دانشگاه آزاد اسلامي واحد ماهشهر
گروه کامپیوتر

پایگاه داده در C#

استاد:

مهندس شکاری

جمع آوری کننده:

شهرام برنا

در این مطلب، که آغاز سری مطالب مرتبط با پایگاه های داده و نحوه برقراری ارتباط با آنها است، در ابتدا به معرفی مختصری از پایگاه داده پرداخته می شود و سپس مقدمات کار با پایگاه داده آموزش داده شده است. از آنجائیکه برقرای ارتباط با پایگاه داده نیازمند داشتن دانشی از زبان SQL است، از اینرو در این سری مطالب، ابتدا به بیان دستورات بنیادی SQL پرداخته شده و سپس نحوه استفاده از آنها بیان شده است. برای سادگی کار و جلوگیری از پیچیده شدن مطالب در ابتدا کار، در مطالب نخستین از پایگاه داده MS Access استفاده شده است که در بسته نرم افزاری MS Office قرار دارد. پس از بررسی مقدمات و آموختن نحوه برقراری ارتباط با یک پایگاه داده، به سراغ مطالب پیشرفته خواهیم رفت و با بسط مطالب و عمیق شدن در زبان SQL، نحوه کار با MS SQL Server 2000 را نیز فرا خواهیم گرفت. در دروس ابتدایی بیشتر سعی شده تا برای یادگیری بهتر و سادگی مطالب، قسمتی از کد نویسی های مربوط به برقراری ارتباط با پایگاه داده را کم کرده و این کار را به Visual Studio IDE محول سازیم اما در ادامه کار، با تمامی قسمتها و کد نویسی کامل آشنا خواهیم شد.

برای شروع کار، به MS Access (ترجیحا نسخه ۲۰۰۳) و Visual Studio .Net (بهتر است از نسخه ۲۰۰۳ استفاده کنید اما تمامی کدها با نسخه ۲۰۰۲ نیز قابل اجرا هستند اما توجه نمایید که در صورت استفاده از Visual Studio .Net 2005 و یا Visual C# Express Edition برای اجرای برخی کدها دچار مشکل خواهید شد.)

مقدمه

پایگاه داده یا همان *Database*، مجموعه ای سازمان داده شده از اطلاعات است که به شکل *جداول* ذخیره می شوند. برای سازمان دهی این اطلاعات روشهای متفاوتی وجود دارد که هدف تمامی آنها فراهم کردن روشهایی مناسب برای سهولت در برقراری ارتباط با پایگاههای داده و استفاده از اطلاعات موجود در آنها است. *سیستم مدیریت پایگاه داده (DBMS)*، مکانیزمی را جهت ذخیره سازی و بازیابی اطلاعات در پایگاه داده فراهم می نماید. در حقیقت DBMS باعث می شود تا برنامه نویس بدون نگرانی درباره چگونگی ذخیره سازی داداه ها در پایگاه داده و ساختار آنها، به اطلاعات دسترسی پیدا کند و بتواند داده های جدید را در آن ذخیره نماید.

امروزه اکثر پایگاه داده های محبوب از نوع *رابطه ای* (*Relational*) هستند. همانند پایگاه داده های عادی، در پایگاه داده های رابطه ای نیز دسترسی به اطلاعات ذخیره شده در جداول از طریق زبان پرس و جوی ساخت یافته یا همان *SQL* میسر میگردد که زبانی استاندارد است و توسط اکثر نرم افزارهای مرتبط با پایگاه داده مورد استفاده قرار می گیرد. از جمله سیستمهای پایگاه داده رابطه ای می توان به MS SQL Server، Oracle™، Sybase™، DB2™، Informix™ و MySQL™ اشاره کرد.

زبانهای برنامه نویسی از طریق یک *Interface* (یا همان نرم افزاری که ارتباط بین DBMS و یک برنامه را فراهم می کند) به پایگاه داده متصل شده و با آنها به تعامل میپردازند. در C# برقراری ارتباط با پایگاه داده از طریق *ADO.Net* انجام میشود. ADO.Net در حقیقت رابط بین نرم افزار و پایگاه داده است و امکانات ویژه ای را جهت دسترسی به اطلاعات موجود در آن در اختیار برنامه نویس قرار می دهد.

مدل پایگاه داده رابطه ای (Relational Database Model)

مدل پایگاه داده رابطه ای، نمایش منطقی ای از داده هاست که رابطه موجود بین داده ها را، بدون درگیر شدن با ساختار فیزیکی آنها، نشان میدهد. پایگاه داده رابطه ای از جداول تشکیل میشود. هر جدول خود از *سطرها* یا *رکوردها* (record/row) و *ستون*ها یا *فیلد*ها (columns/fields) تشکیل میشود. در شکل ۱، جدول نمونه ای به نمایش گذاشته شده است که نام آن Employee است و هدف آن نمایش دادن اطلاعات پرسنلی کارمندان یک اداره است. این جدول از ۶ رکورد تشکیل شده و فیلد number متعلق به هر رکورد در آن *Primary Key* است که جهت ارجاع به داده ها در جدول مورد استفاده قرار می گیرد. *Primary Key* فیلد یا فیلدهایی در جدول است (هستند) که شامل داده هایی یکتا باشند، بدین معنا که هیچ رکورد دیگری مقداری مشابه با آن ندارد. با استفاده از مفهوم Primary Key، این تضمین وجود دارد که هر رکورد حداقل با یک مقدار یکتا قابل شناسایی است. یک مثال خوب برای فیلد Primary Key، فیلدی است که حاوی کد ملی افراد باشد، چراکه در یک جامعه به ازای هر شخص یک کد ملی یا یک شماره منحصر بفرد وجود دارد و هیچ دو شماره ملی مشابهی یافت نمی شود. در جدول نمونه

ما، فیلد Primary Key حاوی شماره پرسنلی کارمندان است. همانطور که مشاهده می شود اطلاعات جدول بر اساس فیلد number *مرتب* شده است. در اینجا، اطلاعات بصورت صعودی مرتب شده اند. (حالت دیگر مرتب شدن حالت نزولی است.)

هر ستون از جدول، فیلدی متفاوت را نشان میدهد. معمولا رکوردها در حدول منحصر بفرد هستند (بوسیله Primary Key) اما مقادیر فیلدهای مختلف می تواند مشابه با یکدیگر باشد. برای مثال، ۳ رکورد مختلف در فیلد Department از جدول Employee حاوی مقدار ۴۱۳ هستند.

number	name	department	salary	location
۲۳۶۰۳	میثم	۴۱۳	۱۰۰۰	تهران
۲۴۵۶۸	علی	۴۱۳	۱۵۰۰	تهران
۳۴۵۸۹	محمد	۶۴۲	۱۳۰۰	مشهد
۳۵۷۶۱	ناصر	۶۱۱	۱۸۰۰	اصفهان
۴۷۱۳۲	مریم	۴۱۳	۱۵۰۰	تهران
۷۸۳۲۱	فاطمه	۶۱۱	۲۵۰۰	اهواز

با توجه به اینکه حجم اطلاعات قابل ذخیره در یک جدول نامحدود است، از اینرو باید بتوان با استفاده از روشی تنها به آن قسمت از اطلاعات دسترسی پیدا کرد که مورد نظر کاربر است. برای این منظور از SQL استفاده می نماییم. SQL مجموعه دستوراتی را فراهم می نماید که با استفاده از آنها قادر خواهیک بود تا اطلاعات مورد نظر را از پایگاه داده انتخاب (SELECT) کرده و مورد استفاده قرار دهیم.

نگاهی بر پایگاه داده رابطه ای

در ادامه مطالب این قسمت، با استفاده از یک پایگاه داده نمونه بنام Book با دستورات اولیه SQL آشنا خواهیم شد. پایگاه داده مورد نظر ما از چهار جدول تشکیل شده است. این جداول به ترتیب حاوی اطلاعاتی درباره نویسنده کتاب (Authors)، ناشر کتاب (Publishers)، کد شناسایی نویسنده (AuthorISBN) و عنوان کتاب (Titles) هستند. جدول Authors از سه فیلد تشکیل شده است که عبارتند از شماره اختصاصی هر نویسنده، نام و نام خانوادگی. در جدول زیر مشخصات جدول Authors نشان داده شده است.

نام فیلد	*توضیحات*
authorID	شماره شناسایی نویسنده را در پایگاه داده مشخص مینماید. در پایگاه داده Book این فیلد از نوع int تعریف شده و بصورت فیلدی اعلان شده که بطور خودکار مقدارش یک واحد یک واحد اضافه میشود (Auto-Increment Field). با اضافه شدن هر رکورد به پایگاه داده مقدار این فیلد یک واحد افزوده میشود که همین امر تضمین میکند که مقدار این فیلد همواره منحصر بفرد خواهد بود.
firstName	نام نویسنده کتاب. (از نوع string)
lastName	نام خانوادگی نویسنده. (از نوع string)

Authors		
lastName	firstName	authorID
Deitel	Harvey	1
Deitel	Paul	2
Nieto	Tem	3
Steinbuhler	Kate	4
Santry	Sean	5

Authors		
lastName	firstName	authorID
Lin	Ted	6
Sadhu	Praveen	7
McPhie	David	8
Yaeger	Cheryl	9
Zlatkina	Marina	10
Wiedermann	Ben	11
Liperi	Jonathan	12
Listfield	Jeffrey	13

جدول **Publisher** از دو فیلد تشکیل گردیده است که نمایش دهنده شماره شناسایی ناشر و همچنین نام ناشر است. شکل زیر مشخصات جدول **Publisher** را نشان میدهد.

نام فیلد	*توضیحات*
publisherID	شماره شناسایی ناشر را در پایگاه داده نشان میدهد. این فیلد که از نوع **int** است و بصورت خودکار مقدارش افزوده میشود، فیلد **Primary Key** جدول **Publisher** نیز میباشد.
publisherName	نام ناشر کتاب. (از نوع **string**)

Publishers	
publisherName	publisherID
Prentice Hall	1
Prentice Hall PTG	2

جدول **AuthorISBN** شامل دو فیلد است که نشان دهنده شماره شناسایی نویسنده و شماره **ISBN** کتابهایی است که یک نویسنده خاص آنها را نوشته است. با استفاده از این جدول میتوان کتابهایی را که یک نویسنده خاص آنها را نوشته بدست آورد. شکل زیر مشخصات جدول **AuthorISBN** را نشان میدهد.

نام فیلد	*توضیحات*
authorID	شماره شناسایی نویسنده کتاب را نشان میدهد. با اشتفاده از این فیلد میتوان کتابهای مرتبط با هر نویسنده را در پایگاه داده پیدا کرد. این فیلد که از نوع **int** میباشد، در جدول **Author** نیز قرار دارد.
isbn	شماره **ISBN** مربوط به هر کتاب. (**string**)

AuthorISBN	
authorID	ISBN
1	0130125075
2	0130125075
1	0130132497

AuthorISBN	
authorID	ISBN
2	0130132497
1	0130161438
2	0130161438
3	0130161438
1	0130284173
2	0130284173
3	0130284173
6	0130284173
7	0130284173
1	0130284181
2	0130284181
3	0130284181
8	0130284181
1	013028419X
2	013028419X
3	013028419X
1	0130293636
2	0130293636

جدول **Titles** از شش فیلد تشکیل شده است که عبارتند از **ISBN** مربوط به هر کتاب، عنوان کتاب، ویرایش، سال انتشار و شماره شناسایی ناشر، نام عکس جلد هر کتاب و قیمت هر کتاب. در شکل زیر مشخصات این جدول نشان داده شده است.

نام فیلد	*توضیحات*
isbn	شماره **ISBN** مربوط به هر کتاب. (**string**)
Title	عنوان کتاب. (**string**)
editionNumber	شماره ویرایش کتاب (**string**)
copyright	سال نشر کتاب (**int**)
publisherID	شماره شناسایی ناشر (از نوع **int**). مقادیر این فیلد میبایست با شماره ای در جدول **Publisher** همخوانی داشته باشد.
imageFile	نام فایلی که عکس روی جلد کتاب در آن قرار دارد . (**string**)
price	قیمت کتاب. (از نوع اعداد حقیقی)

جدول Titles

price	imageFile	copyright	publisherID	editionNumber	Title	ISBN #
$69.95	jhtp3.jpg	2000	1	3	Java How to Program (Java 2)	0130125075
$49.95	gsvc.jpg	1999	1	1	Getting Started with Visual C++ 6 with an Introduction to MFC	0130132497
$69.95	iw3htp1.jpg	2000	1	1	Internet and World Wide Web How to Program	0130161438
$69.95	xmlhtp1.jpg	2001	1	1	XML How to Program	0130284173
$69.95	perlhtp1.jpg	2001	1	1	Perl How to Program	0130284181
$69.95	ebechtp1.jpg	2001	1	1	e-Business and e-Commerce How to Program	013028419X
$69.95	vbnet.jpg	2002	1	2	Visual Basic .NET How to Program	0130293636
$69.95	iw3htp2.jpg	2002	1	2	Internet and World Wide Web How to Program	0130308978
$69.95	ebecm.jpg	2000	1	1	e-Business and e-Commerce for Managers	0130323640
$69.95	jhtp4.jpg	2002	1	4	Java How to Program	0130341517
$69.95	cshtp.jpg	2002	1	1	C# How To Program	0130622214
$69.95	wireless.jpg	2001	1	1	Wireless Internet and Mobile Business How to Program	0130622265
$109.95	javactc4.jpg	2002	2	4	The Complete Java Training Course	0130649341
$99.95	javactc2.jpg	1998	2	2	The Complete Java Training Course (Java 1.1)	0130829277
$109.95	vbctc1.jpg	1999	2	1	The Complete Visual Basic 6 Training Course	0130829293
$109.95	javactc3.jpg	2000	2	3	The Complete Java 2 Training Course	0130852481
$109.95	iw3ctc1.jpg	2000	2	1	The Complete Internet and World Wide Web Programming Training Course	0130856118
$109.95	ebecctc.jpg	2001	2	1	The Complete e-Business & e-Commerce Programming Training Course	0130895512
$109.95	perl.jpg	2001	2	1	The Complete Perl Training Course	0130895547
$109.95	xmlctc.jpg	2001	2	1	The Complete XML Programming Training Course	0130895563
$69.95	advjhtp1.jpg	2002	1	1	Advanced Java 2 Platform How to Program	0130895601
$109.95	iw3ctc2.jpg	2001	2	2	The Complete Internet & World Wide Web Programming Training	013089561X

price	imageFile	copyright	publisherID	editionNumber	Title	ISBN #
					Course	
$109.95	cppctc3.jpg	2001	2	3	The Complete C++ Training Course	0130895636
$69.95	cpphtp3.jpg	2001	1	3	C++ How to Program	0130895717
$69.95	chtp3.jpg	2001	1	3	C How to Program	0130895725
$69.95	python.jpg	2002	1	1	Python How to Program	0130923613
$69.95	cpphtp1.jpg	1994	1	1	C++ How to Program	0131173340
$69.95	chtp.jpg	1992	1	1	C How to Program	0131180436
$49.95	chtp2.jpg	1994	1	2	C How to Program	0132261197
$109.95	javactc.jpg	1998	2	1	Java Multimedia Cyber Classroom	0132719746
$69.95	vbhtp1.jpg	1999	1	1	Visual Basic 6 How to Program	0134569555
$49.95	cpphtp2.jpg	1998	1	2	C++ How to Program	0135289106
$0.00	jhtp1.jpg	1998	1	1	Java How to Program	0136325890
$109.95	javactc2.jpg	1998	2	2	The Complete Java Training Course	0137905696
$49.95	jhtp2.jpg	1998	1	2	Java How to Program (Java 1.1)	0138993947
$109.95	cppctc2.jpg	1998	2	2	The Complete C++ Training Course	0139163050

در شکل زیر رابطه بین جداول موجود در پایگاه داده **Book** نشان داده شده است. خط اول در هر جدول بیانگر نام جـدول اسـت. فیلدهایی که بصورت **Bold** نشان داده شده اند، بیانگر فیلدهای **Primary Key** هستند. همانطور که قبلا نیز گفته شد، فیلدهای **Primary Key** نمایش دهنده رکوردهایی منحصر بفرد در جدول هستند، از اینرو هر رکورد در جدول، میبایست دارای مقـداری در فیلد **Primary Key** خود باشد و این مقدار میبایست منحصر بفرد باشد. از این اصل، بعنوان قانون "*جامعیت موجودیتها*" یاد میشود (**Rule Of Entity Integrity**). توجه نمایید که در جدول **AuthorISBN** دو فیلد بعنوان **Primary Key** نـشان داده است، که این بدین معناست که در این جدول **Primary Key** مرکب وجود دارد، از اینرو هر رکورد در این جدول میبایست دارای **authorID** و **isbn** منحصر بفردی (بطور مشترک) باشد. برای مثال، ممکن است رکوردهای بسیاری در جدول یافـت شـوند کـه مقدار فیلد **authorID** آنها برابر با ۲ باشد، اما چون ترکیب **authorID** و **isbn** میبایست منحصر بفرد باشـد، از اینـرو هـیچ دو رکوردی نباید یافت شود که دارای **authorID** و **isbn** مشابه باشد. همچنین ممکن است رکوردهای مختلفی مقـدار فیلـد **isbn** آنها برابر با ۰۱۳۰۸۹۵۶۰۱ باشد، اما تنها یک فیلد یافت میشود که مقـدار **authorID** آن برابـر بـا ۲ و مقـدار **isbn** آن برابـر بـا ۰۱۳۰۸۹۵۶۰۱ باشد.

توجه : در صورتیکه مقداری برای فیلد **Primary Key** انتخاب نشود، اصل جامعیت موجودیتهـا نقـض شـده، از اینـرو **DBMS** خطایی را گزارش میکند.

توجه : وارد کردن مقادیر مشابه برای فیلد **Primary Key** باعث می شود تا **DBMS** خطایی را گزارش نماید.

خطهای رسم شده بین جداول، رابطه (Relationship) بین جداول را نشان می دهد. برای مثال خط رسم شده بین جدول Publisher و Titles را در نظر بگیرید. همانطور که مشاهده میشود، در سمت Publisher، بر روی خط عدد ۱ قرار دارد و در سمت Titles علامت ∞ (بینهایت) قرار گرفته است. خط رسم شده بین این دو جدول، بیانگر یک رابطه "*یک به چند*" (One-to-Many) است که بیان میدارد، به ازای هر ناشر در جدول Publishers، تعداد زیادی کتاب می تواند در جدول Titles قرار داشته باشد. همچنین توجه داشته باشید که خط رسم شده بین این دو جدول، از فیلد publisherID در جدول Publisher آغاز شده و به فیلد publisherID در جدول Titles ختم شده است. فیلد publisherID در جدول Titles یک Foreign Key است، بدین معنا که به ازای هر موجودیت در این جدول، یک مقدار منحصر بفرد در جدولی دیگر وجود دارد و این مقدار منحصر بفرد فیلد Primary Key جدول دوم است. Foreign Key در زمان ایجاد یک جدول مشخص می شود. با استفاده از Foreign Key، اصل "*جامعیت ارجاع*" (Rule Of Referential Integrity) مطرح می شود که بیان میدارد، مقدار هر Foreign Key باید در فیلد Primary Key جدولی دیگر وجود داشته باشد. با استفاده از Foreign Key میتوان اطلاعات موجود در جداول مختلف را با یکدیگر "*پیوند*" (join) زد و از آنها استفاده نمود. همواره رابطه ای "یک به چند" بین Primary Key و Foreign Key وجود دارد، بدین معنا که مقدار فیلد Foreign Key می تواند در جدول خودش چندین بار ظاهر شود اما در جدول دیگر تنها می تواند یکبار و آنهم بعنوان Primary Key ظاهر شود.

توجه : همانطور که گفته شد، همیشه رابطه ای "یک به چند" از سوی Primary Key به سمت Foreign Key وجود دارد.

توجه : استفاده از مقداری بعنوان فیلد Foreign Key که در Primary Key هیچ جدولی وجود نداشته باشد، اصل جامعیت ارجاع را نقض کرده، از اینرو DBMS خطایی را گزارش خواهد کرد.

زبان پرس و جوی ساخت یافته (SQL)

در این قسمت، به بررسی مختصر زبان SQL خواهیم پرداخت. همانند زبانهای برنامه سازی، زبان SQL نیز از کلمات کلیدی و دستوراتی تشکیل شده است که دستورات کلیدی و مهم آنرا در جدول زیر مشاهده می کنید.

نام دستور SQL	*توضیحات*
SELECT	فیلدهای مورد نظر را از جدول یا جداولی انتخاب می کند.
FROM	جداولی را مشخص می کند که اطلاعات از آنها باید انتخاب شود و یا قرار است اطلاعات از آنها خدف گردد. در هر دستور SELECT و DELETE وجود دارد.
WHERE	شرایطی را مشخص می کند که تحت آن رکوردهایی خاص انتخاب خواهند شد.
INNER JOIN	رکوردهای متفاوتی را از جداول مختلف به یکدیگر متصل مینماید و مجموعه ای جدید از رکوردها را ایجاد مینماید.
GROUP BY	شرایطی را نشان میدهد که بوسیله آن دسته بندی اطلاعات انجام میشود.
ORDER BY	شرایطی را نشان میدهد که طی آن اطلاعات مرتب می شوند.
INSERT	داده را در جدولی خاص وارد می کند.
UPDATE	داده خاصی را در جدول مورد نظر بروز رسانی می کند. (تغییر میدهد)
DELETE	اطلاعات خاصی را از جدول مورد نظر حذف میکند.

دستور SELECT

یکی از ساده ترین دستورات در SQL، دستوری است که در آن اطلاعات از جدولی خاص انتخاب می شود. چنین عملی با استفاده از دستور SELECT صورت میگیرد و ساده ترین فرمت این دستور بصورت زیر است :

```
SELECT * FROM tableName
```

که در آن علامت "*" بیان میدارد که تمامی اطلاعات (تمامی ستونها) موجود در جدول انتخاب شده و به نمایش در خواهند آمد و *tableName* نام جدولی است که میخواهیم اطلاعات را از آن انتخاب نماییم.

```
SELECT * FROM Authors
```

برای انتخاب فیلد یا فیلدهایی خاص از یک جدول، میتوانیم نام فیلدهای مورد نظر آن جدول را در جلوی دستور SELECT و بجای * قرار دهیم و آنها را با کاما "," از یکدیگر جدا نماییم.

```
SELECT authorID, lastName FROM Authors
```

تنها فیلد authorID و lastName از جدول Authors را نشان میدهد.

نکته : در صورتیکه نام فیلد شامل فضای خالی (Space) باشد، در اینصورت میبایست نامو فیلد را داخل دو براکت "[]" قرار دهیم. (SELECT [author ID] FROM Authors)

دستور WHERE

در اکثر موارد کاربر در جدول به دنبال اطلاعاتی میگردد که دارای شرایط خاصی است. در این موارد تنها آن رکوردهایی که با شرایط مورد نظر کاربر همخوانی دارند میبایست نمایش داده شوند. SQL با استفاده از دستور WHERE در دستور SELECT، شرایط مورد نظر را اعمال می نماید. ساده ترین فرم دستور SELECT که بهمراه WHERE باشد، بصورت زیر است :

```
SELECT fieldName1,fieldName2,… FROM tableName
WHERE criteria
```

که در آن *fieldName*، نام فیلدهای مورد نظر، *tableName* نام جدولی که اطلاعات از آن استخراج می شود و *criteria* شرایطی است که بر اساس آن جستجو در جدول صورت میپذیرد. برای مقال، درصورتیکه بخواهیم عناوین کتابهایی را بیابیم که تاریخ نشر آنها بعد از سال ۱۹۹۹ است، از دستور SELECT زیر استفاده می کنیم :

```
SELECT title
FROM Titles
WHERE copyright > 1999
```

نکته : در زبان C# برای اجرای دستورات SQL از یک رشته استفاده می کنیم که این رشته حاوی کل دستور (Query) مورد نظر ما است. همچنین برای بالا رفتن خوانایی برنامه، دستورات SQL را در خطوط جدا مینویسیم.

دستور WHERE می تواند خاوی عملگرهای <، >، =<، =>، = و <> و عملگر LIKE باشد. عملگر LIKE برای "*تطبیق الگو*" (Pattern Matching) مورد استفاده قرار میگیرد و بهمراه * و ? مورد استفاده قرار میگیرد. با استفاده از تطبیق الگو، میتوان به دنبال رشته ای گشت که حاوی الگوی مورد نظر است. برای مثال، query زیر، تمامی رکوردهایی از جدول Authors را نشان میدهد که lastName آنها با کاراکتر "D" شروع شده باشند :

```
SELECT * FROM Authors
WHERE lastName LIKE 'D*'
```

توجه نمایید، استفاده از کاراکتر * بیان میدارد که برای تطبیق الگو تنها کافیست تا کاراکتر اول مقدار D داشته باشد و سایر کاراکترها هر مقداری می توانند داشته باشند. همچنین اهمیتی ندارد که چه تعدا کاراکتر بعد از کاراکتر D در رشته وجود دارد و تنها اهمیت برای ما وجود کاراکتر D در ابتدای رشته است.

نکته : تمامی سیستم های پایگاه داده از عبارت LIKE پشتیبانی نمیکنند.

نکته : در اکثر سیستمهای پایگاه داده، به جای استفاده از کاراکتر "*" در عبارت **LIKE**، از کاراکتر "%" استفاده میشود.
نکته : در برخی از سیستمهای پایگاه داده، کاراکترهای رشته، **Case Sensitive** هستند.
نکته : بهتر است برای تمیز دادن دستورات و عبارات **SQL**، آنها را با حروف بزگ بنویسیم.

توجه نمایید در رشته ای که مورد تطبیق الگو قرار میگیرد، درصورتیکه کاراکتر "?" قرار گیرد، بدین معناست که بجای کاراکتر "?" تنها یک کاراکتر قرار خواهد گرفت. برای مثال در **query** زیر، تمامی رکوردهایی از جدول **Authors** نمایش داده می شوند که **lastName** آنها با هر کاراکتری شروع شده باشد و به دنبال آن کاراکتر "i" آمده باشد و بعد از آن هر تعداد کارارکتر قرار داشته باشد :

```
SELECT * FROM Authors
WHERE lastName LIKE '?i*'
```

نکته : اکثر پایگاه های داده، به جای استفاده از کاراکتر "?" از کاراکتر "_" در عبارت **LIKE** استفاده میکنند.
نکته : توجه نمایید که پس از عبارت **LIKE**، رشته مورد نظر درون یک جفت ' ' قرار میگیرد.

دستور ORDER BY

نتیجه یک **query** می تواند بصورت صعودی یا نزولی مرتب گردد. این کار با استفاده از دستور **ORDER BY** انجام میشود. ساده ترین فرم این دستور بصورت زیر است :

```
SELECT fieldName1, fieldName2, … FROM tableName
ORDER BY field ASC

SELECT fieldName1, fieldName2, … FROM tableName
ORDER BY field DESC
```

که در ***field***، نام فیلدی است که نتایج **query** بر اساس آن مرتب می شود. در صورتیکه بخواهیم نتایج بصورت صعودی مرتب شوند از کلمه **ASC** و درصورتیکه بخواهیم نتایج بصورت نزولی مرتب شوند از کلمه **DESC** استفاده می نماییم. بطور پیش فرض، نتایج **query** بصورت صعودی نمایش داده میشوند.

```
SELECT * FROM Authors
ORDER BY lastName DESC
```

همچنین، با استفاده از **ORDER BY** میتوان نتایج **query** را بر اساس چند فیلد مختلف نیز مرتب نمود :

```
ORDER BY field1 sortingOrder, field2 sortingOrder, …
```

که در آن ***field***، فیلدهایی هستند که مرتب سازی بر اساس آنها صورت میگیرد و ***sortingOrder*** نوع مرتب سازی هر فیلد را نشان میدهد. توجه نمایید، در مواردی که دو یا چند فیلد برای مرتب سازی در نظر گرفته شده اند، ابتدا مرتب سازی بر اساس فیلد اول انجام می شود و پس از آن مرتب سازی بر اساس فیلدهای دیگر به پیش میرود. در مثال زیر،ابتدا نتایج **query** بر اساس نام خانوادگی (**lastName**) و سپس بر اساس نام (**firstName**) مرتب میشوند :

```
SELECT * FROM Authors
ORDER BY lastName, firstName
```

توجه نمایید که در یک **query** می توان از ترکیب دستورات **WHERE** و **ORDER BY** استفاده نمود :

```
SELECT isbn, title, price FROM Titles
WHERE title LIKE '%How To Program'
ORDER BY title DESC
```

ادغام داده ها از جداول مختلف : INNER JOIN

طراحان پایگاه داده، معمولا اطلاعات را به جداول مختلفی تقسیم میکنند تا مطمئن باشند که در پایگاه داده اطلاعات بیهوده و زائد ذخیره نمیشود. برای مثال، پایگاه داده **Book** شامل دو جدول **Authors** و **Titles** است. ما با استفاده از جدول **AuthorISBN**، لینکی بین نویسنده های مختلف و عناوین کتبی که نوشته اند برقرار میکنیم. در صورتیکه این اطلاعات را به جداول مختلف تقسیم نمیکردیم مجبور بودیم تا اطلاعات نویسنده را برای هر عنوان کتاب در جدول **Titles** در نظر بگیریم و از اینرو یکسری اطلاعات اضافی و تکراری در جدول ذخیره میشد چراکه برای عناوین کتبی که نویسنده آنها یکسان است، اطلاعات نویسنده بطور تکراری ذخیره میگردید.
همچنین برای مقاصد آنالیز و تجزیه و تحلیل، ضروری است تا اطلاعات مختلف از جداول مختلف با یکدیگر ادغام شوند و یک مجموعه اطلاعاتی جدید را ایجاد نمایند. با استفاده از عملی تحت عنوان "*ادغام جداول*" ، این عمل قابل اجرا است و بوسیله **INNER JOIN** در دستور **SELECT** آنرا در **SQL** عملی میکنیم. یک **INNER JOIN** رکوردهای مختلفی را از جداول متفاوت با یکدیگر ادغام میکند و این عمل را از طریق تست کردن مقادیر فیلدهایی انجام میدهد که در جداول مورد نظر عمومیت دارند. ساده ترین فرم **INNER JOIN** بشکل زیر است :

```
SELECT fieldName1, fieldName2, …
FROM table1
INNER JOIN table2
ON table1.fieldName = table2.fieldName
```

که در آن قسمت بعد از عبارت **ON**، فیلدهایی را نشان میدهد از دو جدول با یکدیگر مقایسه میشوند تا معین شود چه رکوردهایی با یکدیگر ادغام میشوند. برای مثال، **query** زیر لیستی از تمام نویسنده ها به همراه **ISBN** مربوط به کتابهای نوشته شده توسط آنها را ایجاد میکند :

```
SELECT firstName, lastName, isbn
FROM Authors
INNER JOIN AuthorISBN
ON Authors.authorID = AuthorISBN.authorID
ORDER BY lastName, firstName
```

این **query** فیلدهای **firstName** و **lastName** از جدول **Authors** را با فیلد **isbn** از جدول **AuthorISBN** ادغام میکند و نتیجه را بر اساس فیلدهای **lastName** و **firstName** بصورت صعودی مرتب مینماید. در این **query** به فرمت عبارت نوشته شده بعد از عبارت **ON** که بصورت *tableName.fieldName* است توجه نمایید. در این قسمت مشخص میشود که عمل ادغام بین دو جدول از طریق کدام فیلد انجام خواهد گرفت. استفاده از فرمت نوشتاری "*tableName.*" در مواردی ضروری است که نام فیلد در هر دو جدول یکسان باشد.

ادغام اطلاعات از جداول Authors، AuthorISBN، Titles و Publisher

در اینجا می خواهیم برای پایگاه داده **Book** یک **query** تعریف کنیم که توسط آن عنوان کتاب، شماره **ISBN**، نام نویسنده، نام خانوادگی نویسنده، سال انشار کتاب و نام ناشر هر کتاب نمایش داده شود. برای کتابهایی که بیش از یک نویسنده دارند، این **query** میبایست رکوردهای مجزایی را برای هر نویسنده نمایش دهد. توجه نمایید که در این **query** نیاز است تا بین هر چهار جدول ادغام صورت گیرد. این **query** در زیر نمایش داده شده است :

```
SELECT Titles.Title, Titles.ISBN, Authors.FirstName, Authors.LastName,
Publishers.PublisherName
FROM
      (
       Publishers
       INNER JOIN Titles
```

```
        ON Publishers.PublisherID = Titles.PublisherID
       )
INNER JOIN
       (
        Authors
        INNER JOIN AuthorISBN
        ON Authors.AuthorID = AuthorISBN.AuthorID
       )

ON Titles.ISBN = AuthorISBN.ISBN
ORDER BY Titles.Title;
```

نتیجه این **query** در شکل زیر نمایش داده شده است :

نتیجه **query**

TitleAuthor				
PublisherName	**LastName**	**FirstName**	**ISBN #**	**Title**
Prentice Hall	Deitel	Paul	0130895601	Advanced Java 2 Platform How to Program
Prentice Hall	Deitel	Harvey	0130895601	Advanced Java 2 Platform How to Program
Prentice Hall	Santry	Sean	0130895601	Advanced Java 2 Platform How to Program
Prentice Hall	Deitel	Harvey	0131180436	C How to Program
Prentice Hall	Deitel	Paul	0131180436	C How to Program
Prentice Hall	Deitel	Harvey	0132261197	C How to Program
Prentice Hall	Deitel	Paul	0132261197	C How to Program
Prentice Hall	Deitel	Harvey	0130895725	C How to Program
Prentice Hall	Deitel	Paul	0130895725	C How to Program
Prentice Hall	Yaeger	Cheryl	0130622214	C# How To Program
Prentice Hall	Deitel	Harvey	0130622214	C# How To Program
Prentice Hall	Nieto	Tem	0130622214	C# How To Program
Prentice Hall	Zlatkina	Marina	0130622214	C# How To Program
Prentice Hall	Listfield	Jeffrey	0130622214	C# How To Program
Prentice Hall	Deitel	Paul	0130622214	C# How To Program
Prentice Hall	Deitel	Paul	0130895717	C++ How to Program
Prentice Hall	Deitel	Harvey	0130895717	C++ How to Program
Prentice Hall	Deitel	Paul	0131173340	C++ How to Program
Prentice Hall	Deitel	Harvey	0131173340	C++ How to Program
Prentice Hall	Deitel	Harvey	0135289106	C++ How to Program
Prentice Hall	Deitel	Paul	0135289106	C++ How to Program
Prentice Hall	Deitel	Harvey	0130323640	e-Business and e-Commerce for Managers
Prentice Hall	Steinbuhler	Kate	0130323640	e-Business and e-Commerce for Managers
Prentice Hall	Deitel	Paul	0130323640	e-Business and e-Commerce for Managers
Prentice Hall	Deitel	Harvey	013028419X	e-Business and e-Commerce How to Program
Prentice Hall	Deitel	Paul	013028419X	e-Business and e-Commerce How to Program
Prentice Hall	Nieto	Tem	013028419X	e-Business and e-Commerce How to Program
Prentice Hall	Deitel	Paul	0130132497	Getting Started with Visual C++ 6 with an

TitleAuthor				
PublisherName	LastName	FirstName	ISBN #	Title
				Introduction to MFC
Prentice Hall	Deitel	Harvey	0130132497	Getting Started with Visual C++ 6 with an Introduction to MFC
Prentice Hall	Deitel	Paul	0130161438	Internet and World Wide Web How to Program
Prentice Hall	Nieto	Tem	0130161438	Internet and World Wide Web How to Program
Prentice Hall	Nieto	Tem	0130308978	Internet and World Wide Web How to Program
Prentice Hall	Deitel	Paul	0130308978	Internet and World Wide Web How to Program
Prentice Hall	Deitel	Harvey	0130161438	Internet and World Wide Web How to Program
Prentice Hall	Deitel	Harvey	0130308978	Internet and World Wide Web How to Program
Prentice Hall	Deitel	Harvey	0136325890	Java How to Program
Prentice Hall	Deitel	Paul	0136325890	Java How to Program
Prentice Hall	Deitel	Harvey	0130341517	Java How to Program
Prentice Hall	Deitel	Paul	0130341517	Java How to Program
Prentice Hall	Deitel	Harvey	0138993947	Java How to Program (Java 1.1)
Prentice Hall	Deitel	Paul	0138993947	Java How to Program (Java 1.1)
Prentice Hall	Deitel	Paul	0130125075	Java How to Program (Java 2)
Prentice Hall	Deitel	Harvey	0130125075	Java How to Program (Java 2)
Prentice Hall PTG	Deitel	Harvey	0132719746	Java Multimedia Cyber Classroom
Prentice Hall PTG	Deitel	Paul	0132719746	Java Multimedia Cyber Classroom
Prentice Hall	Deitel	Harvey	0130284181	Perl How to Program
Prentice Hall	McPhie	David	0130284181	Perl How to Program
Prentice Hall	Nieto	Tem	0130284181	Perl How to Program
Prentice Hall	Deitel	Paul	0130284181	Perl How to Program
Prentice Hall	Wiedermann	Ben	0130923613	Python How to Program
Prentice Hall	Deitel	Harvey	0130923613	Python How to Program
Prentice Hall	Liperi	Jonathan	0130923613	Python How to Program
Prentice Hall	Deitel	Paul	0130923613	Python How to Program
Prentice Hall PTG	Deitel	Paul	0139163050	The Complete C++ Training Course
Prentice Hall PTG	Deitel	Harvey	0139163050	The Complete C++ Training Course
Prentice Hall PTG	Deitel	Harvey	0130895636	The Complete C++ Training Course
Prentice Hall PTG	Deitel	Paul	0130895636	The Complete C++ Training Course
Prentice Hall PTG	Deitel	Paul	0130895512	The Complete e-Business & e-Commerce Programming Training Course
Prentice Hall PTG	Nieto	Tem	0130895512	The Complete e-Business & e-Commerce Programming Training Course
Prentice Hall PTG	Deitel	Harvey	0130895512	The Complete e-Business & e-Commerce Programming Training Course
Prentice Hall PTG	Deitel	Harvey	013089561X	The Complete Internet & World Wide Web Programming Training Course
Prentice Hall PTG	Deitel	Paul	013089561X	The Complete Internet & World Wide Web Programming Training Course
Prentice Hall PTG	Nieto	Tem	013089561X	The Complete Internet & World Wide Web

TitleAuthor				
PublisherName	LastName	FirstName	ISBN #	Title
				Programming Training Course
Prentice Hall PTG	Deitel	Harvey	0130856118	The Complete Internet and World Wide Web Programming Training Course
Prentice Hall PTG	Deitel	Paul	0130856118	The Complete Internet and World Wide Web Programming Training Course
Prentice Hall PTG	Nieto	Tem	0130856118	The Complete Internet and World Wide Web Programming Training Course
Prentice Hall PTG	Deitel	Paul	0130852481	The Complete Java 2 Training Course
Prentice Hall PTG	Deitel	Harvey	0130852481	The Complete Java 2 Training Course
Prentice Hall PTG	Deitel	Paul	0130649341	The Complete Java Training Course
Prentice Hall PTG	Deitel	Harvey	0130649341	The Complete Java Training Course
Prentice Hall PTG	Deitel	Paul	0137905696	The Complete Java Training Course
Prentice Hall PTG	Deitel	Harvey	0137905696	The Complete Java Training Course
Prentice Hall PTG	Deitel	Harvey	0130829277	The Complete Java Training Course (Java 1.1)
Prentice Hall PTG	Deitel	Paul	0130829277	The Complete Java Training Course (Java 1.1)
Prentice Hall PTG	McPhie	David	0130895547	The Complete Perl Training Course
Prentice Hall PTG	Deitel	Harvey	0130895547	The Complete Perl Training Course
Prentice Hall PTG	Deitel	Paul	0130895547	The Complete Perl Training Course
Prentice Hall PTG	Nieto	Tem	0130895547	The Complete Perl Training Course
Prentice Hall PTG	Deitel	Harvey	0130829293	The Complete Visual Basic 6 Training Course
Prentice Hall PTG	Deitel	Paul	0130829293	The Complete Visual Basic 6 Training Course
Prentice Hall PTG	Nieto	Tem	0130829293	The Complete Visual Basic 6 Training Course
Prentice Hall PTG	Nieto	Tem	0130895563	The Complete XML Programming Training Course
Prentice Hall PTG	Deitel	Paul	0130895563	The Complete XML Programming Training Course
Prentice Hall PTG	Deitel	Harvey	0130895563	The Complete XML Programming Training Course
Prentice Hall	Nieto	Tem	0130293636	Visual Basic .NET How to Program
Prentice Hall	Deitel	Harvey	0130293636	Visual Basic .NET How to Program
Prentice Hall	Deitel	Paul	0130293636	Visual Basic .NET How to Program
Prentice Hall	Deitel	Harvey	0134569555	Visual Basic 6 How to Program
Prentice Hall	Nieto	Tem	0134569555	Visual Basic 6 How to Program
Prentice Hall	Deitel	Paul	0134569555	Visual Basic 6 How to Program
Prentice Hall	Steinbuhler	Kate	0130622265	Wireless Internet and Mobile Business How to Program
Prentice Hall	Nieto	Tem	0130622265	Wireless Internet and Mobile Business How to Program
Prentice Hall	Deitel	Paul	0130622265	Wireless Internet and Mobile Business How to Program
Prentice Hall	Deitel	Harvey	0130622265	Wireless Internet and Mobile Business How to Program
Prentice Hall	Deitel	Paul	0130284173	XML How to Program

TitleAuthor				
PublisherName	LastName	FirstName	ISBN #	Title
Prentice Hall	Deitel	Harvey	0130284173	XML How to Program
Prentice Hall	Sadhu	Praveen	0130284173	XML How to Program
Prentice Hall	Lin	Ted	0130284173	XML How to Program
Prentice Hall	Nieto	Tem	0130284173	XML How to Program

حال به بررسی این **query** میپردازیم. در خطوط ۱و۲ این **query**، فیلدهای مورد نظر که بعنوان نتیجه در خروجی نمایش داده میشوند، قرار گرفته اند. توجه نمایید که ترتیب قرار گرفتن فیلدها در این لیست، از این جهت اهیمت می یابد که فیلدها در خروجی به همین ترتیب نمایش داده خواهند شد. این **query**، فیلدهای **title** و **isbn** از جدول **Titles**، فیلدهای **lastName** و **firstName** از جدول **Authors**، فیلد **copyright** از جدول **Titles** و فیلد **publisherName** از جدول **Publishers** را انتخاب مینماید. برای اینکه متن این **query** خوانا تر باشد، قبل از نام هر فیلد نام جدول آنرا نیز آورده ایم. (به این روش **Fully Qualified Name** گفته میشود.) در خطهای بعدی، عمل ادغام بین جداول مختلف صورت گرفته است. در این **query** از سه عمل **INNER JOIN** استفاده شده است. باید به نکته توجه شود، اگرچه **INNER JOIN** بر روی دو جدول انجام میشود، اما نتیجه آن می تواند بعنوان ورودی برای **INNER JOIN** دیگری مورد استفاده قرار گیرد چراکه نتیجه یک **INNER JOIN** خود یک جدول است. همچنین برای اولویت بخشیدن به اجرای دستورات از پرانتز استفاده نموده ایم. در ابتدا با دستور زیر آغاز میکنیم :

```
(Publishers INNER JOIN Titles
  ON Publishers.publisherID = Titles.publisherID)
```

که جدول **Publishers** و **Titles** را با استفاده از شرایطی که در آن فیلدها **publisherID** دو جدول با یکدیگر منطبق هستند، ادغام میکند. جدول موقتی که از انجام این **INNER JOIN** تشکیل میشود حاوی اطلاعات کتاب و ناشر مربوط به آن است.
دیگر **INNET JOIN** این **query** عبارت است از :

```
( Authors
 INNER JOIN AuthorISBN
 ON Authors.AuthorID = AuthorISBN.AuthorID )
```

که جدول **Authors** و **AuthorISBN** را در شرایطی که فیلد **AuthorID** با یکدیگر منطبق باشند، ادغام مینماید.
سومین **INNER JOIN** در این **query** عبارت است از :

```
(
 Publishers
 INNER JOIN Titles
 ON Publishers.PublisherID = Titles.PublisherID
)
INNER JOIN
(
 Authors
 INNER JOIN AuthorISBN
 ON Authors.AuthorID = AuthorISBN.AuthorID
)

ON Titles.ISBN = AuthorISBN.ISBN
```

که باعث ادغام دو جدول موقتی ایجاد شده از دو INNER JOIN قبلی، تحت شرایطی میشود که در آن فیلد Titles.isbn برای هر رکورد در اولین جدول موقت با فیلد AuthorISBN.isbn برای هر رکورد از جدول موقت دوم، منطبق باشند. نتیجه نهایی این INNER JOIN نیز، جدول موقتی است که نتیجه مورد نظر را در بر دارد.
در انتهای query نیز، نتایج بر اساس فیلد Titles.title مرتب میشوند.

دستور INSERT

این دستور باعث وارد کردن رکوردی جدید در جدول میشود. ساده ترین فرم این دستور بشکل زیر است :

```
INSERT INTO tableName (fieldName1, fieldName2, … )
VALUES (value1, value2, …)
```

که در آن *tableName* جدولی است که در آن رکورد جدید وارد میشود. به دنبال *tableName* لیستی از فیلدها قرار میگیرد و پس از آن عبارت VALUES و به دنبال آن لیستی از مقادیر. توجه نمایید که به ازای هر *fieldName*، مقدار *value* در جدول وارد میشود. همچنین باید توجه شود که ترتیب و نوع مقادیر وارد شده در *value* میبایست با نام و نوع *field* مورد نظر همخوانی کامل داشته باشد. برای مثال، اگر *fieldName1* بعنوان firstName در نظر گرفته شود، آنگاه مقدار *value1* باید رشته ای باشد که درون یک جفت
' ' قرار گرفته و بیانگر نام باشد. بعنوان یک مثال، query زیر را در نظر بگیرید :

```
INSER INTO Authors (lastName, firstName)
VALUES ('Smith' , 'Sue')
```

این query باعث میشود تا رکوردی جدید در جدول Authors وارد شود. توجه نمایید در دستور INSERT حتما برای فیلدهای Primary Key میبایست مقداری وارد گردد، اما در اینجا چون فیلد Primary Key، فیلد authorID است و این فیلد از نوع auto-increment تعریف شده، نیازی به مقداردهی ندارد، چراکه با وارد شده هر رکورد جدید بطور خودکار مقداری برای آن در نظر گرفته میشود. همچنین دقت کنید، بغییر از فیلدهای Primary Key، در صورتیکه برای فیلدی از یک جدول در دستور INSERT مقداری در نظر گرفته نشود، مقدار آن فیلد تهی (Null) در نظر گرفته خواهد شد.

نکته : SQL برای نشان دادن رشته ها از یک جفت ' ' استفاده میکند. برای نشان دادن رشته هایی که خود حاوی ' هستند (مانند O'Reily) باید از فرمت 'O' 'Reily' استفاده شود.

دستور UPDATE

این دستور باعث ایجاد تغییر در داده های جدول میشود و ساده ترین فرم آن بصورت زیر است :

```
UPDATE tableName
SET fieldName1 = value1, fieldName2 = value2, …
WHERE criteria
```

که در آن *fieldName* هایی که مقادیر آنها تغییر میبابند مشخص شده و مقادیر جدید (*value*) به آنها تخصیص داده میشود. وجود عبارت WHERE در دستور UPDATE باعث میشود تا رکوردهایی که این فیلدها در آنها تغییر می یابند معین شوند.
برای مثال :

```
UPDATE Authors
SET lastName = 'Jones'
WHERE lastName = 'Smith' AND firstName = 'Sue'
```

باعث میشود تا رکوردی که در آن Sue Smith قرار داشته است، به Sue Jones تغییر یابد.

توجه : عدم استفاده از عبارت **WHERE** پس از دستور **UPDATE** ممکن است باعث بروز خطایی منطقی گردد.

دستور DELETE

دستور **DELETE** باعث خذف داده ها از جدول میشود و ساده ترین فرم آن بصورت زیر است :

```
DELETE FROM tableName
WHERE criteria
```

که در آن **criteria** مشخص کننده رکورد یا رکوردهایی است که از جدول حذف میشوند.

```
DELETE FROM Authors
WHERE lastName = 'Jones' AND firstName = 'Sue'
```

باعث خذف شدن رکوردی میشود که نام و نام خانوادگی آن **Sue Jones** است.

نکته : عبارت **WHERE** ممکن است با چندین رکورد در جدول مورد نظر همخوانی داشته باشد، از اینرو در دستور **DELETE** دقت نمایید که شرط نوشته شده در **WHERE** تنها با رکورد مورد نظر شما همخوانی داشته باشد تا از خذف شدن ناخواسته اطلاعات جلوگیری شود.

مدل شیء ADO.NET (ADO.Net Object Model)

مدل شیء **ADO.Net** فراهم کننده **API** ای است که بوسیله آن می توان از طریق کد نویسی به پایگاه داده دسترسی پیدا کرد. **ADO.Net** مخصوص **Net Framework**. طراحی شده و نسل جدید **ADO** میباشد. (کلمه **ADO** مخفف **ActiveX Data Objects** است.)

در **Net Framework**، **System.Data** فضایی است که در آن تمامی **API** های مربوط به **ADO.Net** قرار دارند. **Namespace** های پایه **ADO.Net** عبارتند از **System.Data.OleDb** و **System.Data.SqlClient** که با استفاده از آنها امکان دسترسی و ایجاد تغییر در **Data Source** های مختلف فراهم میگردد. (منظور از **Data Source** هر منبعی داده ایست که میتوان در آن اطلاعات و داده ها را ذخیره نمود.)
System.Data.OleDb شامل کلاسهایی است که بوسیله آنها می توان به هر **Data Source** ای متصل شد، در حالیکه **System.Data.SqlClient** تنها شامل کلاسهایی جهت استفاده از **MS SQL Server 2000** است.

شروع برنامه نویسی با ADO.Net–اتصال به MS Access

برای آغاز کار برنامه نویسی و اتصال به پایگاه داده، با پایگاه داده **Access** شروع میکنیم که پایگاه داده ای ساده بوده و به همراه بسته نرم افزاری **MS Office** قابل نصب است. توجه کنید در برنامه زیر، از ساده ترین روش برای اتصال به پایگاه داده استفاده شده و مطالب پیشرفته تر را در آینده بررسی خواهیم نمود.

برای اتصال به پایگاه داده **Access** میبایست از کلاسهای موجود در **System.Data.OleDb** استفاده نماییم.
مراحل کار با پایگاه داده بشرح زیر میباشد :

- اتصال به پایگاه داده
- بدست آوردن اطلاعات مورد نظر از پایگاه داده
- نمایش اطلاعات بدست آمده

- قطع ارتباط با پایگاه داده

برای اتصال به پایگاه داده Access از شیء OleDbConnection استفاده میکنیم. توجه کنید که برای اتصال به هر پایگاه داده ای نیاز به یکسری پارامترها و مسیرها است که این اطلاعات در رشته ای تحت عنوان connection string قرار داده میشود و از طریق اطلاعات این رشته به پایگاه داده متصل میشویم. برای اتصال به Access نیز میبایست از یک connection string استفاده نماییم. برای مثال، رشته زیر را در نظر بگیرید :

```
string connectionString =
  "provider=Microsoft.Jet.OLEDB.4.0;" +
  "data source=F:\\Samples\\Books.mdb";
```

این رشته، نمونه ای از یک connection string جهت برقراری ارتباط با پایگاه داده Access است. در قسمت provider نام شرکت پشتیبانی کننده پایگاه داده نوشته میشود که برای Access این مقدار عبارت است از Microsoft.Jet.OLEDB.4.0 و در قسمت data source محل قرار گرفتن پایگاه داده مشخص میگردد. (محل فیزیکی که در آن پایگاه داده ذخیره شده است.) همچنین به استفاده از "\\" جهت نمایش "\" توجه نمایید.

در مثال زیر، برنامه نمونه ای مطرح شده که در آن نحوه برقراری ارتباط با پایگاه داده Access از طریق شیء OledbConnection به نمایش در آمده است.

```
using System;
using System.Data;
using System.Data.OleDb;

class OleDbConnectionAccess
{
  public static void Main()
  {
    // بیان رشته ای که در آن جزئیات ارتباط با پایگاه داده مشخص میشود.
    string connectionString = "provider=Microsoft.Jet.OLEDB.4.0;" +
                              "data source=F:\\Samples\\Books.mdb";

    //ایجاد شیء جدیدی از OleDbConnection جهت برقراری ارتباط با پایگاه
    //.که به سازنده آن connectionString ارسال میشود
    OleDbConnection myOleDbConnection =
            new OleDbConnection(connectionString);

    // ایجاد شیء OleDbCommand
    OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();

    // قرار دادن ویژگی CommandText از شیء OleDbCommand به دستور SQL
    //.که توسط آن یک سطر از جدول Authors انتخاب میشود
    myOleDbCommand.CommandText =
      "SELECT * "+
      "FROM Authors "+
      "WHERE authorID=1";

    // باز کردن ارتباط بین پایگاه داده با استفاده از متد Open()
    myOleDbConnection.Open();
```

```
    // ایجاد شیء OleDbDataReader و فراخوانی متد ExecuteReader() از
    // شیء OleDbCommand جهت اجرای دستور SELECT
      OleDbDataReader myOleDbDataReader =
                              myOleDbCommand.ExecuteReader();

    // خواندن یک سطر از OleDbDataReader
    // با استفاده از متد Read()
    myOleDbDataReader.Read();

    // نمایش نتیجه و خروجی
    Console.WriteLine("myOleDbDataReader[\" firstName\"] = "+
      myOleDbDataReader["firstName"]);
    Console.WriteLine("myOleDbDataReader[\" lastName\"] = "+
      myOleDbDataReader["lastName"]);
    Console.WriteLine("myOleDbDataReader[\" AuthorID\"] = "+
      myOleDbDataReader["authorID"]);

    // بستن شیء OleDbDataReader با استفاده از متد Close()
    myOleDbDataReader.Close();

    // بستن ارتباط بین پایگاه داده از شیء OleDbConnection
    myOleDbConnection.Close();
  }
}
```

خروجی این برنامه بشکل زیر است :

```
myOleDbDataReader[" firstName"] = Harvey
myOleDbDataReader[" lastName"] = Deitel
myOleDbDataReader[" AuthorID"] = 1
```