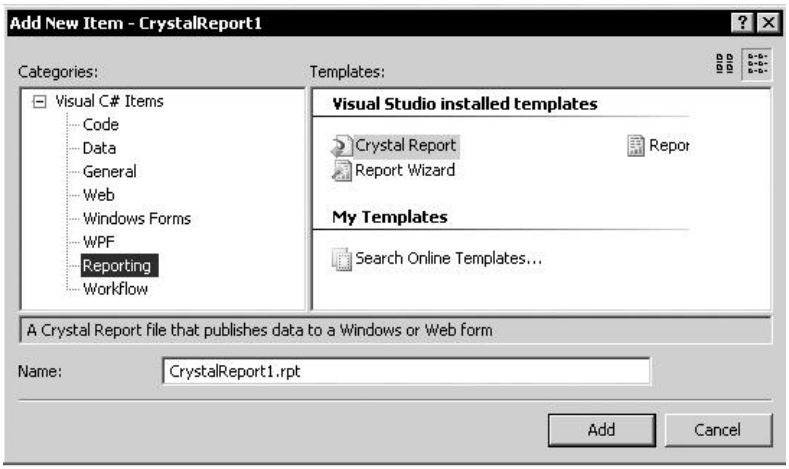
۹-۳. ایجاد گزارش با ویزارد

برای ایجاد گزارش با ویزارد مراحل زیررا انجام دهید:

۱.پروژه جدیدی ایجاد کنید.

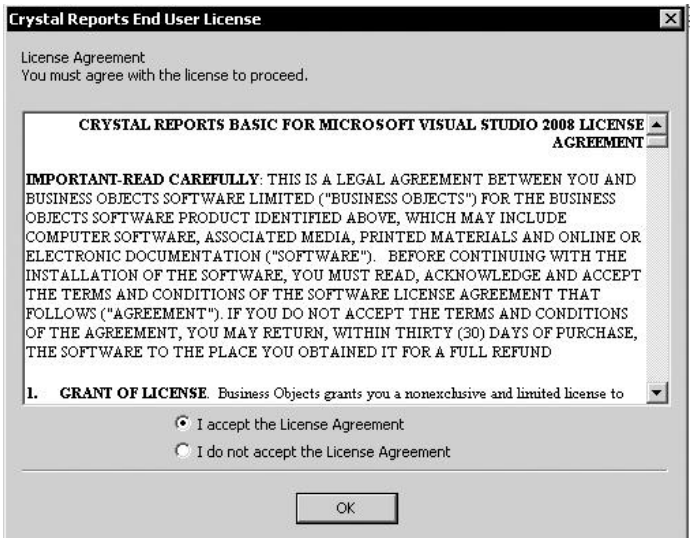
۲.گزینه Project/Add New Item را کلیک کنید تا پنجره Add New Item ظاهر شود.

۳.در بخش Categories، گزینه Reporting را کلیک کنید تا پنجـره Add New Item بـه شـکل ۱-۹ تغییـر یابد.



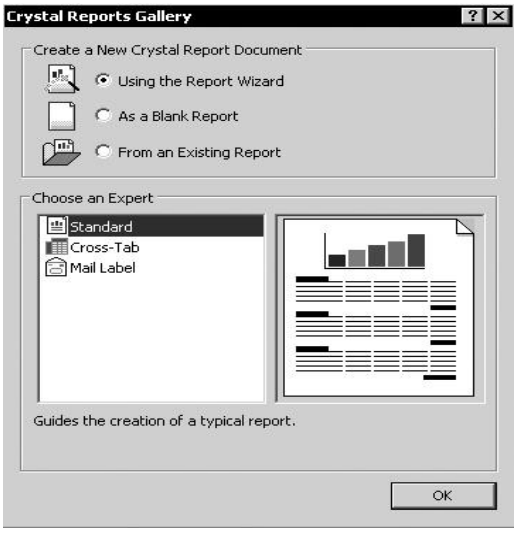
شکل ۱-۹ پنجره Add New Item.

٤.در بخش Templates گزینه Crystal Report را انتخـاب کـرده، دکمـه Add را کلیـک کنیـد تـا پنجـره License Agreement ظاهر شود (شکل ۲-۹).



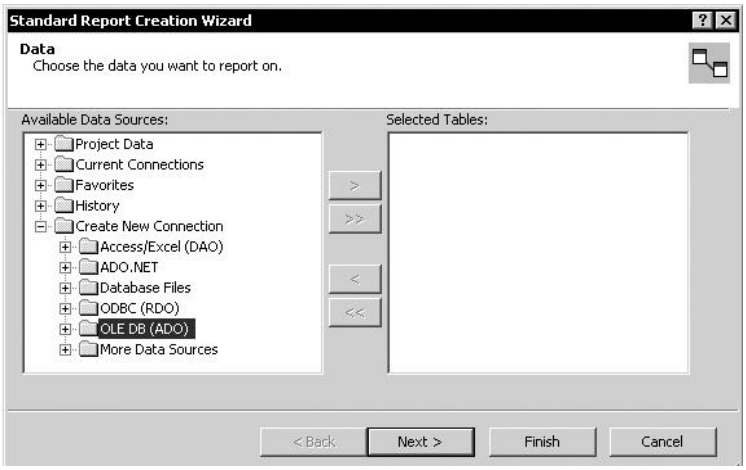
شکل۲-۹ پنجره License Agreement.

۵.در این پنجره، دکمه OK را کلیک کنید تا پنجره Crystal Reports Gallery ظاهر شود (شکل۳-۹).



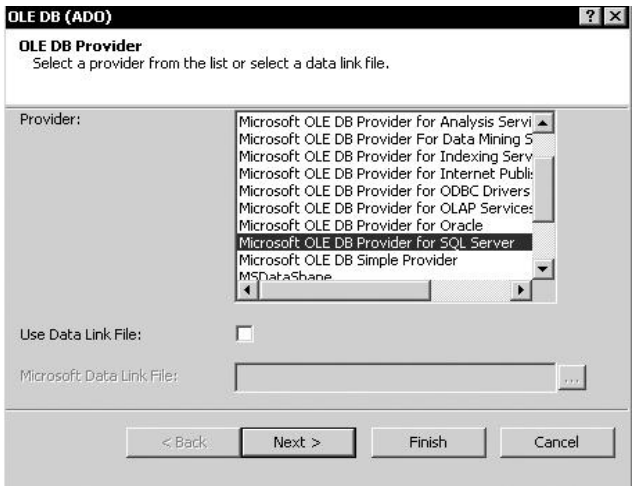
شکل۳-۹ پنجره Crystal Reports Gallery.

٦.در پنجره Crystal Reports Gallery، گزینه Using the Report Wizard را انتخاب کرده، دکمه OK را کلیک کنید تا پنجره Data ظاهر شود (شکل٤-۹). در این پنجره، آیکن Create New Connection را کلیک کنید تا آیکن‌های آن ظاهر شود.



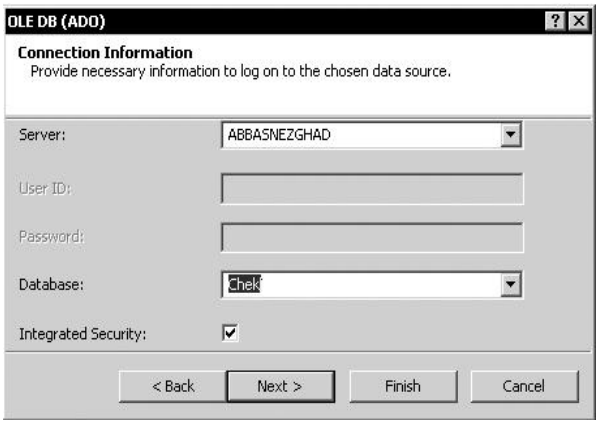
شکل۴-۹ پنجره Data.

۷.در پنجره Data، آیکن OLE DB(ADO) را کلیک کنید تا پنجره OLE DB_Provider ظاهر شود (شکل۵-۹).



شکل۵-۹ پنجره OLE DB Provider.

۸.در این پنجره، گزینه Microsoft OLE DB Provider for SQL Server را انتخاب کرده دکمه Next را کلیک کنید تا پنجره Connection Information ظاهر شود (شکل۶-۹).

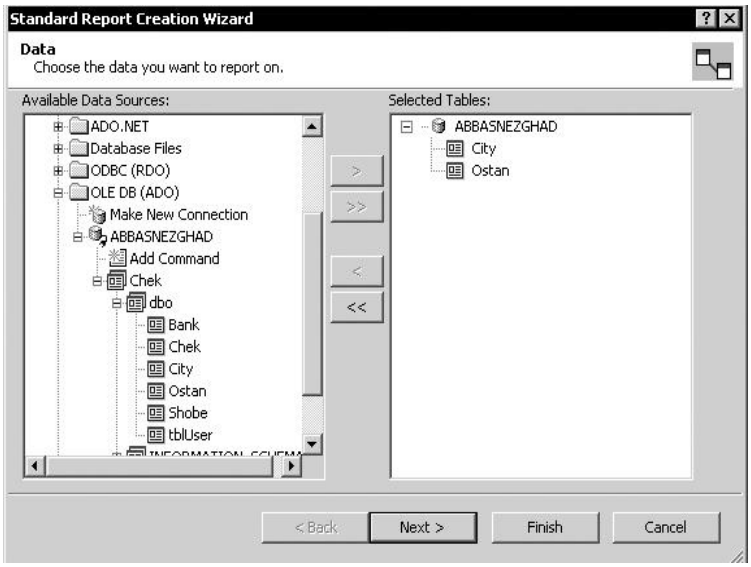


شکل۶-۹ پنجره Connection Information.

۹.در این پنجره اطلاعات را مانند شکل۶-۹ وارد کرده، دکمه Finish را کلیک کنید تا به پنجره Data برگردید.

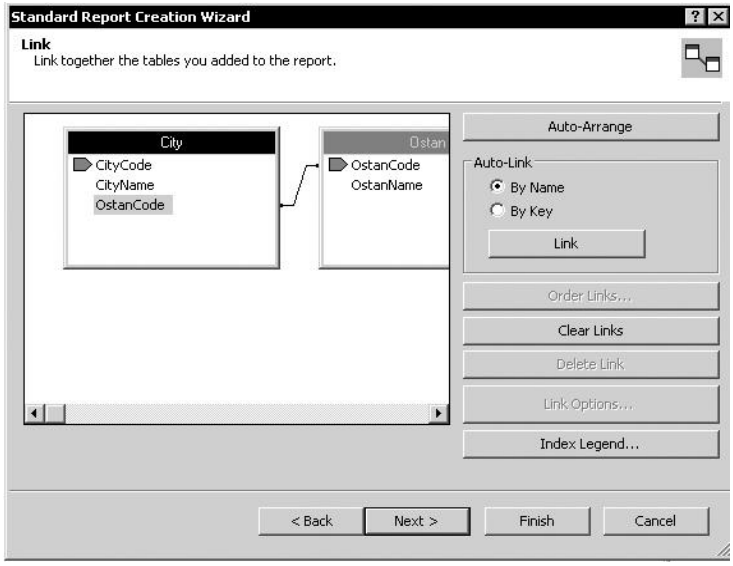
۱۰.در پنجره Data، آیکن‌های ABBASNEZGHAD، Chek و dbo را باز کنید تا لیست جداول آن را ببینید.

۱۱.جداول City و Ostan را کلیک مضاعف کنید تا شکل۷-۹ ظاهر شود.



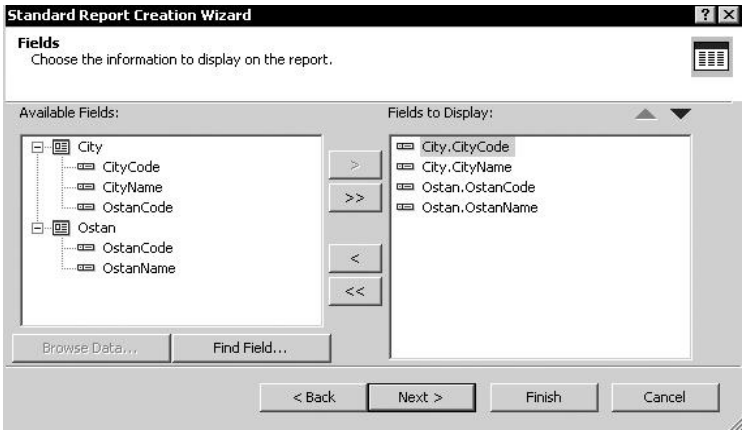
شکل۷-۹ انتخاب جداول در پنجره Data.

۱۲.در این پنجره دکمه Next را کلیک کنید تا پنجره Link ظاهر شود (شکل۸-۹). این پنجره برای ایجاد ارتباط بین جداول بانک اطلاعات به کار می‌رود.



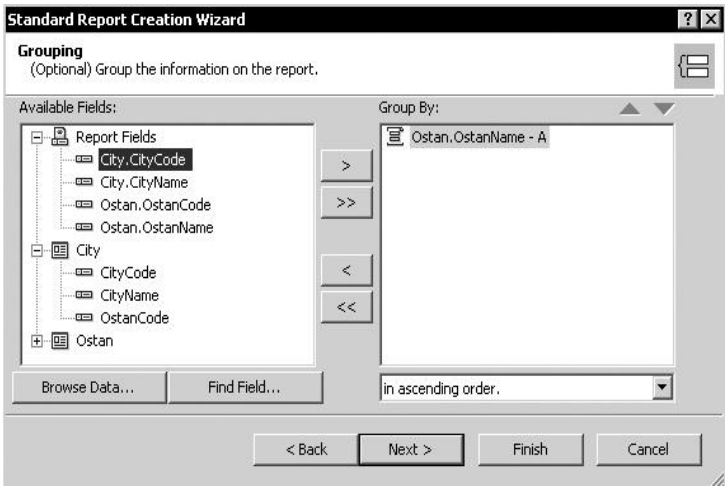
شکل۸-۹ پنجره Link.

۱۳.در پنجره Link، ارتباط بین جداول را ایجاد کرده دکمه Next را کلیک کنید تا پنجره Fields ظاهر شود (شکل۹-۹).



شکل۹-۹ پنجره انتخاب فیلدها (Fields).

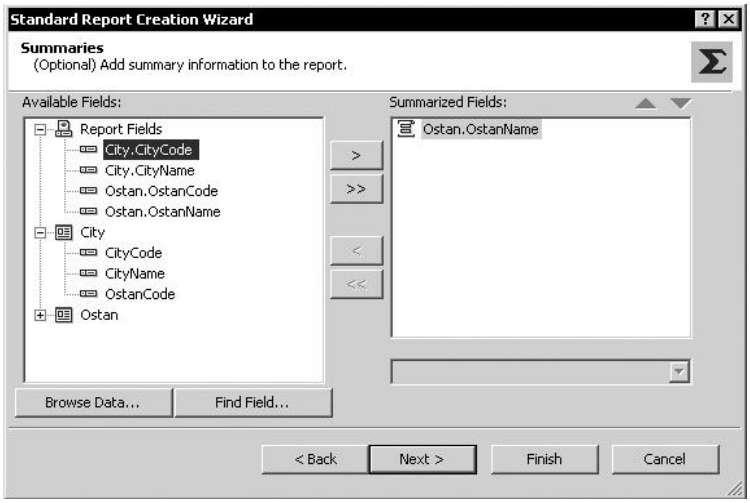
۱٤.فیلدهای CityCode، CityName از جدول City و OstanCode و OstanName از جدول Ostan را در بخش Available Fields کلیک مضاعف کرده تا این فیلدها در بخش Fields to Display کپی شوند. اکنون دکمه Next را کلیک کنید تا پنجره Grouping ظاهر شود (شکل۱۰-۹).



شکل ۱۰-۹ پنجره Grouping.

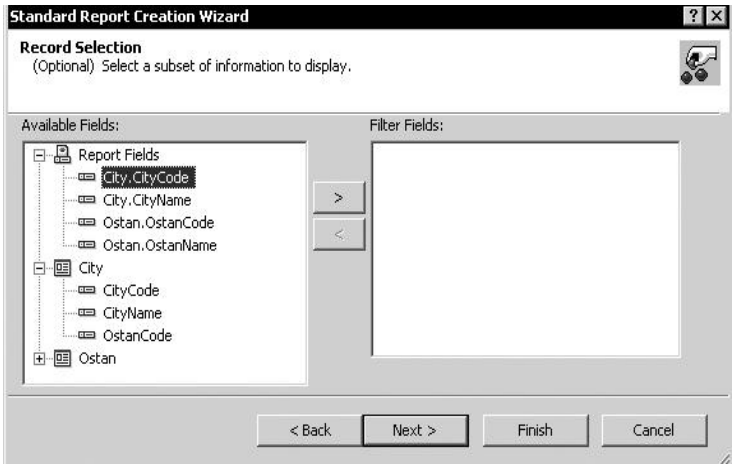
۱۵.در این پنجره، فیلد OstanName را کلیک مضاعف کرده تا پنجره Grouping (مانند شکل ۱۰-۹) ظاهر شود. اکنون گروه‌بندی اطلاعات براساس نام استان انجام می‌شود.

۱۶.در این پنجره، دکمه Next را کلیک کنید تا پنجره Summaries ظاهر شود (شکل ۱۱-۹). دکمه Next

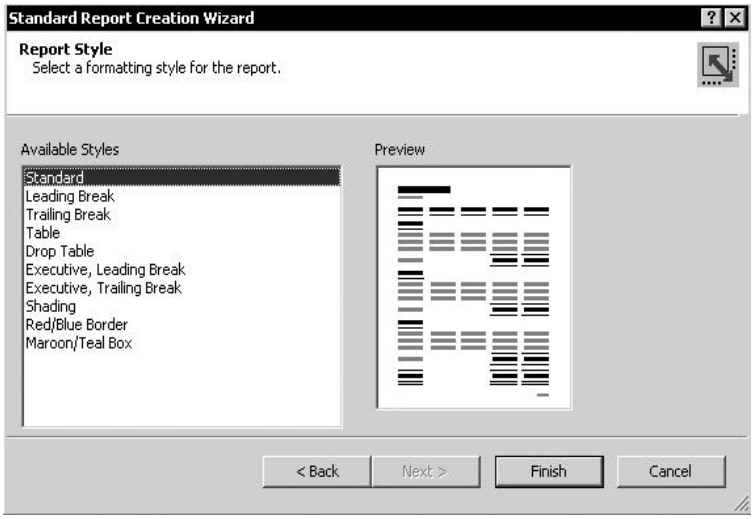


شکل ۱۱-۹ پنجره Summaries.

را کلیک کنید تا پنجره Record Selection ظاهر شود (شکل ۱۲-۹). در این پنجره دکمه Next را کلیک کنید (چون در این بخش نمی‌خواهیم رکوردهای خاصی را نمایش دهیم) تا پنجره Report Style ظاهر شود (۱۳-۹).



شکل۱۲-۹ پنجره Record Selection.



شکل۱۳-۹ پنجره Record Style.

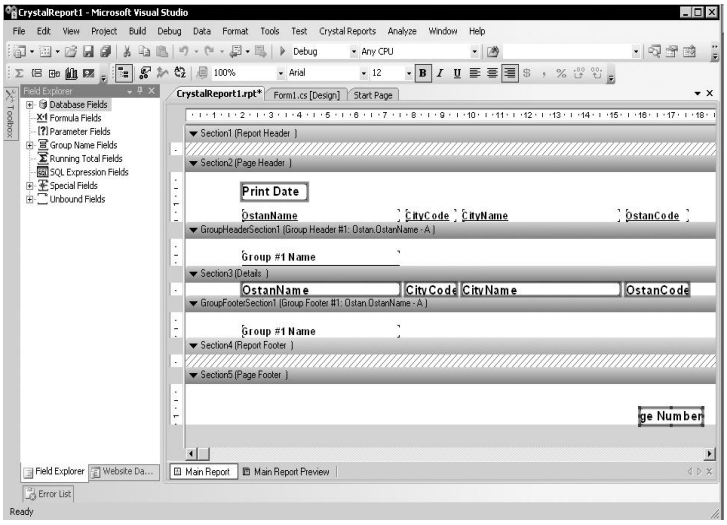
۱۷.در این پنجره، سبک گزارش را انتخاب کرده و دکمه Finish را کلیک نمایید. اکنون پنجره Preview گزارش ظاهر می‌شود (شکل۱٤-۹).

۹-۴. بخش‌های گزارش

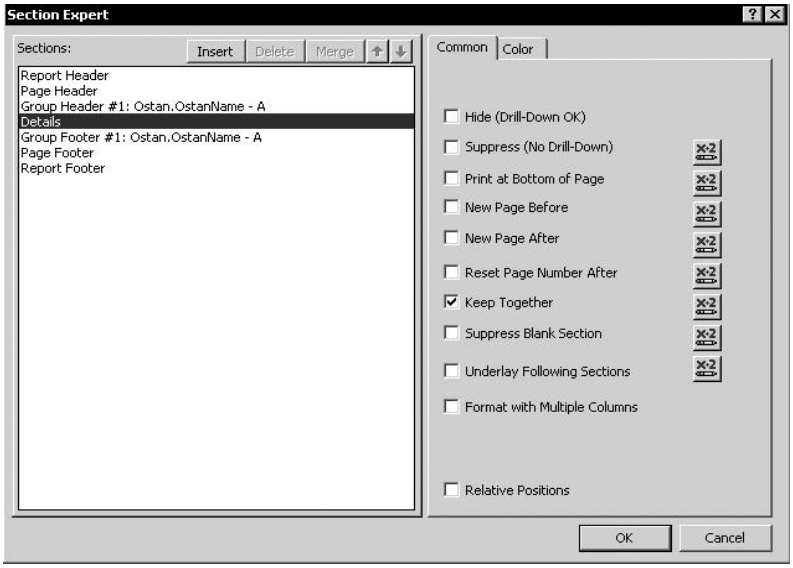
هر گزارش دارای بخش‌های مختلفی است که در جدول ۱-۹ آمده‌اند (شکل۱٤-۹). برای کار کردن با بخش‌های گزارش به نکات زیر توجه کنید:

- برای حذف بخشی از گزارش، بر روی آن کلیک راست کرده از منویی که ظاهر می‌شود، گزینه Delete Section را اجرا کنید (البته بخش‌های اصلی گزارش را نمی‌توانید حذف نمایید).

- برای اضافه کردن بخش جدید به گزارش، بر روی بخش کلیک راست کرده از منویی که ظاهر می‌شود، گزینه Insert Section Below را اجرا کنید.
- برای تعیین خواص بخشی از گزارش بر روی آن کلیک راست کرده از منویی که ظاهر می‌شود، گزینه Section Expert را اجرا کنید تا کادر محاوره Section Expert را ببینید (شکل۱۵-۹). اطلاعات این شکل خواص بخش را تعیین می‌کنند که در جدول۲-۹ آمده‌اند. خواص این بخش‌ها را تعیین کرده دکمه OK را کلیک کنید.



شکل۱۴-۹ پنجره Preview گزارش.



شکل۱۵-۹ پنجره خواص Section Expert.

جدول ۱-۹ بخش‌های گزارش.	
بخش	**هدف**
Report Header	عنوان گزارش را تعیین می‌کند. اطلاعات این بخش فقط یک بار در اولین صفحه گزارش نمایش داده می‌شوند.
Page Header	اطلاعات سرصفحه گزارش را تعیین می‌کند. اطلاعات این بخش در ابتدای هر صفحه گزارش چاپ می‌گردند.
Details	اطلاعاتی که در این بخش قرار می‌گیرند، به ازای هر رکورد بانک اطلاعاتی در گزارش چاپ خواهند شد. در این بخش معمولاً نام فیلدهای جداول بانک اطلاعات قرار می‌گیرند.
Page Footer	اطلاعات این بخش در پایان هر صفحه گزارش فقط یک بار نمایش داده می‌شود (مانند جمع فیلدهای عددی برای هر صفحه)
Report Footer	اطلاعات این بخش در آخرین صفحه گزارش یک بار نمایش داده می‌شوند (فقط در آخرین صفحه). مانند خلاصه‌ای از فیلدهای عددی یا امضاء مسئولین.

فصل ۱۰ بخشی از پشتیبان‌گیری و بازیابی پشتیبان از بانک اطلاعات

یکی از مهم بخش‌های 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).

۴. 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[40] 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 ] ]
 [ 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 } ]
```

[40]- Log

```
[ [ , ] { 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.

جدول ۱-۱۰ پارامترهای دستور BACKUP DATABASE.

پارامتر	هدف
database_name	نام بانک اطلاعاتی است که باید از آن پشتیبان‌گیری شود.
@database_name_var	نام بانک اطلاعاتی که باید از آن پشتیبان‌گیری شود، در این متغیر قرار دارد.
backup_device	نام دستگاهی را تعیین می‌کند که پشتیبان باید در آن قرار داده شود.
DESCREPTION	توصیف پشتیبان را مشخص می‌نماید.
BLOKSIZE	اندازه فیزیکی بلوک را به بایت تعیین می‌کند.
DIFFERENTIAL	از اطلاعاتی که تغییر یافته‌اند، پشتیبان می‌گیرد.
EXPIREDATE	تاریخ و زمان انقضای پشتیبان را تعیین می‌کند.
RETAINDAYS	تعیین می‌نماید چند روز بعد می‌توان بر روی این مجموعه پشتیبان بازنویسی (رونویسی) کرد.
PASSWORD	کلمه عبور بر روی پشتیبان قرار می‌دهد تا هر کاربری نتواند آن را بازیابی کند.
FORMAT	تعیین می‌کند سرآیند[41] رسانه باید بر روی تمام Volumeهایی که برای عمل پشتیبان استفاده شده‌اند، نوشته شود.
NOFORMAT	تعیین می‌کند سرآیند رسانه بر روی تمام Volumeهایی که برای عمل پشتیبان‌گیری استفاده شده‌اند، نوشته نشود.
INIT	تمام مجموعه پشتیبان به جز سرآیند رسانه باید بازنویسی شود.
NOINIT	مجموعه پشتیبان را به یک دستگاه دیسک یا نوار اضافه می‌کند و مجموعه پشتیبان‌های قبلی را حفظ می‌کند.
MEDIADESCRIPTION	توصیف رسانه را مشخص می‌کند.

[41]- Header

ادامه جدول ۱-۱۰ پارامترهای دستور BACKUP DATABASE.	
پارامتر	هدف
MEDIANAME	نام رسانه را تعیین می‌کند که حداکثر ۱۲۸ کاراکتر است.
MEDIAPASSWORD	کلمه عبور را بر روی رسانه قرار می‌دهد تا هر کاربری نتواند آن رسانه را باز کند.
NAME	نام پشتیبان را مشخص می‌کند که حداکثر ۱۲۸ کاراکتر می‌باشد.
NOSKIP	سرآیند رسانه را می‌خواند و تست‌های از قبیل کلمه عبور را انجام می‌دهد و اگر پارامتر MEDIANAME تعیین گردد، چک می‌کند آیا نام رسانه دریافتی با نام رسانه در سرآیند برابر است یا خیر.
SKIP	تاریخ انقضای و تست اولیه مجموعه پشتیبان را غیرفعال می‌کند.
NOUNLOAD	پس از پشتیبان‌گیری، نوار را از نوارخوان بیرون نمی‌آورد.
UNLOAD	پس از پشتیبان‌گیری، نوار را از نوارخوان بیرون می‌آورد.
NOREWIND	پس از پشتیبان‌گیری، نوار را به عقب برنمی‌گرداند.
REWIND	پس از پشتیبان‌گیری، نوار را به عقب برمی‌گرداند.
RESTART	اگر پشتیبان‌گیری قطع شده باشد، پشتیبان‌گیری را از مکانی که قطع شده است، ادامه می‌دهد.
STATS	فاصله زمانی را تعیین می‌کند که باید درصد پیشرفت کار نمایش داده شود.

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.

مثال۱-۱۰

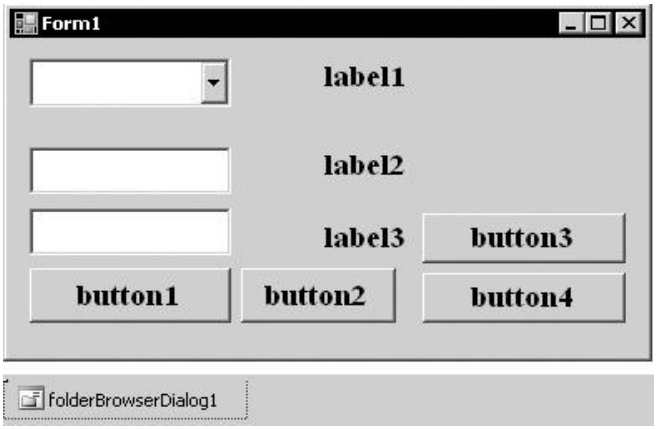
برنامه‌ای که با استفاده از دستور Backup و Restore از بانک اطلاعات پشتیبان تهیه کرده یا پشتیبان را بازیابی می‌کند.

مراحل طراحی و اجرا

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

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

```
using System.Data.SqlClient;
using 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.**

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

```
private void Form1_Load(object sender, EventArgs e)
{
  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 برای بازیابی اطلاعات";
  label3.Text = "مسیر";
  comboBox1Fill();
}
```

این دستورات، خواص کنترل‌های روی فرم را مقداردهی می‌کنند و تابع comboBox1Fill() را فراخوانی 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:**

```
private string SQL = "SELECT name FROM sys.databases";
private  string connStr = "Data Source=.\\SQLEXPRESS;Initial
      Catalog=;Integrated Security=True";
// Declare global objects...
SqlConnection conn;
SqlDataAdapter da;
DataSet ds = new DataSet();
```

این دستورات، ابتدا دستور SQLی را تعریف می‌کنند که برای بازیابی بانک‌های اطلاعات به کار می‌رود. در ادامه، رشته اتصال را تعریف کرده، مقداردهی می‌کنند و در پایان، اشیایی از قبیل SqlConnection، SqlDataAdapter و DataSet را تعریف می‌نمایند.

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

```
private void 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";
}
```

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 متصل به سرویس دهنده را به da انتقال می‌دهد (با دستور SELECT name FROM sys.databases) و در پایان، نام بانک‌های 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 زیر تغییر دهید:

```
private void button1_Click(object sender, EventArgs e)
{
  try
  {
   if (textBox2.Text == "") MessageBox.Show("مسیر انتخاب نشد");
    else
    {
      if (conn.State == ConnectionState.Open) conn.Close();
       conn.ConnectionString = connStr;
       conn.Open();
       string cmdBackup = "BACKUP DATABASE " + comboBox1.Text
          + " TO DISK = '" + textBox2.Text + comboBox1.Text +
          "1.bak', " + "DISK='" + textBox2.Text +
          comboBox1.Text + "2.bak' ";
       SqlCommand cmd = new SqlCommand(cmdBackup, conn);
       cmd.CommandType = CommandType.Text;
       cmd.ExecuteNonQuery();
       conn.Close();
       MessageBox.Show("از بانک اطلاعات مورد نظر پشتیبان گیری گرفته شد");
      }
    }
    catch
```

```
    {
      MessageBox.Show("خطا در پشتیبان گیری رخ داد");
    }
  }
```

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)، در بخش else ابتدا 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 void button2_Click(object sender, EventArgs e)
{
  try
  {
    if (textBox1.Text == "" || textBox2.Text == "")
      MessageBox.Show("نام بانک یا مسیر انتخاب نشد");
    else
    {
      connStr = "Integrated Security=SSPI;
        Data Source=.\\SQLEXPRESS;";
        conn = new SqlConnection(connStr);
      if (conn.State == ConnectionState.Open)     conn.Close();
      conn.ConnectionString = connStr;
      conn.Open();
      string cmdRestore = "RESTORE DATABASE " + textBox1.Text +
        " FROM DISK = '" + textBox2.Text + textBox1.Text +
        "1.bak', " + "DISK =  '" + textBox2.Text  +
        textBox1.Text + "2.bak' ";
      SqlCommand cmd = new SqlCommand(cmdRestore, conn);
      cmd.CommandType = CommandType.Text;
      cmd.ExecuteNonQuery();
      MessageBox.Show("بانک اطلاعات مورد نظر بازیابی شد");
      conn.Close();
    }
  }
  catch
  {
    MessageBox.Show("خطا در بازیابی رخ داد");
  }
}
```

این دستورات، بانک اطلاعاتی که نـام آن در textBox1 وارد شـده اسـت و پشـتیبان آن در مسـیری کـه در textBox2 وارد گردید، را بازیابی می‌کنند. برای انجام این کار، از try استفاده می‌کند تا در صورت بروز خطا در بازیابی پشتیبان، برنامه 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 یـک شـیء اتصـال ایجـاد کرده اگر اتصال باز باشد، آن را می‌بندد. در ادامه، رشته اتصـال را مقـدار داده و اتصـال را بـا متـد ()Open بـاز می‌کند. در پایان، دستور SQLایی که برای بازیابی به کار می‌رود را ایجاد کرده با ایجـاد شـیء 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 آن را به صورت زیر تغییر دهید:

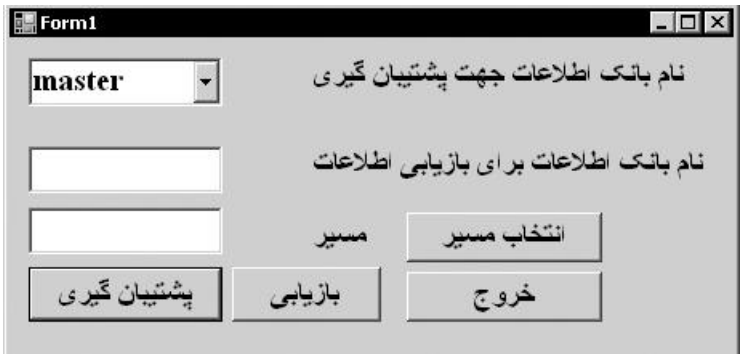
```
private void button3_Click(object sender, EventArgs e)
{
  folderBrowserDialog1.Description = "مسیر مورد نظر را انتخاب کنید";
  folderBrowserDialog1.ShowDialog();
  textBox2.Text = folderBrowserDialog1.SelectedPath;
  if (textBox2.Text.Length >  3) textBox2.Text = textBox2.Text + "\\";
}
```

این دستورات، با استفاده از کنترل FolderBrowserDialog مسیر پشتیبان‌گیری و بازیابی پشتیبان را انتخـاب می‌کنند. اگر اطلاعات مسیر انتخاب شده بیش از سه کاراکتر باشد، به انتهای مسیر کارکتر '\' اضافه می‌گردد.

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

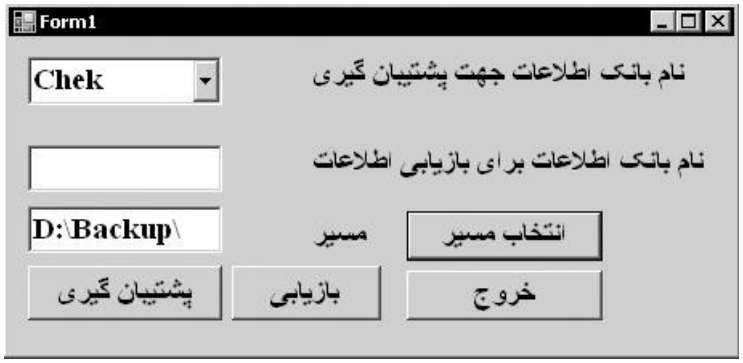
```
private void button4_Click(object sender, EventArgs e)
{
   Close();
}
```

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



شکل۱-۱۰ نمونه اولیه خروجی مثال۱-۱۰.

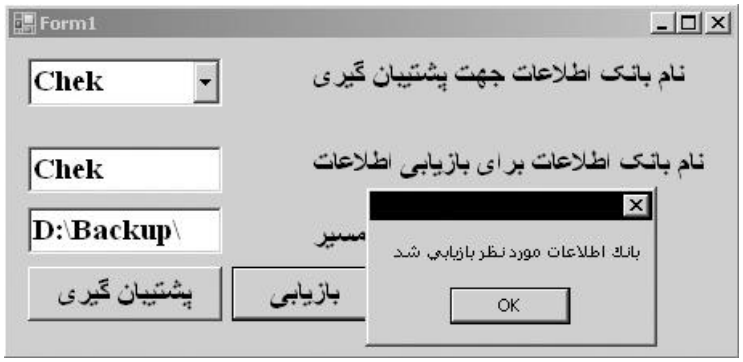
در comboBox1 نام بانک اطلاعات را انتخاب کنید (من بانک اطلاعات Chek را انتخاب نمـودم) و دکمـه انتخاب مسیر را کلیک کنید تا شکل انتخاب مسیر ظاهر شود. در این شکل مسیر مـورد نظـر (بـه عنـوان مثـال، D:\Backup را انتخاب کنید و دکمه OK را کلیک کرده تا مسیر انتخاب شده را در textBox2 ببینیـد (شـکل۲-۱۰)).



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.