ASP.NET CORE 2.2 MVC – Plan and develop your software using this modern framework

Thiago Medeiros

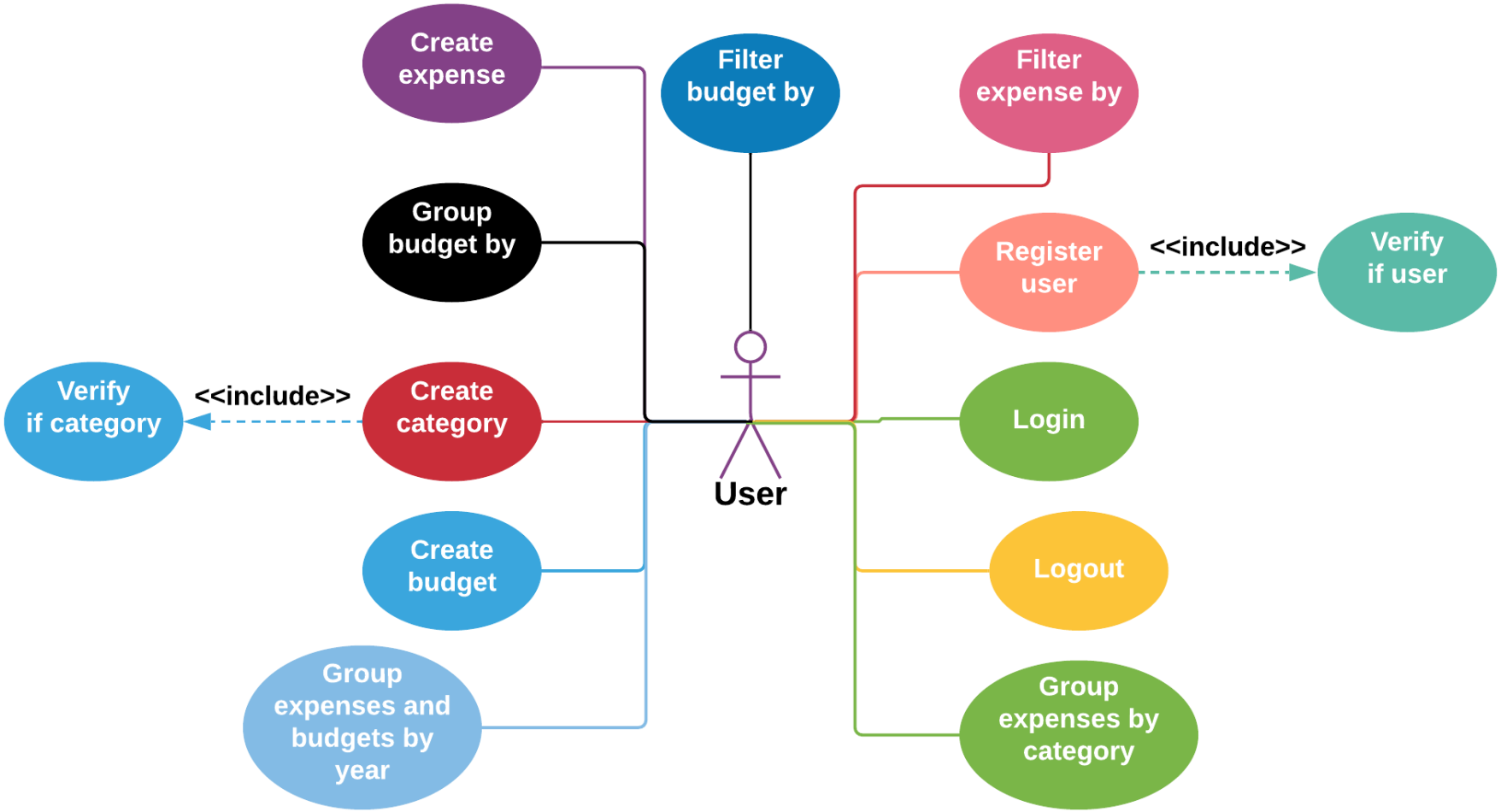
UML

- It is a visual language used to model software using Object Orientation;
- It became an international standard used on Software Enginnering;
- It helps software engineers to understand the feature of softwares, like their comportments, functionalities and structure;
- Actual Version : 2.5.1

UML

- This is not a programming language;
- It is independent of any kind of technology and platforms;
- The Use case diagram will be used in this course;

Use Case Diagram



Use Case Diagram

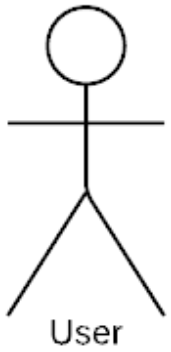
- It is a Behavior diagram used to help discovering the functionalities of the software. The functionalities are explained by its users;

- Users can easily understand them and realize how the software will work;

- It identifies the actors and functionalities of the software.

Use Case Diagram

- **Actors** : Represents the users of the system. An user can be a human or another system.

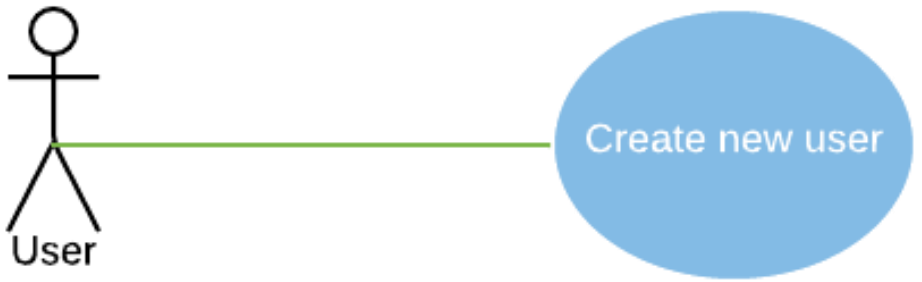


Use Case Diagram

- **Use case** : Use cases are functionalities, tasks or services that can be used on the system. Create a product is an example.

Use Case Diagram

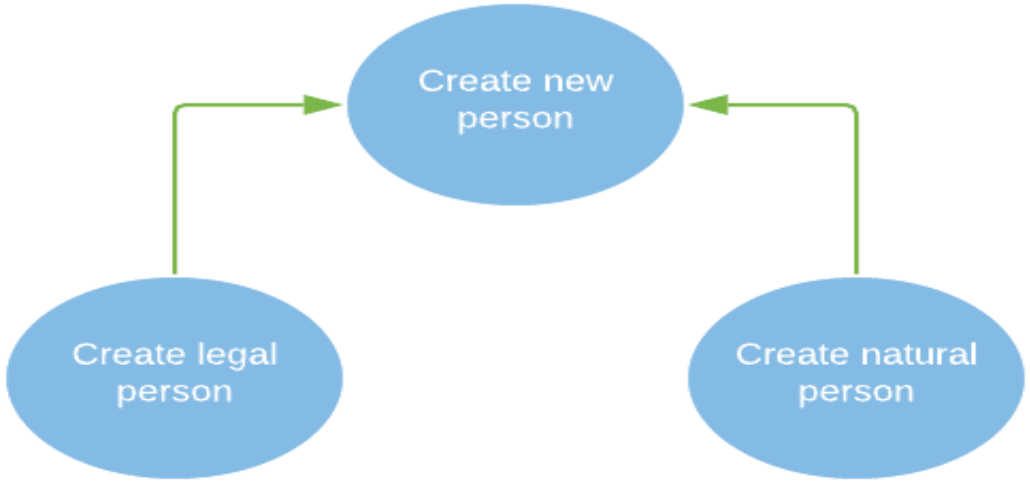
- **Associations** : They are relationships between users and use cases or between use cases.



Use Case Diagram

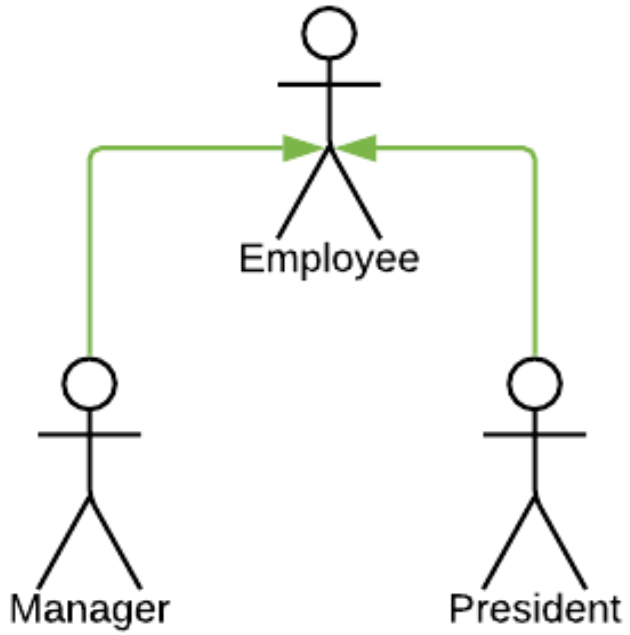
- **Generalization**

- **Generalization between use cases** : In this kind of relationship, the child use cases inherits all the features of the father use case, like association and documentation. New features can be included on it.



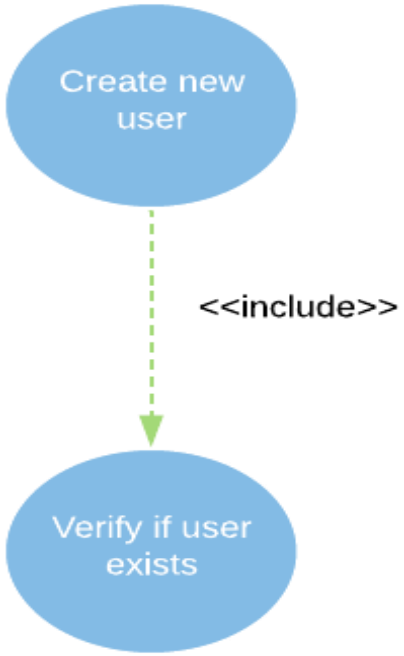
Use Case Diagram

- **Generalization between actors** : This is an association between actors that can inherit features of another actor. The child actor inherits all the features, documentation and associations of the father actor. New feature can be added to it.



Use Case Diagram

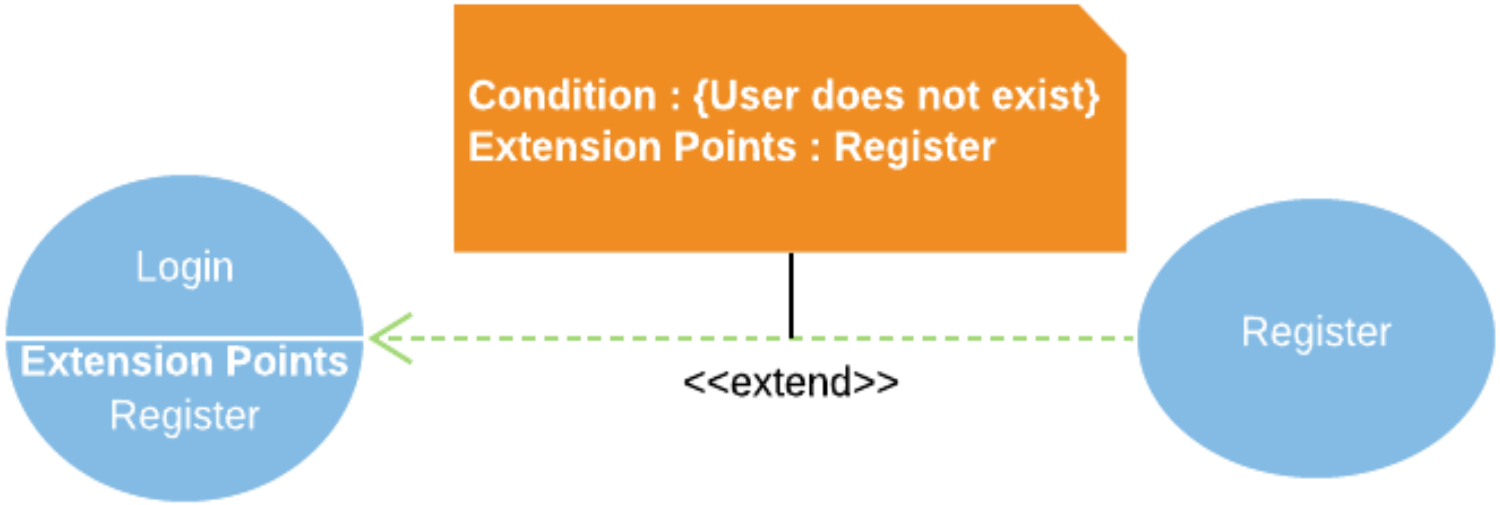
- **Include** : The execution of a use case will do another use case execute too.



- On this example, “Verify if user exists” use case will be executed when “Create new user" be executed.

Use Case Diagram

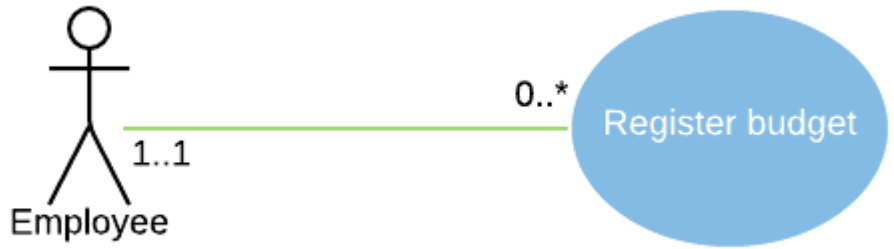
- **Extend** : If an use case is executed, there will be a condition to another use case be executed.



- If “Login “ is executed, “Register” will be executed due a condition.

Use Case Diagram

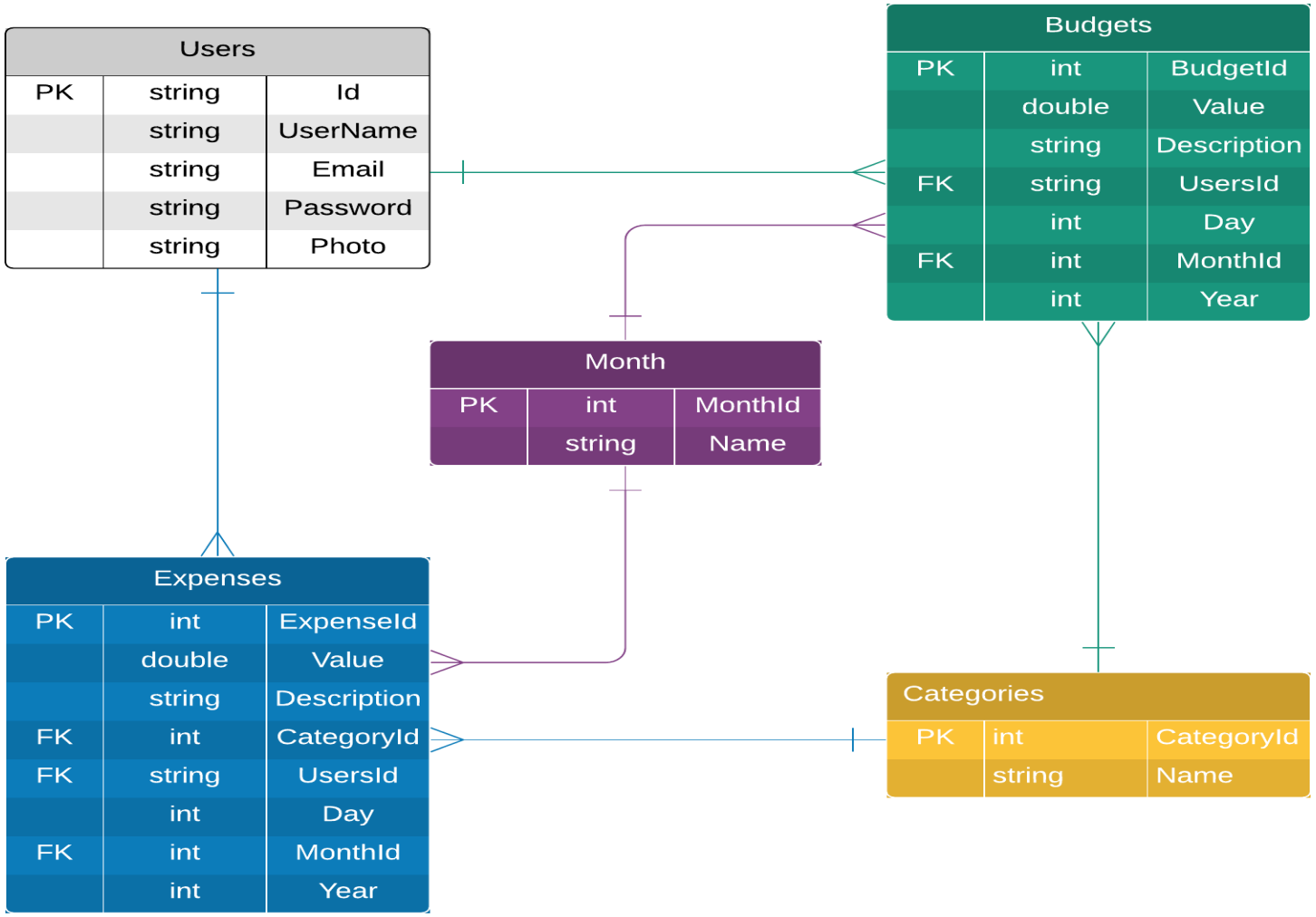
- **Multiplicity** : Specifies the number of times that an actor can use a use case.



Let's practice

- An organization needs a software to it's clients managing their finances. This software needs to allow registering expenses and earnings (budgets);
- They need to describe the form that they earned that budget on the system;
- Categories are necessary to group budgets and expenses;
- Clients want to filter their expenses by month and years grouped by categories;
- They also want to filter their budgets by month and years grouped by days. An user should be able to register and log in / out on the system without any help;
- They want to see the total of budgets and expenses and see if they have more budgets or more expenses;

Entity Relationship Diagram

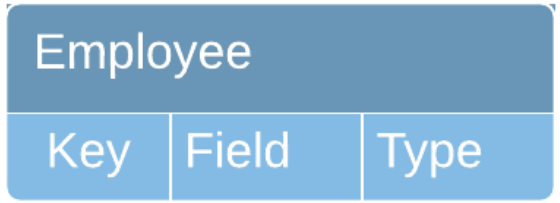


Entity Relationship Diagram

- It is a diagram used to design databases;
- It has the entities of a database, it means the tables;
- It contains diferents symbols and connectors;
- It allows us seeing the relationships between entities.

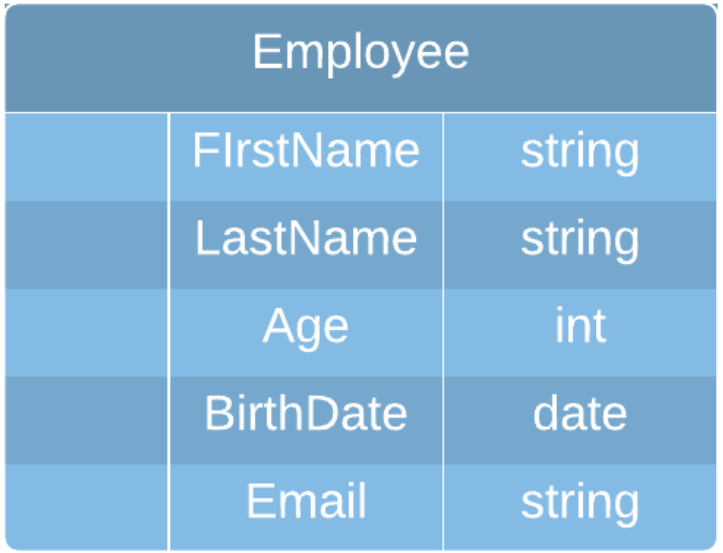
Entity Relationship Diagram

- **Entity**: It is an object that needs to have data stored. Data of people, companies, flights and etc.



Entity Relationship Diagram

- **Attributes** : It can be called column too. It is a characteristisc of the entity that needs it;
- It has a name and a type, like int, float, string, etc.



Entity Relationship Diagram

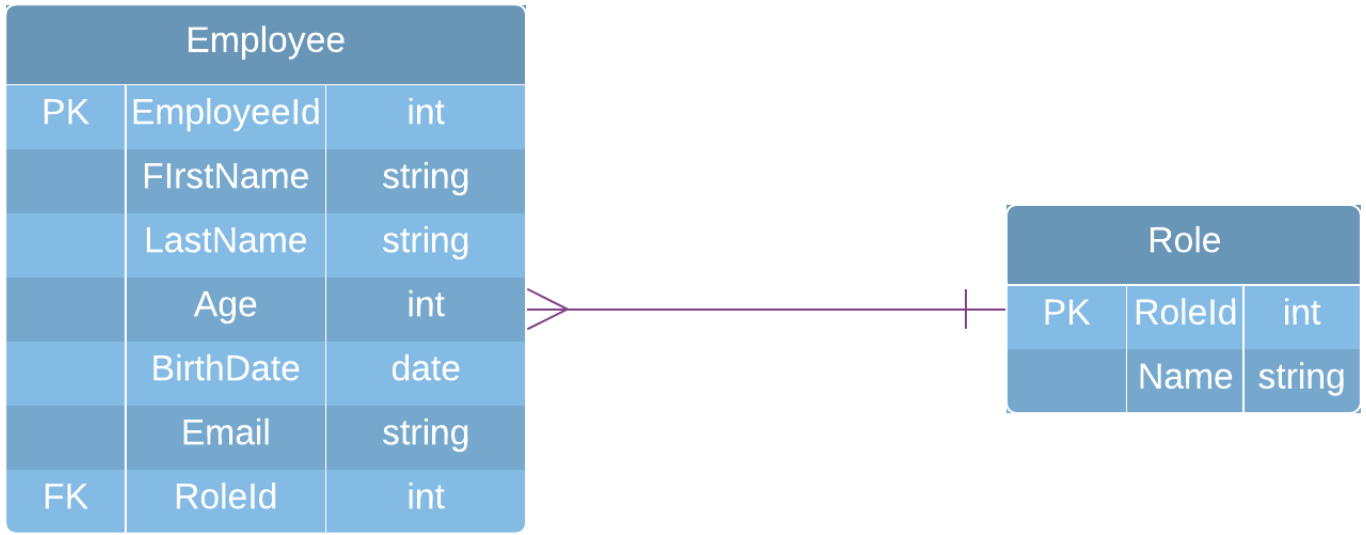
- **Primary Key** : Also known as PK, it is a special attribute that uniquely defines a record in a database table;

- A database can't have two or more equal values on this atribute.

Employee		
PK	EmployeeId	int
	FIrstName	string
	LastName	string
	Age	int
	BirthDate	date
	Email	string

Entity Relationship Diagram

- **Foreign Key** : Also known as FK, it references the primary key of any table;
- This table can be the same or another table.



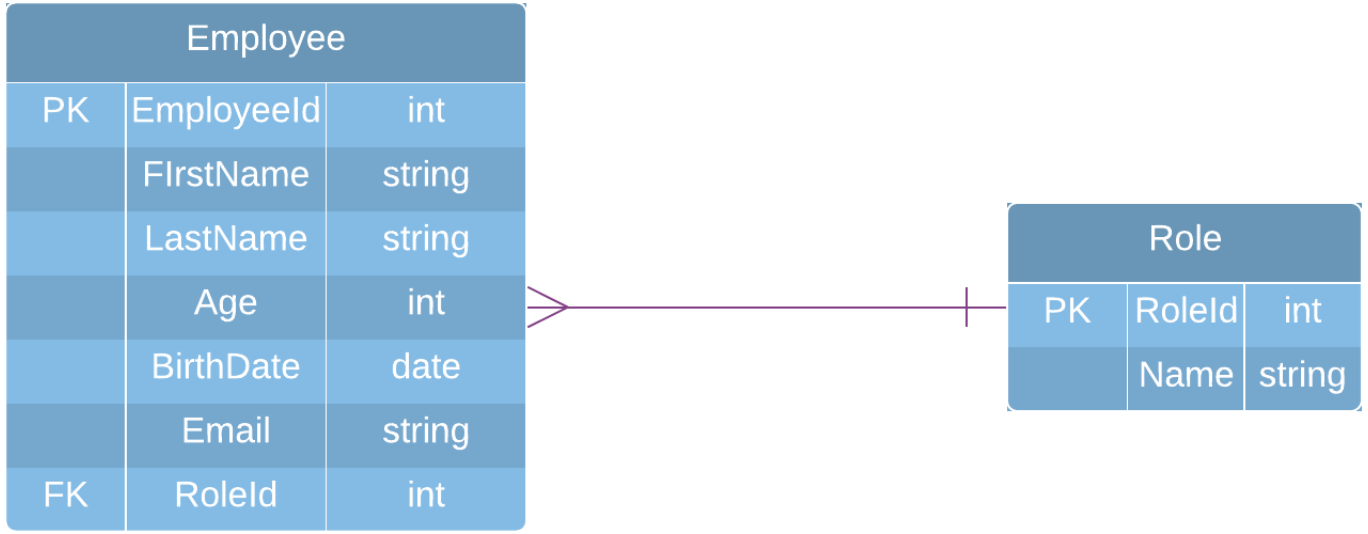
Entity Relationship Diagram

EmployeeId	FirstName	LastName	RoleId
1	John	Doe	1
2	Mary	Jayme	2
3	Abe	Johnson	1
4	Steve	Jobs	4

RoleId	Name
1	Programmer
2	HR
3	Assistant
4	CEO

Entity Relationship Diagram

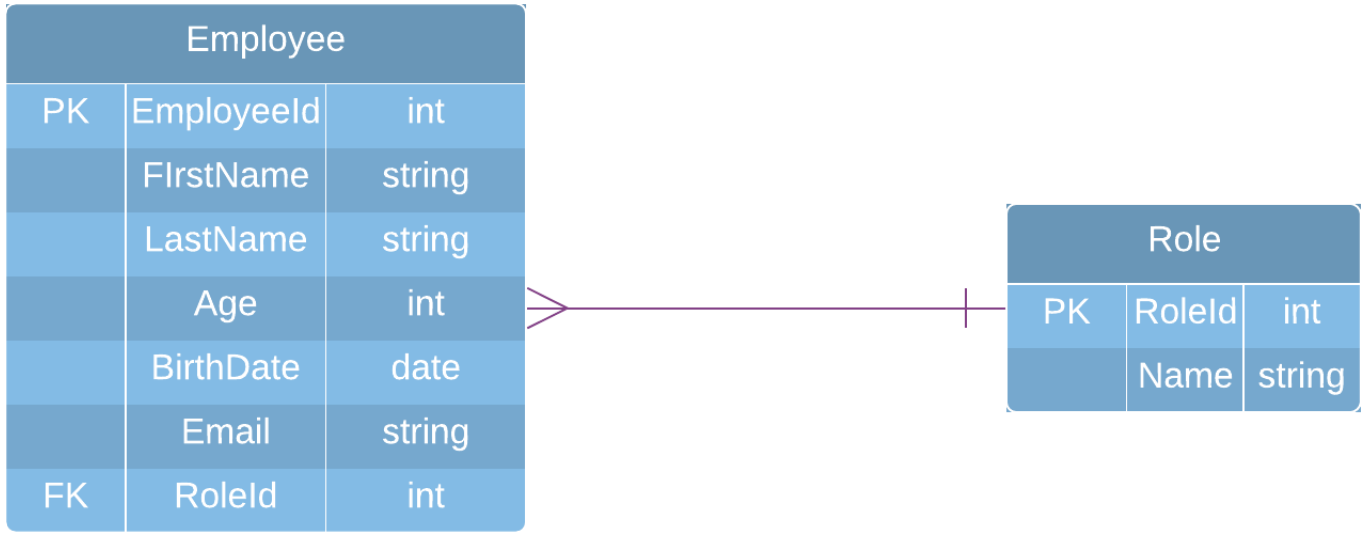
- **Relationship** : A relationship means there are entities associated with each other.



- In this example, Employee and Role are related between each other.

Entity Relationship Diagram

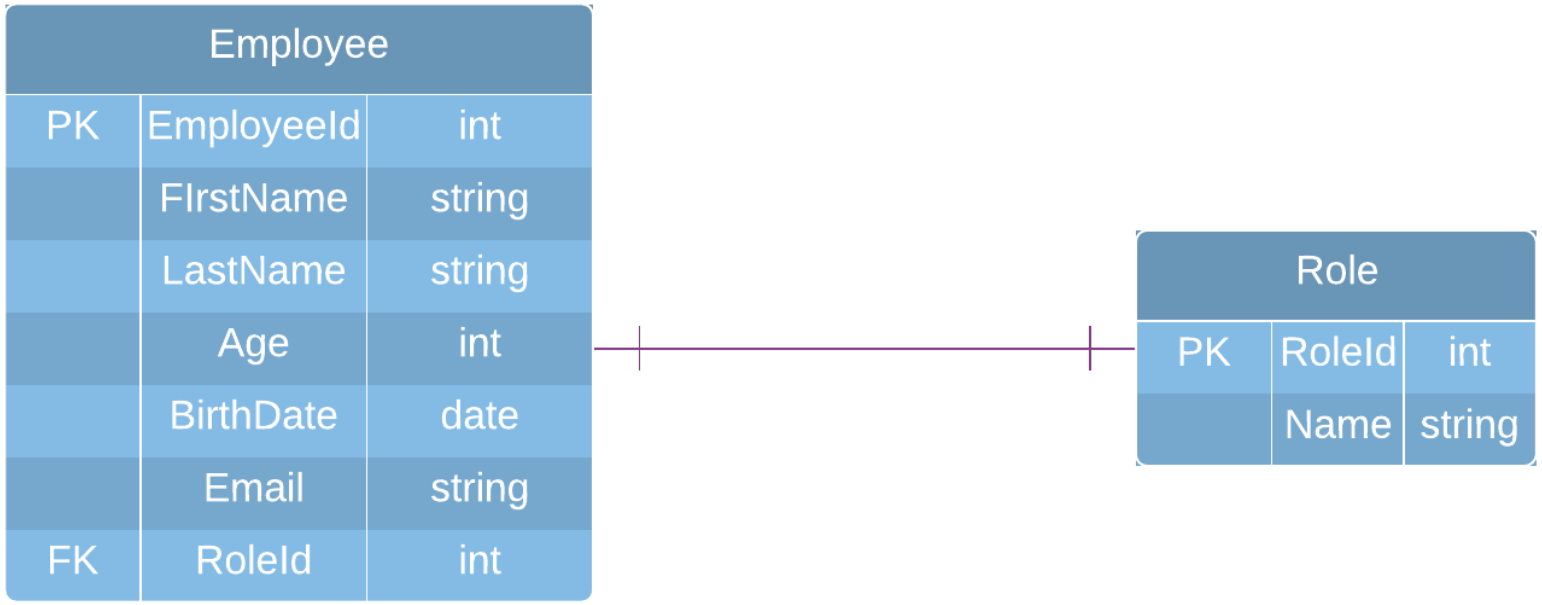
- **Cardinality** : Specifies the number of ocurrence between each entity.



- In this example, Employee is related with one Role and a Role is related with many Employees. It means that an Employee has only one Role, but a Role has many Employees related to it.

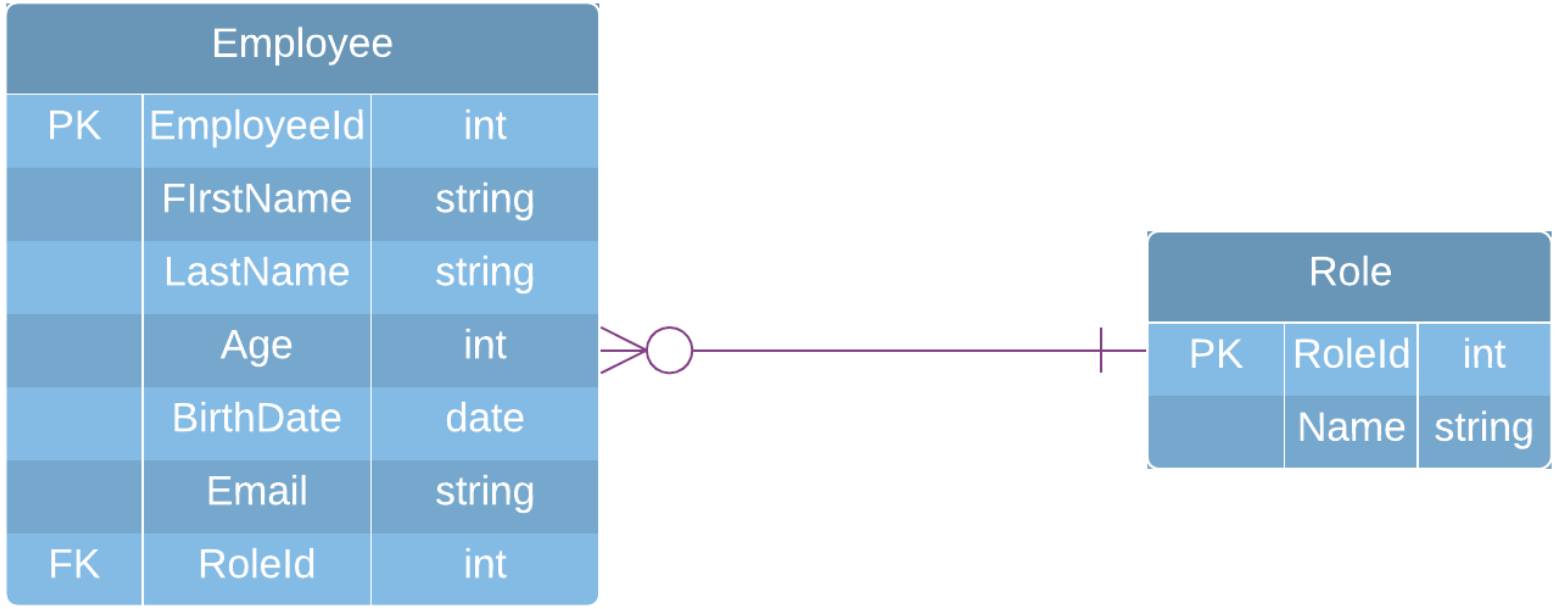
Entity Relationship Diagram

- One-to-One cardinality



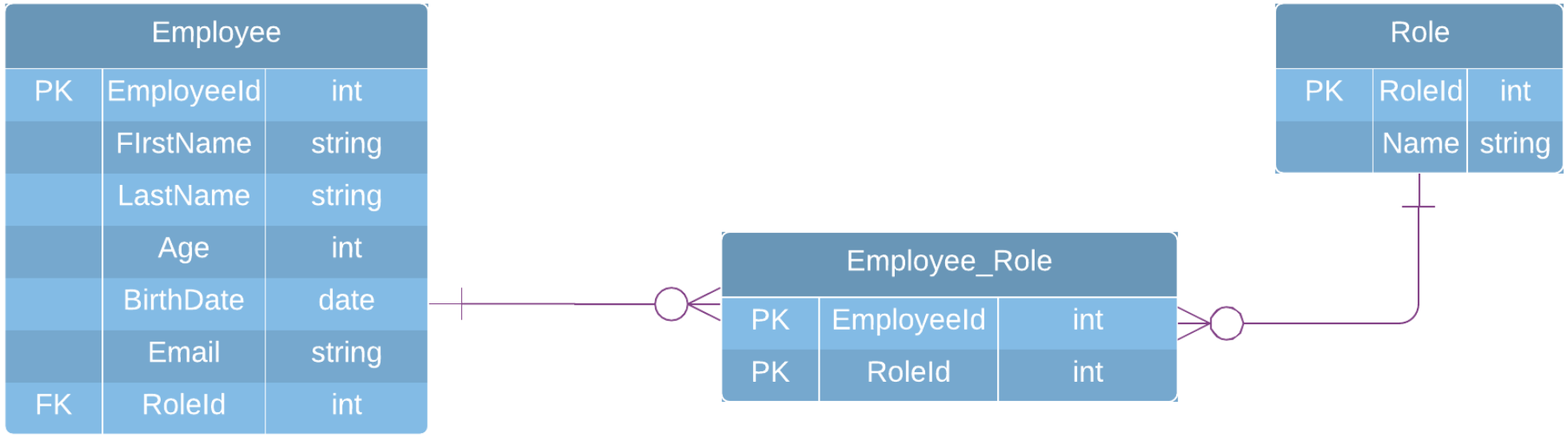
Entity Relationship Diagram

- One-to-Many cardinality



Entity Relationship Diagram

- Many-to-Many cardinality



Let's practice

- An organization needs a software to it's clients managing their finances. This software needs to allow registering expenses and earnings (budgets);
- They need to describe the form that they earned that budget on the system;
- Categories are necessary to group budgets and expenses;
- Clients want to filter their expenses by month and years grouped by categories;
- They also want to filter their budgets by month and years grouped by days. An user should be able to register and log in / out on the system without any help;
- They want to see the total of budgets and expenses and see if they have more budgets or more expenses;

The software's development