- It’s impossible to build a modern, full-featured website without database.
- Access databases using a query language called SQL (Structured Query Language).
- ADO.NET Entity Framework provides easier access to database operations directly from code (as an additional layer on top of your database to access data in an object-oriented way with minimal code).
- Database is a collection of data that is arranged so it can be accessed, managed, and updated easily.
- The most popular type of database is the relational database. The relational database is not the only one. Other types exist, including flat-file, NoSQL, object-relational, and object-oriented databases, but these are less common in Internet applications.
- A relational database has the notion of tables, where data is stored in rows and columns, much like a spreadsheet. Each row in a table contains the complete information about an item that is stored in the table. Each column, on the other hand, contains information about a specific property of the rows in the table.
- The term “relational” refers to the way the different tables in the database can be related to each other. Instead of duplicating the same data over and over again, you store repeating data in its own table and then create a relationship between that table and other tables.
- You can use many different kinds of databases in your ASP.NET projects, including Microsoft Access, SQL Server, Oracle, SQLite, and MySQL. However, the most commonly used database in ASP.NET websites is probably Microsoft SQL Server.
    - Link to download: https://www.microsoft.com/en-us/download/details.aspx?id=29062

Working on example:

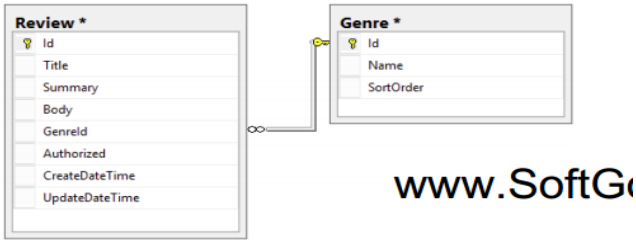

FIGURE 12-24

- retrieving and manipulating data With sQl: CRUD operaions (Create, Read, Update, and Delete)
- SELECT Id, Name FROM Genre
- SELECT Id AS GenreId, Name FROM Genre
- SELECT TOP 3 Id, Name FROM Genre ORDER BY Name
- SELECT Id FROM Genre WHERE Name = 'Grunge'

- SELECT Name FROM Genre WHERE Id = 8

OPERATOR	DESCRIPTION
=	The *equals* operator matches only when the left side and the right side of the comparison are identical.
>	The *greater than* operator matches when the left side of the comparison represents a larger value than the right side.
>=	The *greater than or equal* operator matches when the left side of the comparison is equal to or larger than the right side.
<	The *less than* operator matches when the left side of the comparison represents a smaller value than the right side.
<=	The *less than or equal* operator matches when the left side of the comparison is equal to or smaller than the right side.

OPERATOR	DESCRIPTION
AND	Enables you to join two expressions. For example, the `WHERE` clause `WHERE Id > 20 AND Id < 30` gives you all rows with IDs that fall between 20 and 30 (with 20 and 30 themselves not included).
OR	Enables you to define multiple criteria of which only one has to match (although more matches are allowed). For example, the `WHERE` clause `WHERE GenreId = 5 OR GenreId = 8` gives you all the rows with a `GenreId` of 5 or 8.
BETWEEN	Enables you to specify a range of values that you want to match with a lower and upper bound. For example, `WHERE Id BETWEEN 10 AND 35` gives you all rows whose IDs are between 10 and 35 (including 10 and 35 themselves if they exist in the database).
LIKE	Used to determine if a value matches a specific pattern. You can use wildcards like `%` to match any string of zero or more characters, and the underscore (_) to match a single character. For example, the `WHERE` clause `WHERE Name LIKE '%rock%'` returns all genres that have `rock` in their name, including Indie Rock, Hard Rock, and so on.

- To combine multiple WHERE criteria use logical operators.
- If no rows match the WHERE clause, you don't get an error, but you simply get zero results back.
- ORDER BY clause comes at the end of the SQL statement and can contain one or more column names or expressions, which can optionally include ASC or DESC to determine if items are sorted in ascending order (with ASC, which is the default if you leave out the keyword) or in descending order (using DESC).

- SELECT Id, Name FROM Genre ORDER BY Name

- SELECT Id, Name FROM Genre ORDER BY Name DESC

- SELECT Id, Name FROM Genre ORDER BY SortOrder DESC

- SELECT Id, Name, SortOrder FROM Genre WHERE (Id > 4) AND (Name LIKE '%rock%') ORDER BY SortOrder DESC

- SELECT Review.Id, Review.Title, Genre.Name FROM Review INNER JOIN Genre ON Review.GenreId = Genre.Id

- SELECT Genre.Id, Genre.Name, Review.Title FROM Genre LEFT OUTER JOIN Review ON Genre.Id = Review.GenreId

- Besides the LEFT OUTER JOIN, there is also a RIGHT OUTER JOIN that returns all the rows from the table listed at the right side of the JOIN.

- **Creating data:**

- INSERT INTO Genre (Name, SortOrder) VALUES ('Tribal House', 20)

- **updating data**

- UPDATE Genre SET Name = 'Trance', SortOrder = 5 WHERE Id = 13

- **deleting data**

  If you leave out the WHERE clause, all rows will be deleted from the table.
- DELETE FROM Genre WHERE Id = 13
- When you identify a column as a primary key, the database engine ensures that no two rows can end up with the same value.
- An identity column is a numeric column whose sequential values are generated automatically whenever a new row is inserted.
- Create DateTime column once and then on the Column Properties grid, type getdate()in the field for the Default Value or Binding property, as shown in Figure 12-22. This inserts the current date and time for new rows if you don’t supply an explicit value.
- With a proper relationship set up, the database will stop you from accidentally deleting rows in one table that still have other rows attached to it.