Django Relationships

A relationship is a connection between two data entities. These relationships help represent real-world associations like:

- A person has one passport.
- A teacher teaches many students.
- A student can enroll in many courses, and each course can have many students.

Types of Relationships

There are three main types of relationships: One-to-One, One-to-Many and Many-to-Many

1. One-to-One (1:1)

One record in a table is associated with exactly one record in another table.

Example:

- One person has one passport.
- One user has one profile.

```
class User(models.Model):
    name = models.CharField(max_length=100)

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField()
```

Two entities have a strict 1:1 mapping.

You want to split data into related tables for organizational or performance reasons.

2. One-to-Many (1:N)

One record in a table is associated with multiple records in another table.

Example:

- One author writes many books.
- One department has many employees.

```
class Author(models.Model):
    name = models.CharField(max_length=100)

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
```

A parent object can have multiple child objects, but each child belongs to only one parent.

3. Many-to-Many (M:N)

Multiple records in one table are associated with multiple records in another table.

Example:

- Students enroll in many courses, and each course has many students.
- Movies have many actors, and actors act in many movies.

```
class Student(models.Model):
    name = models.CharField(max_length=100)

class Course(models.Model):
    title = models.CharField(max_length=100)
    students = models.ManyToManyField(Student)
```

You need a bidirectional and multi-linked relationship.

Django will create an intermediate table behind the scenes to store the links.