**Videofile:74**

**Codefile:ch64**

**Django Model One to One Relationship**

## Django Model One-to-One Relationship

* A **One-to-One relationship** is like a "special case" of ForeignKey where **one model is linked to exactly one other model**.
* It ensures **uniqueness**: each record in one table corresponds to only one record in another table.
* Django provides this through the field:
* models.OneToOneField(OtherModel, on\_delete=models.CASCADE)

### ✅ When to Use?

* When you want to **extend an existing model** without modifying it.
* Example:
  + A User can have **only one Profile**.
  + A Student can have **only one AdmitCard**.

### 🔑 Key Points:

1. **Uniqueness enforced** → no two objects can reference the same relation.
2. **Access**:
   * student.profile → forward relation.
   * profile.student → reverse relation.
3. on\_delete is required (commonly CASCADE).

👉 Example for notes:

from django.db import models

class Student(models.Model):

    name = models.CharField(max\_length=100)

    roll = models.IntegerField()

class Profile(models.Model):

    student = models.OneToOneField(Student, on\_delete=models.CASCADE)

    address = models.CharField(max\_length=255)

    phone = models.CharField(max\_length=20)

# Usage:

# s = Student.objects.create(name="Ali", roll=101)

# p = Profile.objects.create(student=s, address="Lahore", phone="03001234567")

# print(s.profile.address)   # Access profile via student

# print(p.student.name)      # Access student via profile

⚡ **Summary for Notes:**  
One-to-One relationship in Django links one model to exactly one other model. It’s useful for extending models (like User → Profile) and ensures uniqueness between related records.

Self notes:

We can work with one to one relationship between user and our table student

In this we create a model having student and import user table

And in student we can set onetoone relation with the user :

And do makemigrationsand migrate:

Models.py:

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

    user = models.OneToOneField(User,on\_delete=models.CASCADE) #here we also pass primary key

    name= models.CharField(max\_length=255)

    city= models.CharField(max\_length=255)

By using cascade:

1,If we delete the User ,, the Profile is automatically delete’s that associate or rleation between of that certain user

2,If we go into profile and delete’s so we see that profile delete and user cannot be delete its exist

Shortly:

If want to delete the user profile also deletes with that

If want to delete the profile user cannot be delete

By using PROTECT:

Models.py:

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

    user = models.OneToOneField(User,on\_delete=models.PROTECT)

    name= models.CharField(max\_length=255)

    city= models.CharField(max\_length=255)

1. By using protect if we make the user and than create the profile ,, and than we want to delete that cannot be delete
2. If we first delete profile than go into the user and try to delete now its easily delete

By using cascade (limit):

By using this we check who can create the proflie:

Model.py:

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

    user = models.OneToOneField(User,on\_delete=models.CASCADE,limit\_choices\_to={'is\_staff':True})

    name= models.CharField(max\_length=255)

    city= models.CharField(max\_length=255)

1. IN THIS WE CAN LIMIT ONLY IS\_STAFF CAN ALLOW TO
2. ONLY USERS THAT ARE IS\_STAFF allows to add their profiles

By using do noting:

We cannot use this in one to one table because its shows constriants error

But we can use that if we completely check all the integrity

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

    user = models.OneToOneField(User,on\_delete=models.DO\_NOTHING)

    name= models.CharField(max\_length=255)

    city= models.CharField(max\_length=255)

both can go independent and we can delete user and

also independenly delet’s the profile

ONE TO ONE RELATIONSHIP:

MODELS.PY:

LIKE USER WITH PAGE AND PAGE WITH LIKES :

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

    # user = models.OneToOneField(User,on\_delete=models.CASCADE) #here we also pass primary key

    # user = models.OneToOneField(User,on\_delete=models.PROTECT)

    # user = models.OneToOneField(User,on\_delete=models.CASCADE,limit\_choices\_to={'is\_staff':True})

    user = models.OneToOneField(User,on\_delete=models.DO\_NOTHING)

    name= models.CharField(max\_length=255)

    city= models.CharField(max\_length=255)

class Page(models.Model):

    profile = models.OneToOneField(User,on\_delete=models.CASCADE)

    page\_name = models.CharField(max\_length=255)

class Like(Page):

    page = models.OneToOneField(Page, on\_delete=models.CASCADE,parent\_link=True)

    likes = models.IntegerField()