1. Introduction to Database

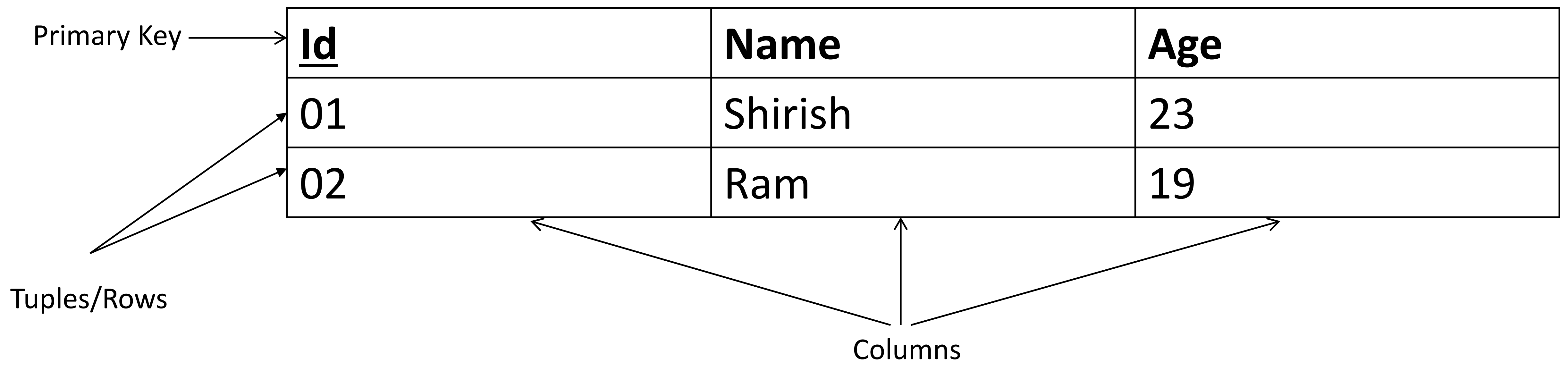
- collection of data that is saved and organized to allow easy retrieval when needed
- collection of schemas, tables, queries, reports, views, and other objects

2. Types of database

1. Relational Database
2. Non Relational Database

3. Database Concepts

1. Table : Collection of rows and column.
2. Column: Represent a field or property.
3. Rows/Tuples: Represents a single entry.



Id	**Name**	**Age**
01	Shirish	23
02	Ram	19

4. Keys

- **Primary Key** : column that uniquely identifies tuples (rows) in that table
- Foreign Key : columns of a table that points to the primary key of another table
- Super Key : set of one or more columns (attributes) to uniquely identify rows in a table
- Candidate Key : column that can uniquely identify a row.
- Alternate Key : Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys

5. SQL Basic

- Select Command
- Insert Command
- Update Command
- Delete Command

Select Command

- display all or selected records from a table.

Syntax:

a. SELECT <field1>,<field2>.....<fieldN> FROM <table_name>

b. SELECT * FROM <table_name> // * represents all the field name

c. SELECT * FROM <table_name> WHERE <Experssion>

Example:

a. SELECT * FROM Student where id<10 // display all the fields with only those records whose id is less than 10

Insert Command

- used to insert new record into a table

Syntax:

a. INSERT INTO <table_name> (field1,field2-----fieldN) VALUES (value1, value2----------valueN)

b. INSERT INTO <table_name> values (value1, value2...........valueN)

Example:

a. INSERT INTO Student values(1,’Ram’,10)

b. INSERT INTO Student (id,name) values (2,’Ram’) // NULL value will be inserted for roll field

Update Command

- used to modify selected or all records from a table

Syntax:

a. UPDATE <table_name> SET field1=newvalue1, field2=newvalue2.........fieldN=newvalueN

b. UPDATE <table_name> SET field1=newvalue2, field2=newvalue2.........fielnN=newvaluN WHERE <Expression>

Example:

a. UPDATE Student SET roll=5

b. UPDATE Student SET roll=5 WHERE name=‘ram’

Delete Command

- delete all or selected records from a table

Syntax:

a. DELETE FROM <table_name> // deletes all records from table

b. DELETE FROM <table_name> WHERE <Expression>

Example:

a. DELETE FROM tbl_student // delete entire records from tbl_student

b. DELETE from tbl_student WHERE id>10 // delete all records whose id >10

6. SQLite

- lightweight, file-based relational database

Features:

a. Light weight

b. Serverless

c. Cross platform

7. ORM (Object Relational Mapping)

- Technique that maps database tables to programming objects
- creates a bridge between object-oriented programs and database

Popular ORM tools for .NET:

a. EF Core

b. NHibernate

c. Dapper

Advantages of ORM:

a. Abstraction of Database Complexity

b. Maintenance and Scalability

c. Portability

d. Change Tracking

e. Built-in Querying Support like LINQ

8. Entity Framework Core

- Open-source, lightweight and extensible
- Cross-platform ORM
- Migrations
- LINQ support
- Change Tracking and Audit Log
- Improved performance (i.e. Lazy Loading, Eager Loading and Explicit Loading)

Setup EF Core:

a. Install Nuget Packages:
   - Microsoft.EntityFrameworkCore
   - Microsoft.EntityFrameworkCore.SqlServer
   - Microsoft.EntityFrameworkCore.Design
   - Microsoft.EntityFrameworkCore.Tools

b. Create the Model classes

c. Configure Connection Strings

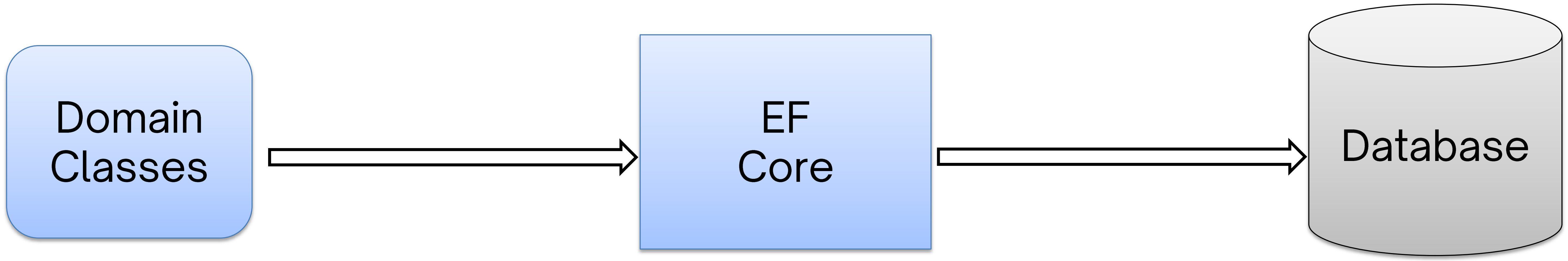
d. Create the DbContext class

e. Register DbContext in Program.cs

f. Use EF Migrations to create or update the database

9. Code-First Approach

- In this approach, database schema is designed using classes.
- First a class is defined then EF core will map that class to a table in database.



10. Database Relationships

a. One-to-One

b. One-to-Many

c. Many-to-Many

One-to-one Relationship

- used when one entity is associated with at most one other entity

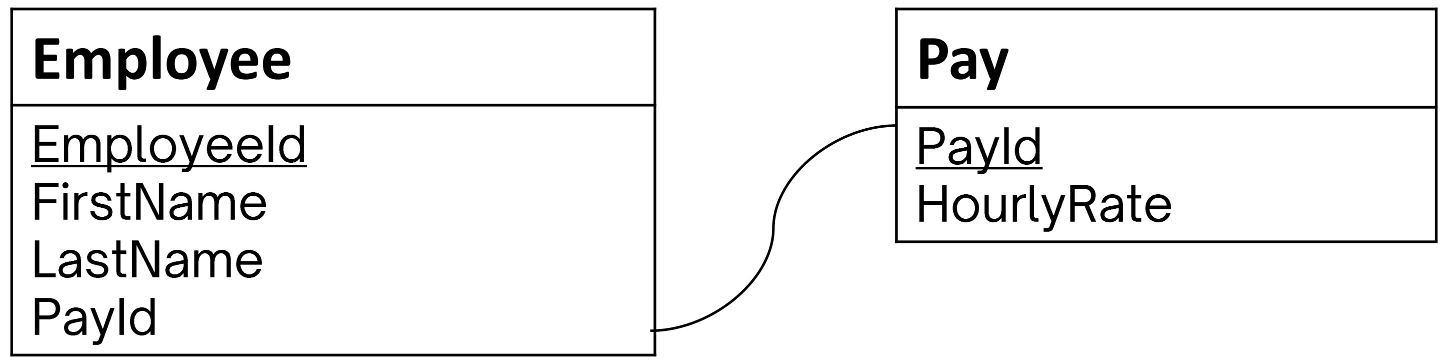
```
public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }

    // Navigation property to pay
    public Pay Pay { get; set; }
}
```

```
public class Pay
{
    public int Id { get; set; }

    public decimal HourlyRate { get; set; }

    // back nagivation to employee
    public Employee Employee { get; set; }
}
```



Many-to-Many Relationship

- used when any number entities of one entity type is associated with any number of entities of the same or another entity type

EmployeeModel.cs

```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }

    // Navigation property
    public ICollection<EmployeePay> EmployeePay { get; set; }
}
```

EmployeePay.cs

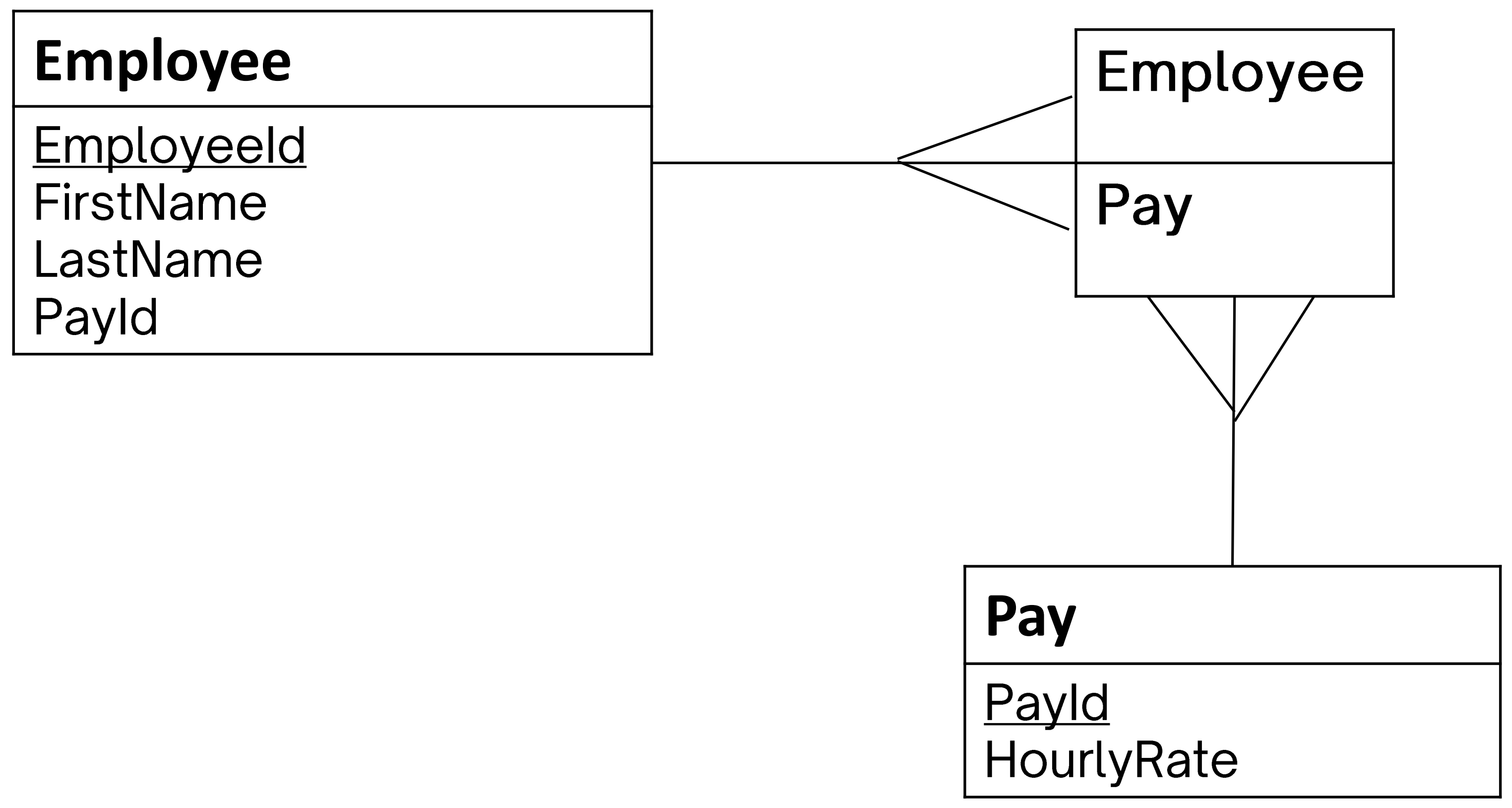
```
public class EmployeePay
{
    public int EmployeeId { get; set; }
    public Employee Employee { get; set; }

    public int PayId { get; set; }
    public Pay Pay { get; set; }
}
```

PayModel.cs

```
public class Pay
{
    public int Id { get; set; }
    public decimal HourlyRate { get; set; }

    // Navigation property
    public ICollection<EmployeePay> EmployeePay { get; set; }
}
```



11. Eager and Lazy Loading

Eager Loading

- Loads related data immediately along with the main entity.
- Achieved using ‘ Include( ) ’ Method.
- Good when you know you will need related data.
- Reduces multiple queries

Lazy Loading

- Loads related data only when accessed
- Requires virtual navigation properties and lazy loading proxies.
- Good when related data is not always needed.
- Uses multiple queries