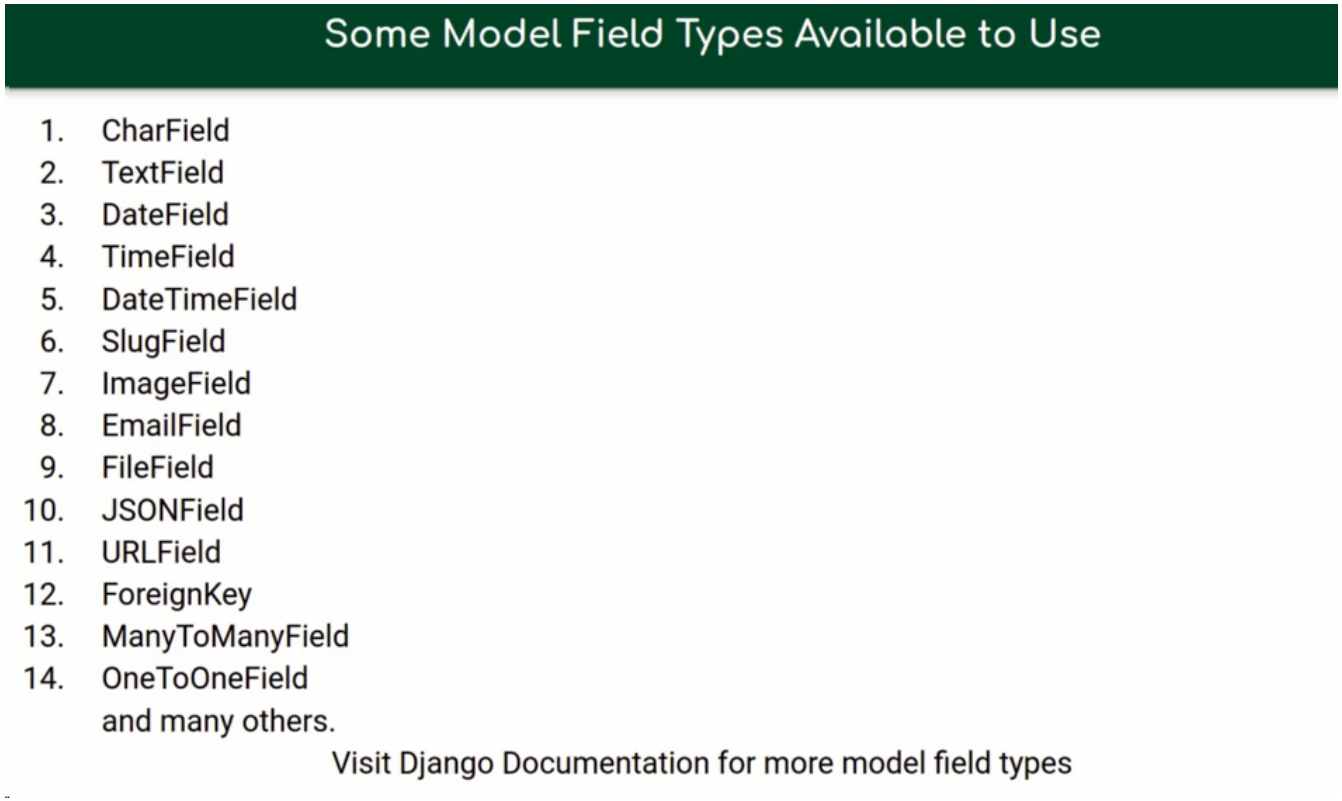
charfield : used for fixed length of character

textfield : used for storing variable length of character

1. CharField
2. TextField
3. DateField
4. TimeField
5. DateTimeField
6. SlugField
7. ImageField
8. EmailField
9. FileField
10. JSONField
11. URLField
12. ForeignKey
13. ManyToManyField
14. OneToOneField

and many others.



Some Model Field Options Available to Use

1. null
2. blank
3. choices
4. default
5. editable
6. error_messages
7. help_text
8. primary_key
9. unique
10. verbose_name

and many more

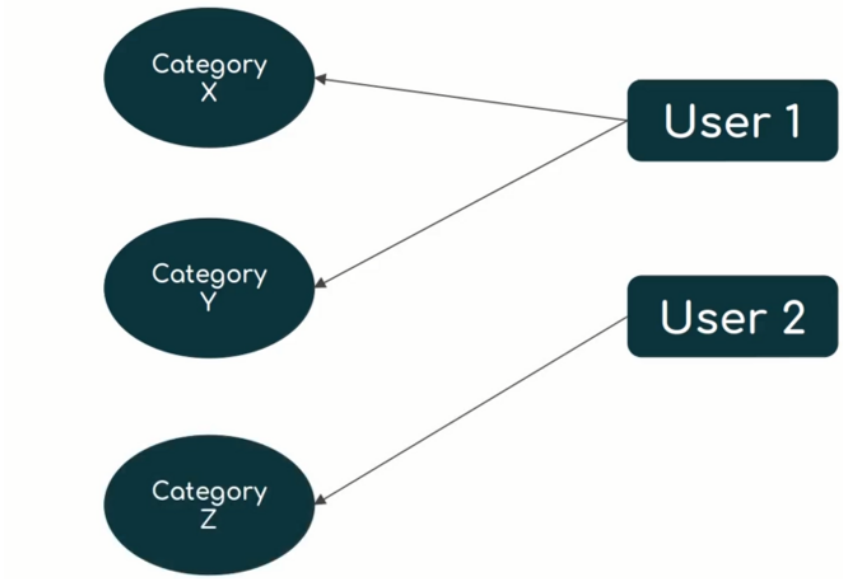
Visit Official Documentation for more model field options

null = True in the database we can have null values

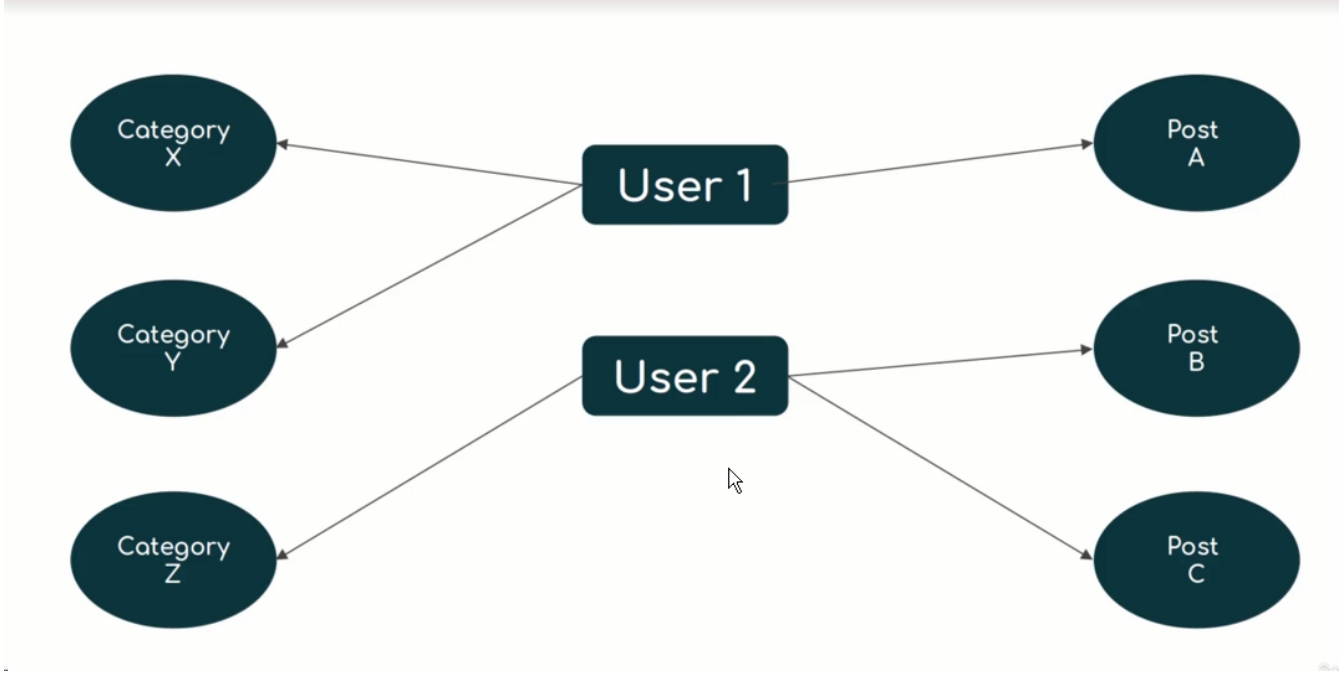
blank = True when a user dosent enter any value in the field dont raise any error

auto_now True update
auto_now_add= True create

Many to One Relationships



Many to One Relationships



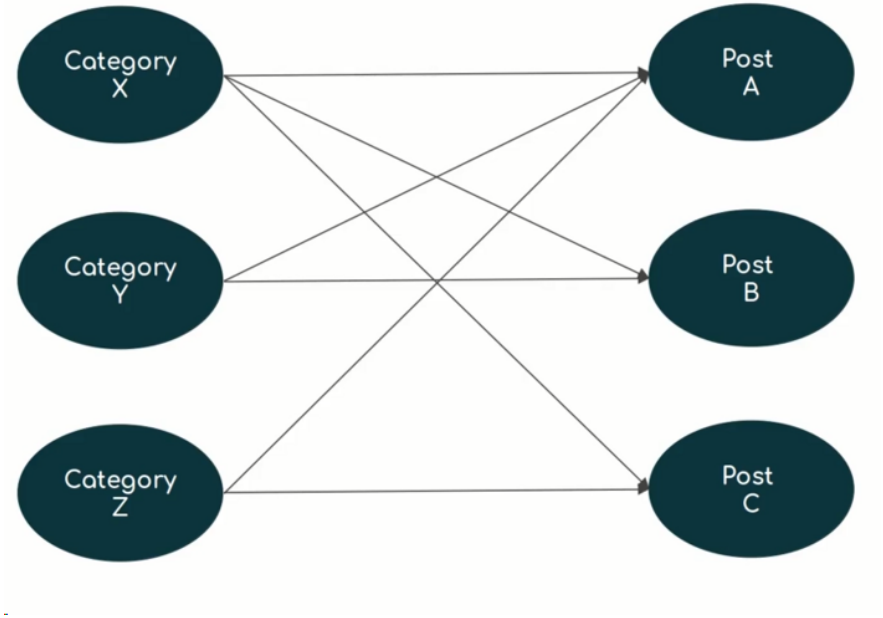
one user can create multiple post

one post can be created by one user

created_by=models.Foreignkey (User,on_delete=models.Cascade)

when the user is deleted it also delete corresponding objects

Many to Many Relationships



one category can have multiple post
one post can belongs to many category

Primary Key & Verbose Field Name

- By default, Django gives each model an auto-incrementing primary key with the type specified per app in AppConfig.default_auto_field or globally in the DEFAULT_AUTO_FIELD setting.
- For example: id = models.BigAutoField(primary_key=True)
- If you'd like to specify a custom primary key, specify primary_key=True on one of your fields.
- If Django sees you've explicitly set Field.primary_key, it won't add the automatic id column.
- Each model requires exactly one field to have primary_key=True (either explicitly declared or automatically added).

name= models.Charfield(verbose name="Category name")

override predefine models methods

Introduction to Meta Options in Django

- Model metadata is "anything that's not a field", such as ordering options (ordering), database table name (db_table), or human-readable singular and plural names (verbose_name and verbose_name_plural).
- None are required, and adding class Meta to a model is completely optional.

Some Available Meta Options To Use

1. abstract
2. verbose_name
3. db_table
4. order_with_respect_to
5. ordering
6. permissions
7. indexes
8. unique_together
9. index_together

And many more

Visit Official Django Documentation to Know More

```
class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    body = models.TextField(max_length=500, blank=True, null=True)
    commented_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
    commented_by = models.CharField(max_length=50)
```

when i delete the post all the corresponding comments also be deleted

Model Manager

- A Manager is the interface through which database query operations are provided to Django models.
- At least one Manager exists for every model in a Django application.
- Django adds a Manager with the name **objects** to every Django model class.
- We can also create our own model manager or modify the functionality of existing model manager.

lazy loading

Many to One Relationship:
following relationship forward
following relationship backward

many to many relationmship:
following relationship forward
p1
<Post: My Educational Post 1>
p1.category.all()
following relationship backward

```
for category in Category.objects.all():
    print(category.id)
```

3
4
Category.objects.get(id=3)
<Category: Educational>
c3=Category.objects.get(id=3)
c3
<Category: Educational>
c3.post_set.all()
<QuerySet [<Post: My Educational Post 1>]>

f expression

```
Post.objects.filter(last_updated_on__gt=F("published_on") + timedelta(seconds=1))
<QuerySet [<Post: My Educational Post 1>]>
```