Chapter – 6: Working with Database

Data:

Data refers to any collection of facts, numbers, statistics, observations, measurements, or any other kind of information that is gathered through observation, experience, or experimentation. Data is the raw material that is used to generate information and knowledge.

In the context of computer science, data refers to digital information that is stored, manipulated, and transmitted by computers. This can include text, images, audio, video, and other forms of digital information.

Data can be classified as structured or unstructured. **Structured data** is data that has a defined format, such as a database table or spreadsheet. **Unstructured data** is data that does not have a predefined format, such as text documents, emails, or social media posts.

Data is the foundation of many applications and is used to generate insights and knowledge that can help businesses make informed decisions. With the growth of big data, machine learning, and artificial intelligence, the importance of data has only continued to increase, and it is now considered one of the most valuable assets for many organizations.

Database:

A database is an organized collection of data that is stored and managed in a way that allows it to be easily accessed, updated, and queried. It is a digital repository that stores and organizes large amounts of data and provides a way to retrieve that data when it is needed.

Databases are used in many different types of software applications, from basic inventory management systems to complex web applications and enterprise-level systems. They are essential for storing and managing large amounts of data, and are designed to be efficient, scalable, and secure.

- **There are different types of databases, including:**

1. Relational databases: These are the most common type of database, and are based on the relational model. Data is stored in tables with predefined columns and rows, and relationships can be established between tables.

2. **NoSQL databases:** These are non-relational databases that are designed to handle large volumes of unstructured or semi-structured data.

3. **Object-oriented databases:** These are databases that store data as objects, which can include attributes and methods.

Databases are typically accessed through a database management system (DBMS), which provides a set of tools for managing the data. Some common DBMS include MySQL, Microsoft SQL Server, Oracle, and PostgreSQL. Databases are an essential component of many software applications, and are used to store and manage critical business data.

Q. Why we need Database:

Databases are an essential part of many software applications, including those written in C#. Here are some reasons why we need databases in C#:

1. Data Persistence: Applications need to store data in a way that it can be accessed later. Databases provide a way to persist data over time, so that it can be stored and retrieved by the application as needed.

2. Data Management: Databases provide a way to manage data in a structured and organized way. This allows the application to query and manipulate the data more easily.

3. Scalability: As the amount of data stored by an application grows, the need for a robust and scalable data storage solution becomes more important. Databases are designed to handle large amounts of data and support high levels of concurrency.

4. Security: Databases provide a way to secure data by enforcing access controls and encrypting sensitive data. This is especially important for applications that handle personal or sensitive information.
5. Performance: Databases can be optimized for performance by using techniques such as indexing, caching, and partitioning. This helps ensure that the application can access and manipulate data quickly and efficiently.

In C#, databases are typically accessed using ADO.NET, a data access technology provided by Microsoft. ADO.NET provides a set of classes that enable C# applications to connect to a database, execute SQL commands, and retrieve and manipulate data. Using a database in C# can greatly improve the functionality and performance of an application, and is a critical component of many software projects.

● Introduction to ADO.NET

ADO.NET is a set of classes (a framework) to interact with data sources such as databases and XML files. ADO is the acronym for ActiveX Data Objects. It allows us to connect to underlying data or databases. It has classes and methods to retrieve and manipulate data.

The following are a few of the .NET applications that use ADO.NET to connect to a database, execute commands and retrieve data from the database.

- ASP.NET Web Applications
- Console Applications
- Windows Applications.

● ADO.NET consists of two main components:

1. Connected data access: In this mode, a connection is established to the data source and data is retrieved or updated in a connected manner. The data is typically retrieved into a dataset or data reader.

2. Disconnected data access: In this mode, data is retrieved from the data source and loaded into a dataset or a data table. The data can then be modified locally in the dataset, and the changes can be persisted back to the data source at a later time.

ADO.NET is a set of .NET libraries and APIs that are used to access and manipulate data from different data sources, including relational databases, XML documents, and flat files. ADO.NET provides a consistent programming model for accessing data from different data sources, making it easier for developers to work with data in their .NET applications.

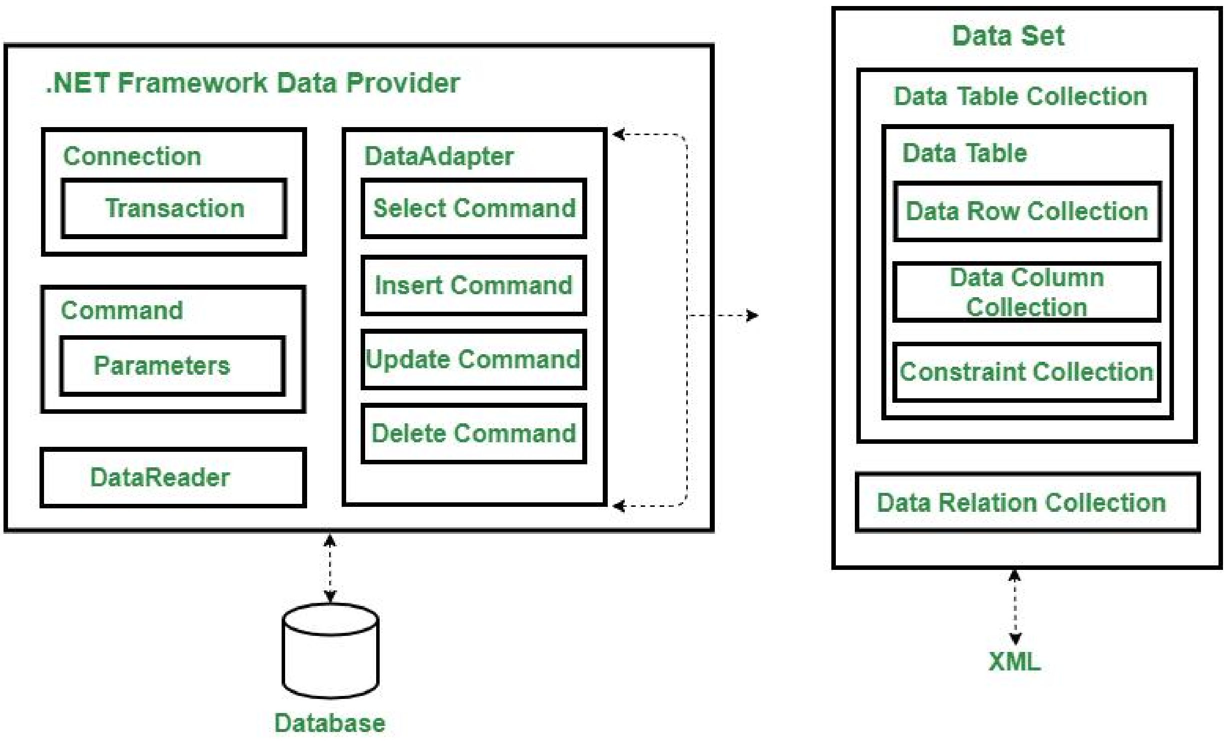
ADO.NET supports a variety of data providers that can be used to access different types of data sources. For example, the SQL Server data provider can be used to access SQL Server databases, while the OLE DB data provider can be used to access a variety of other data sources, including Oracle, Access, and Excel.

❖ Architecture of ADO.NET :

ADO.NET uses a multi-layered architecture that revolves around a few key concepts as :

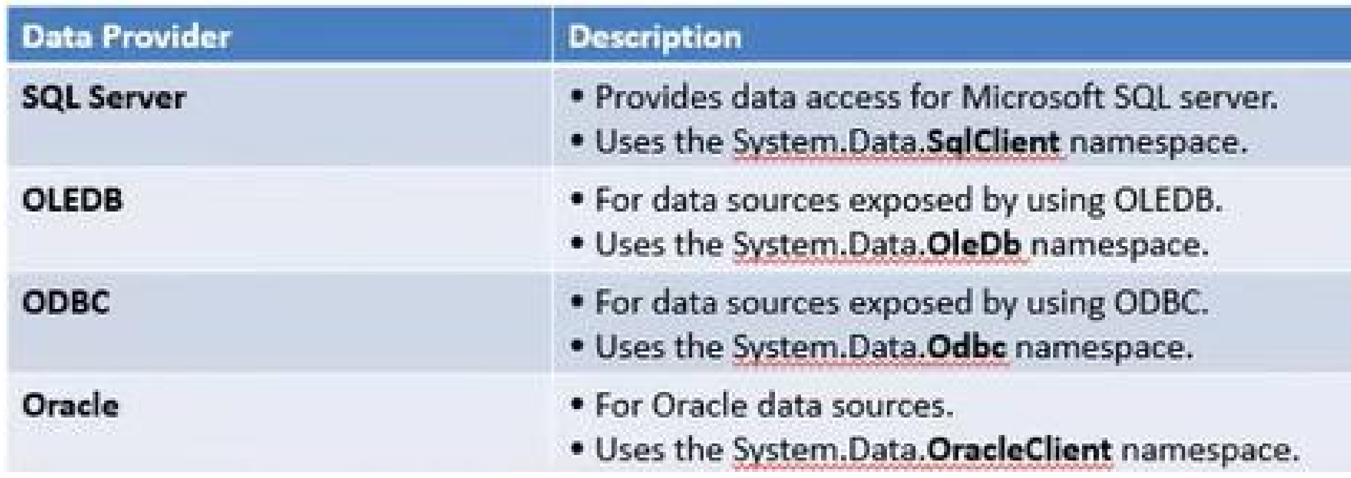
- asConnection
- Command
- DataSet objects

The ADO.NET architecture is a little bit different from the ADO, which can be shown from the following figure of the architecture of ADO.NET.



❖ DataProvider consist of following classes:

- Connection Class: used to establish connection
- Command Class: used to execute sql query
- DataReader Class: used to read the result-set in connected manner
- DataAdapter Class: Interface between Dataset and Database



Data Provider	Description
SQL Server	• Provides data access for Microsoft SQL server. • Uses the System.Data.**SqlClient** namespace.
OLEDB	• For data sources exposed by using OLEDB. • Uses the System.Data.**OleDb** namespace.
ODBC	• For data sources exposed by using ODBC. • Uses the System.Data.**Odbc** namespace.
Oracle	• For Oracle data sources. • Uses the System.Data.**OracleClient** namespace.

❖ **The features of ADO.NET:**

1. Interoperability:

- ADO.NET uses XML for data exchange, making it easy to edit and understand since XML is a text-based format.

2. Maintainability:
- ADO.NET promotes the separation of data logic and user interface, allowing for the creation of applications in independent layers, enhancing maintainability.

3. Programmability (Typed Programming):
- It follows a programming style where we use user-friendly words to construct statements. For instance, accessing the "Marks" column for "Ram" in the "Student" table is done like this:

```
DataSet.Student("Ram").Marks
```

4. Performance:
- ADO.NET utilizes a disconnected data architecture, handling most tasks on the client-side. This improves performance by reducing the load on the database.

5. Scalability:
- ADO.NET encourages efficient resource use, allowing multiple users to access data simultaneously without degrading performance.

❖ Setting up a database environment:

It involves several steps to ensure that you have a stable and functional system for storing and managing your data. Below are the general steps involved in setting up a database environment:

- **Define Requirements:**

Understand the requirements of your application and determine the type of data you need to store, the volume of data, and the expected workload.

- **Select a Database Management System (DBMS):**

Choose a database management system that best fits your requirements. Common options include:

- Relational Database Management Systems (RDBMS) like MySQL, PostgreSQL, SQL Server, Oracle.
- NoSQL databases like MongoDB, Cassandra, Redis, Couchbase.

● **Choose Hosting Options:**

Decide whether you want to host the database on-premises or use cloud-based solutions like AWS RDS, Google Cloud SQL, or Azure Database.

● **Install the DBMS:**

Install the selected DBMS software on the server where you plan to host your database. Follow the installation instructions provided by the database vendor.

● **Configure the DBMS:**

Configure the DBMS settings according to your requirements. This includes setting up database users, authentication methods, network configurations, etc.

● **Create Databases and Tables:**

Once the DBMS is installed and configured, create the databases and tables required for your application. Define the schema for each table, including data types, constraints, indexes, etc.