BEZIEHUNGEN

Many to One, Many to Many, One to One

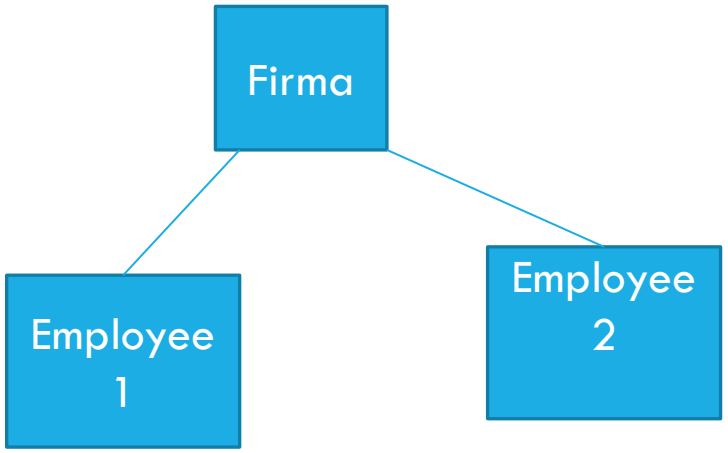
ONE TO MANY BEZIEHUNG (FOREIGN KEY)

```
from django.db import models


# Eine Firma viele Angestellte aber ein Angestellter nur eine Firma

class Company(models.Model):

    name = models.CharField(max_length=100)


class Employee(models.Model):

    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    date_of_entry = models.DateField()
    company = models.ForeignKey(Company,
on_delete=models.CASCADE, related_name='employees')
```

A) ERKLÄRUNG

Zu einer Company kann es mehrere Angestellte geben, und umgekehrt arbeitet jeder Angestellter in einer Firma.

Mit dem Feldtyp ForeignKey (dt. Fremdschlüssel) wird eine One-To-Many Relation (dt. Eins-zu-viele-Relation) festgelegt, die besagt, dass es zu einem Angestellten genau eine Firma gibt. Das on_delete - Argument legt fest, was mit den Angestellten passieren soll, wenn die Firma gelöscht wird, im Falle von Cascade werden diese einfach gelöscht. Der related_name gibt an, wie der Zugriff auf die Angestellten via dem Firmen-Objekt heissen soll.



B) BEISPIEL: COMPANY / EMPLOYEE

```
comp = Company.objects.get(pk=1)

a = Employee(date_of_entry=date(2009,3,3), first_name='James', last_name='Bond')

b = Employee(date_of_entry=date(2019,1,12), first_name='Bob', last_name='Sponge')


a.company = comp

a.save()

b.company = comp

b.save()

print(comp.employees) # Zugriff auf Employees via related_name
```

```
James Bond
Bob Sponge
```

C) BEISPIEL COMPANY / EMPLOYEE

über den Related Name (employees) auf den Related Manager zugreifen

```
company = Company.objects.get(pk=1)
```

```
company.employees
```

```
<django.db.models.fields.related_descriptors.create_reverse_many_to_one_manager.<locals>.RelatedManager object at 0x7f2fe829bf10>
```

Der Related Manager ist auch ein Manager (siehe Manager) und besitzt alle Methoden des Managers.

```
company.employees.all()
```

```
company.employees.get(pk=1)
```

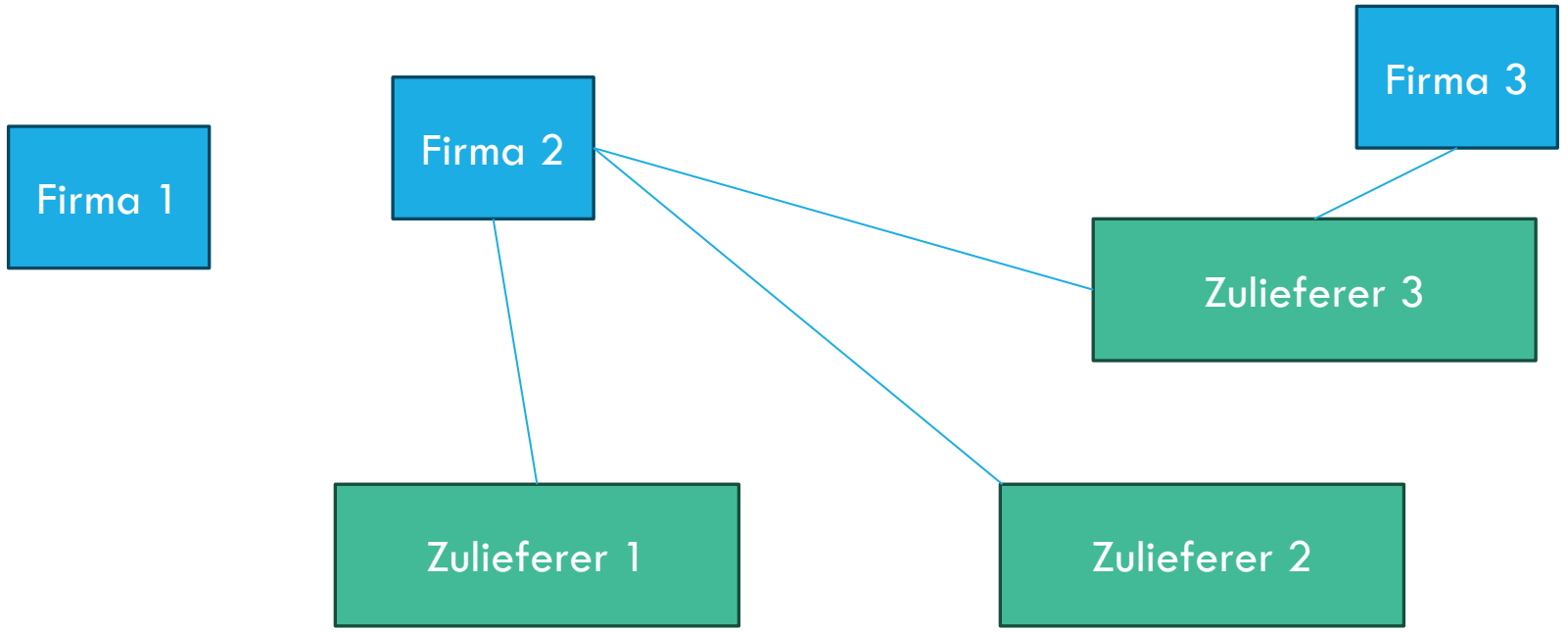
```
company.employees.order_by('-entry_date')
```

MANY TO MANY BEZIEHUNG

```
class Supplier(models.Model):
    name = models.CharField(max_length=50)


# Eine Firma, viele Zulieferer. Ein Zulieferer für viele Firmen
class Company(models.Model):
    name = models.CharField(max_length=50)
    supplier = models.ManyToManyField(Supplier, related_name="companies")
```

ERKLÄRUNG

Jede Firma hat mehrere Zulieferer (Supplier) und jeder Zulieferer arbeitet für mehrere Firmen. Mit dem Feldtyp ManyToManyField wird eine Many-To-Many Relation (dt. Viele-zu-vielen-Relation) festgelegt, die besagt, dass es zu einer Firma viele Zulieferer und zu einem Zulieferer viele Firmen gibt.



BEISPIEL: COMPANY / SUPPLIER

```
company1 = Company.objects.create(name='Company 1')
company2 = Company.objects.create(name='Company 2')

# Lieferanten erstellen
supplier1 = Supplier.objects.create(name='Supplier 1')
supplier2 = Supplier.objects.create(name='Supplier 2')

# Zuweisen des Lieferanten zu den Firmen
supplier1.companies.add(company1, company2)
supplier2.companies.add(company1)
```

ONE TO ONE BEZIEHUNG

```
class EmployeeProfile(models.Model):
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    adress = models.CharField(max_length=50)

class Employee(models.Model):
    name = models.CharField(max_length=50)
    age = models.IntegerField()
    profile = models.OneToOneField(EmployeeProfile)
```