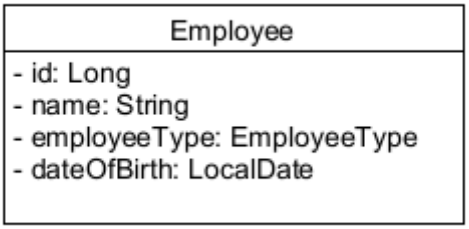
# JPA tábla generálás cheat sheet

Az alábbiakban azt foglaltam össze, hogy a JPA milyen módon menti le adatbázisba az entitásokat a különböző esetekben.

# Egy egyszerű entitás leképezése adatbázisba

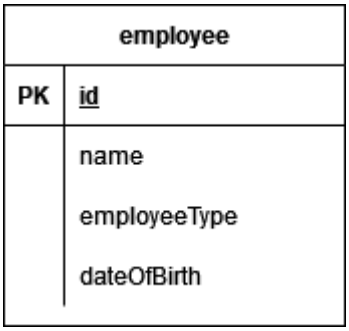
Az entitás UML-diagramja:




Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    private EmployeeType employeeType;

    private LocalDate dateOfBirth;
}
```

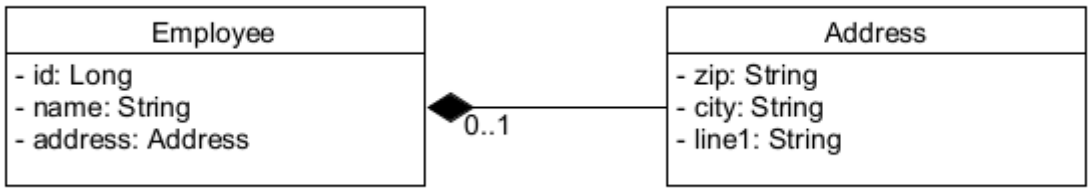
A JPA által alapértelmezetten generált tábla az adatbázisban:



# `@Embedded` annotációval ellátott attribútum esetén

Ekkor az attribútum maga nem kollekció, de nem is entitás. Az attribútum osztályán rajta kell lennie az `@Embeddable` annotációnak.

Az entitás és az általa tartalmazott osztály UML-diagramja:




Forráskód:

```java
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @Embedded
    private Address address;
}
```

```java
@Embeddable
public class Address {

    private String zip;

    private String city;

    private String line1;
}
```

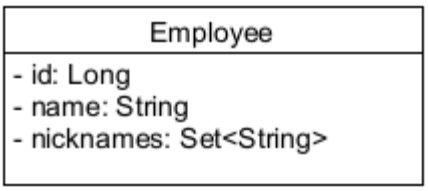
A JPA által alapértelmezetten generált tábla az adatbázisban:

| employee | |
|---|---|
| PK | id |
| | name |
| | zip |
| | city |
| | line1 |

# `@ElementCollection` annotációval ellátott, `List` vagy `Set` típusú attribútum esetén

Ekkor az entitásnak valamilyen `List` vagy `Set` típusú attribútuma van, de maguk a kollekció elemei nem entitások.

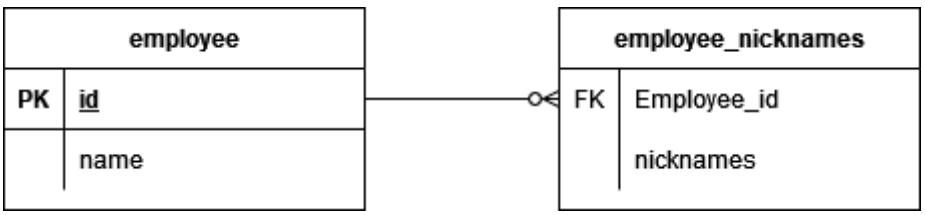
Az entitás UML-diagramja:




Forráskód

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @ElementCollection
    private Set<String> nicknames;
}
```

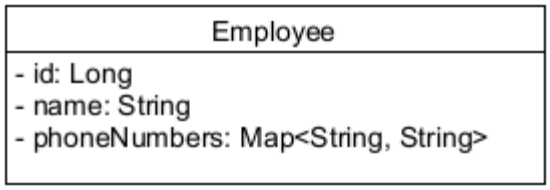
A JPA által alapértelmezetten generált táblák az adatbázisban:



# `@ElementCollection` annotációval ellátott, `Map` típusú attribútum esetén

Ekkor az entitásnak valamilyen `Map` típusú attribútuma van, de maguk a kollekció elemei nem entitások.

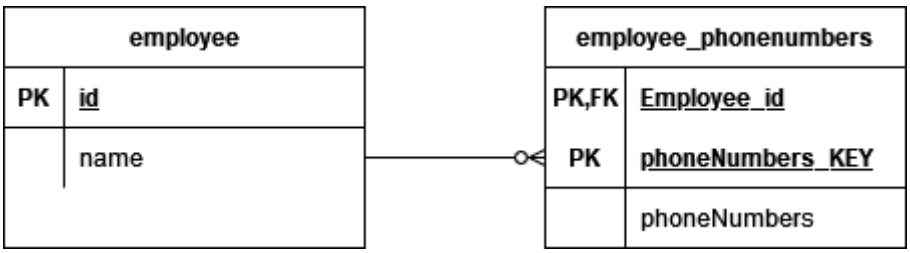
Az entitás UML-diagramja:




Forráskód

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @ElementCollection
    private Map<String, String> phoneNumbers;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:

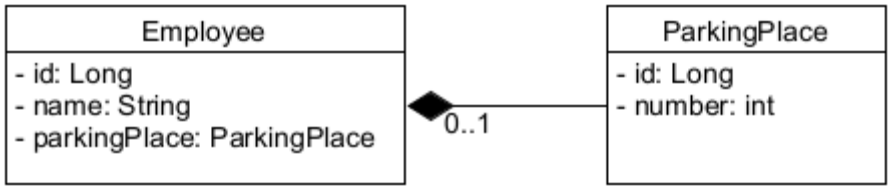


# Kapcsolatok esetén

Kapcsolatról akkor beszélünk, ha egy entitásnak egy másik entitás szerepel valamilyen módon az attribútumai között. A kapcsolatok lehetnek egyirányúak (amikor csak az egyik entitásnak van a másik entitás típusú vagy a másik entitást tartalmazó kollekció típusú attribútuma), illetve kétirányúak (amikor mindkét entitásnak van a másik entitás típusú vagy a másik entitást tartalmazó kollekció típusú attribútuma).

## Egyirányú egy-egy kapcsolat esetén

Az entitások UML-diagramja:

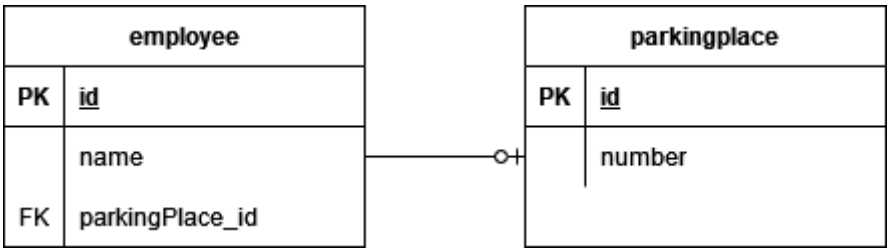



Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @OneToOne
    private ParkingPlace parkingPlace;
}
```

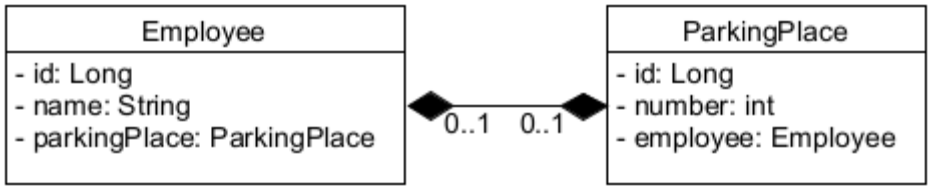
```
@Entity
public class ParkingPlace {

    @Id
    private Long id;

    private int number;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Kétirányú egy-egy kapcsolat esetén

Az entitások UML-diagramja:




A `@OneToOne` annotáció esetében megválasztható, hogy a kapcsolatnak melyik oldala legyen az inverse side (melyik oldalon legyen beállítva az annotáció `mappedBy` eleme), és ettől függ az is, hogy melyik táblába kerül a foreign key hivatkozás.

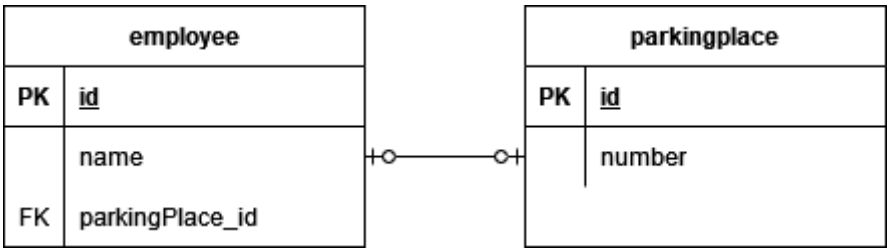
- Ha a kapcsolat inverse side-ja a `ParkingPlace` entitásban lévő kapcsolati attribútum:

Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @OneToOne
    private ParkingPlace parkingPlace;
}
```

```
@Entity
public class ParkingPlace {

    @Id
    private Long id;

    private int number;

    @OneToOne(mappedBy = "parkingPlace")
    private Employee employee;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



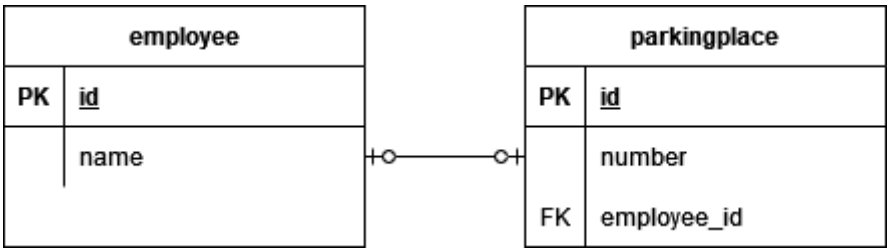

- Ha a kapcsolat inverse side-ja az `Employee` entitásban lévő kapcsolati attribútum:

Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @OneToOne(mappedBy = "employee")
    private ParkingPlace parkingPlace;
}
```

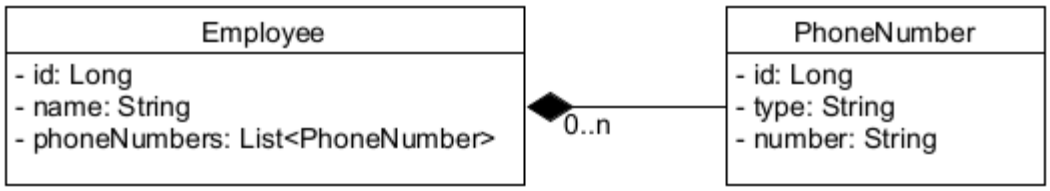
```
@Entity
public class ParkingPlace {

    @Id
    private Long id;

    private int number;

    @OneToOne
    private Employee employee;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Egyirányú egy-több kapcsolat esetén

Az entitások UML-diagramja:

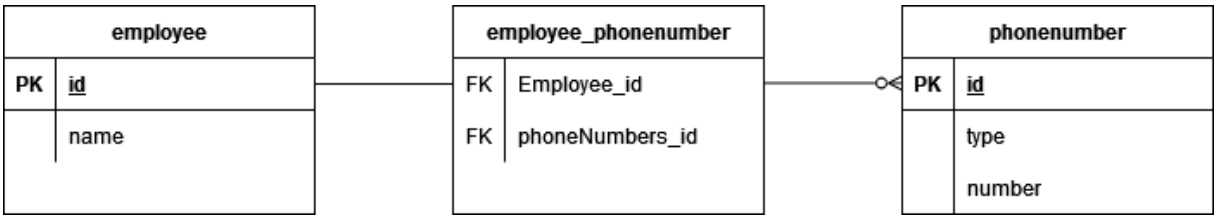



Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @OneToMany
    private List<PhoneNumber>
phoneNumbers;
}
```

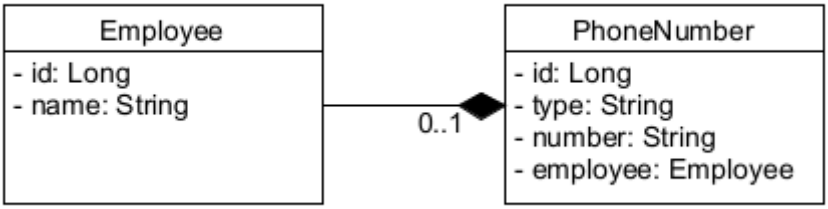
```
@Entity
public class PhoneNumber {

    @Id
    private Long id;

    private String type;

    private String number;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Egyirányú több-egy kapcsolat esetén

Az entitások UML-diagramja:

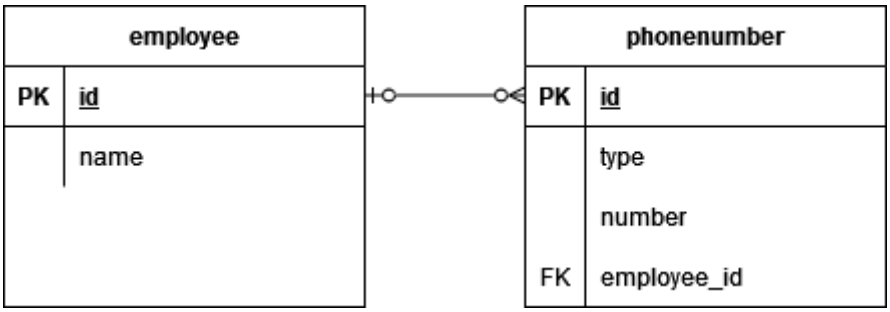



Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;
}
```

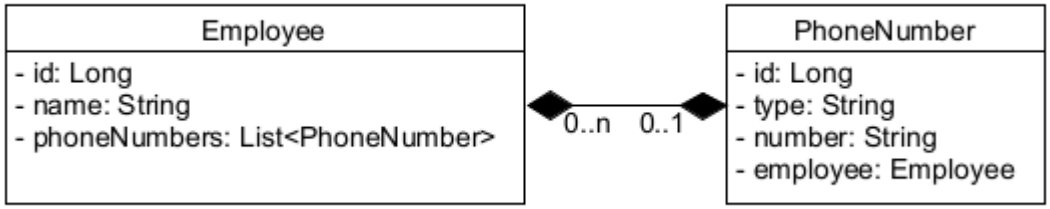
```
@Entity
public class PhoneNumber {

    @Id
    private Long id;

    private String type;

    private String number;

    @ManyToOne
    private Employee employee;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Kétirányú egy-több kapcsolat esetén

Az entitások UML-diagramja:

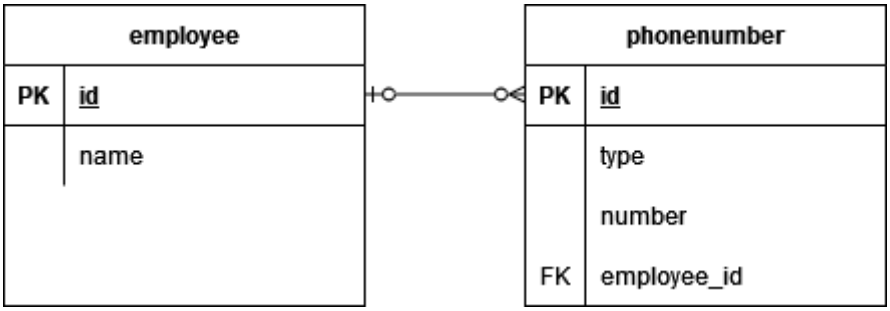



Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @OneToMany(mappedBy = "employee")
    private List<PhoneNumber>
phoneNumbers;
}
```

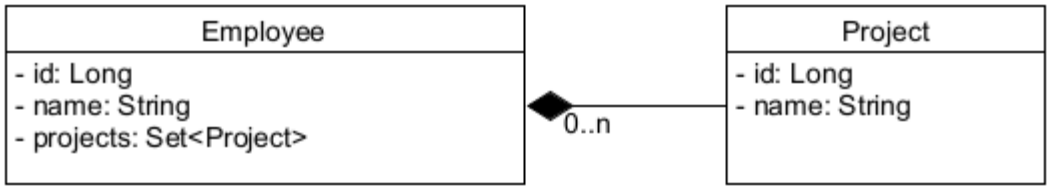
```
@Entity
public class PhoneNumber {

    @Id
    private Long id;

    private String type;

    private String number;

    @ManyToOne
    private Employee employee;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Egyirányú több-több kapcsolat esetén

Az entitások UML-diagramja:

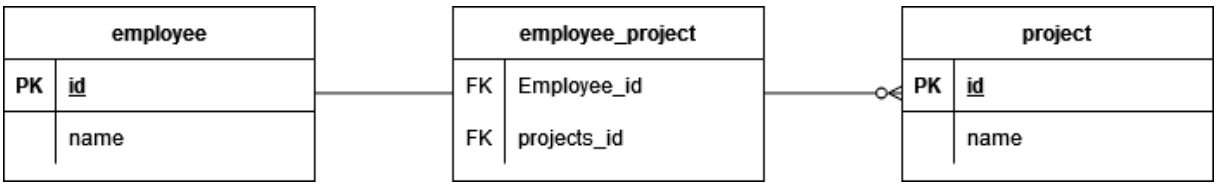



Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @ManyToMany
    private Set<Project> projects;
}
```

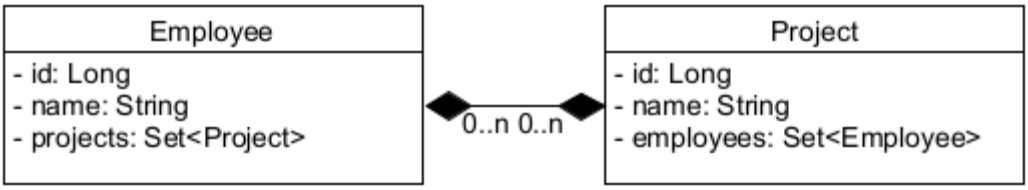
```
@Entity
public class Project {

    @Id
    private Long id;

    private String name;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Kétirányú több-több kapcsolat esetén

Az entitások UML-diagramja:




A `@ManyToMany` annotáció esetében is megválasztható, hogy a kapcsolatnak melyik oldala legyen az inverse side (melyik oldalon legyen beállítva az annotáció `mappedBy` eleme). Itt viszont bármelyik oldal is az inverse side, mindenképpen egy kapcsolótábla kerül legenerálásra, és a különbség egyedül a tábla nevében és oszlopainak sorrendjében mutatkozik meg.

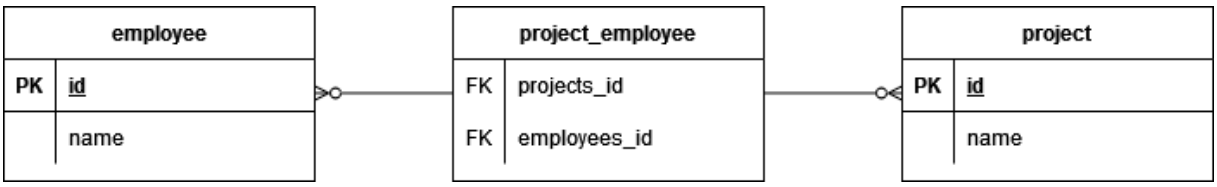
- Ha a kapcsolat inverse side-ja az `Employee` entitásban lévő kapcsolati attribútum:

Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @ManyToMany(mappedBy = "employees")
    private Set<Project> projects;
}
```

```
@Entity
public class Project {

    @Id
    private Long id;

    private String name;

    @ManyToMany
    private Set<Employee> employees;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



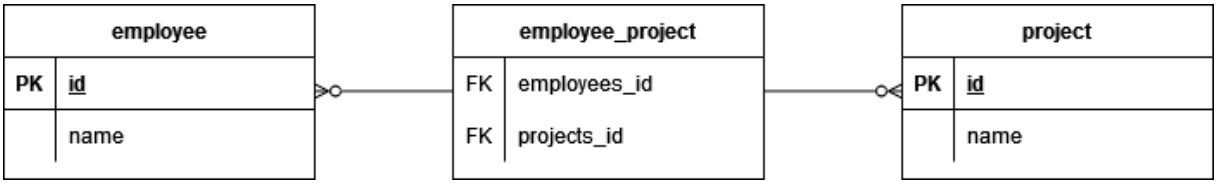

- Ha a kapcsolat inverse side-ja a `Project` entitásban lévő kapcsolati attribútum:

Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;

    @ManyToMany
    private Set<Project> projects;
}
```

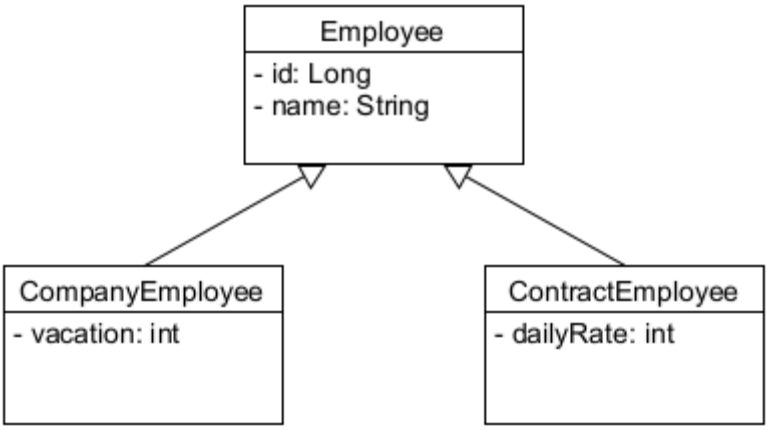
```
@Entity
public class Project {

    @Id
    private Long id;

    private String name;

    @ManyToMany(mappedBy = "projects")
    private Set<Employee> employees;
}
```

A JPA által alapértelmezetten generált táblák az adatbázisban:



# Öröklődés esetén

Az entitások UML-diagramja:




Öröklődés esetén háromféle lehetőségünk van az egymásból öröklődő osztályokat adatbázisba menteni. Ha a JPA alapértelmezett működését felül akarjuk bírálni, azt az ős entitás osztályra tett `@Inheritance` annotációval tehetjük meg.

Forráskód:

```
@Entity
public class Employee {

    @Id
    private Long id;

    private String name;
}
```

vagy

```
@Entity
@Inheritance(strategy =
InheritanceType.SINGLE_TA
BLE)
public class Employee {

    @Id
    private Long id;

    private String name;
}
```

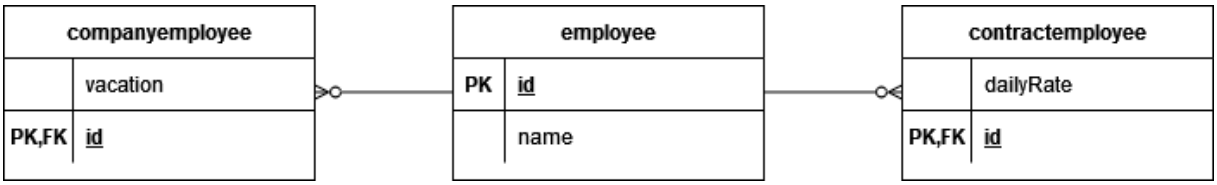
```
@Entity
public class CompanyEmployee extends
Employee {

    private int vacation;
}
```

```
@Entity
public class ContractEmployee extends
Employee {

    private int dailyRate;
}
```

A JPA által alapértelmezetten generált tábla az adatbázisban (ez megegyezik azzal, ha a tábla generálási stratégiát úgy adjuk meg, hogy `@Inheritance(strategy = InheritanceType.SINGLE_TABLE)`):

| employee | |
|---|---|
| | DTYPE |
| PK | id |
| | name |
| | vacation |
| | dailyRate |

A JPA által generált táblák az adatbázisban, ha a tábla generálási stratégiát úgy adjuk meg, hogy `@Inheritance(strategy = InheritanceType.JOINED)`:




A JPA által generált táblák az adatbázisban, ha a tábla generálási stratégiát úgy adjuk meg, hogy `@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)`:

| employee | |
|---|---|
| PK | id |
| | name |

| companyemployee | |
|---|---|
| PK | id |
| | name |
| | vacation |

| contractemployee | |
|---|---|
| PK | id |
| | name |
| | dailyRate |