Yes, Entity Framework (EF) is an Object-Relational Mapping (ORM) framework, and one of its key functionalities is managing the relationships between database tables by representing them as relationships between objects in your code.

In a relational database, tables are connected through relationships, such as one-to-one, one-to-many, or many-to-many. Entity Framework allows you to model these relationships in your code using C# classes and properties.

Here's a brief overview of how Entity Framework handles relationships:

1. \*\*One-to-One Relationship:\*\*

- In the database, this is often represented by having a foreign key in one table that refers to the primary key in another.

- In EF, you would represent this using navigation properties in your model classes.

2. \*\*One-to-Many Relationship:\*\*

- In the database, this is typically represented by having a foreign key in one table that refers to the primary key in another.

- In EF, you would represent this using navigation properties as well, but in this case, you might have a collection navigation property in one class that represents the "many" side.

3. \*\*Many-to-Many Relationship:\*\*

- In the database, this is often implemented using a junction table that connects the primary keys of two other tables.

- In EF, you represent this using navigation properties and collections on both sides of the relationship.

Here's a simple example to illustrate:

```csharp

public class Author

{

public int AuthorId { get; set; }

public string Name { get; set; }

// One-to-Many relationship

public List<Book> Books { get; set; }

}

public class Book

{

public int BookId { get; set; }

public string Title { get; set; }

// Many-to-One relationship

public int AuthorId { get; set; }

public Author Author { get; set; }

}

```

In this example, the `Author` class has a one-to-many relationship with the `Book` class, and the `Book` class has a many-to-one relationship with the `Author` class. The relationships are established through the `AuthorId` and `Books` properties.

Entity Framework uses conventions and configurations to understand these relationships and generate appropriate database schema and SQL queries to maintain and retrieve related data.