CSE-465

Web Programming

Django ORM

Setting up the DB Model

- At first run the following command for initial migration:

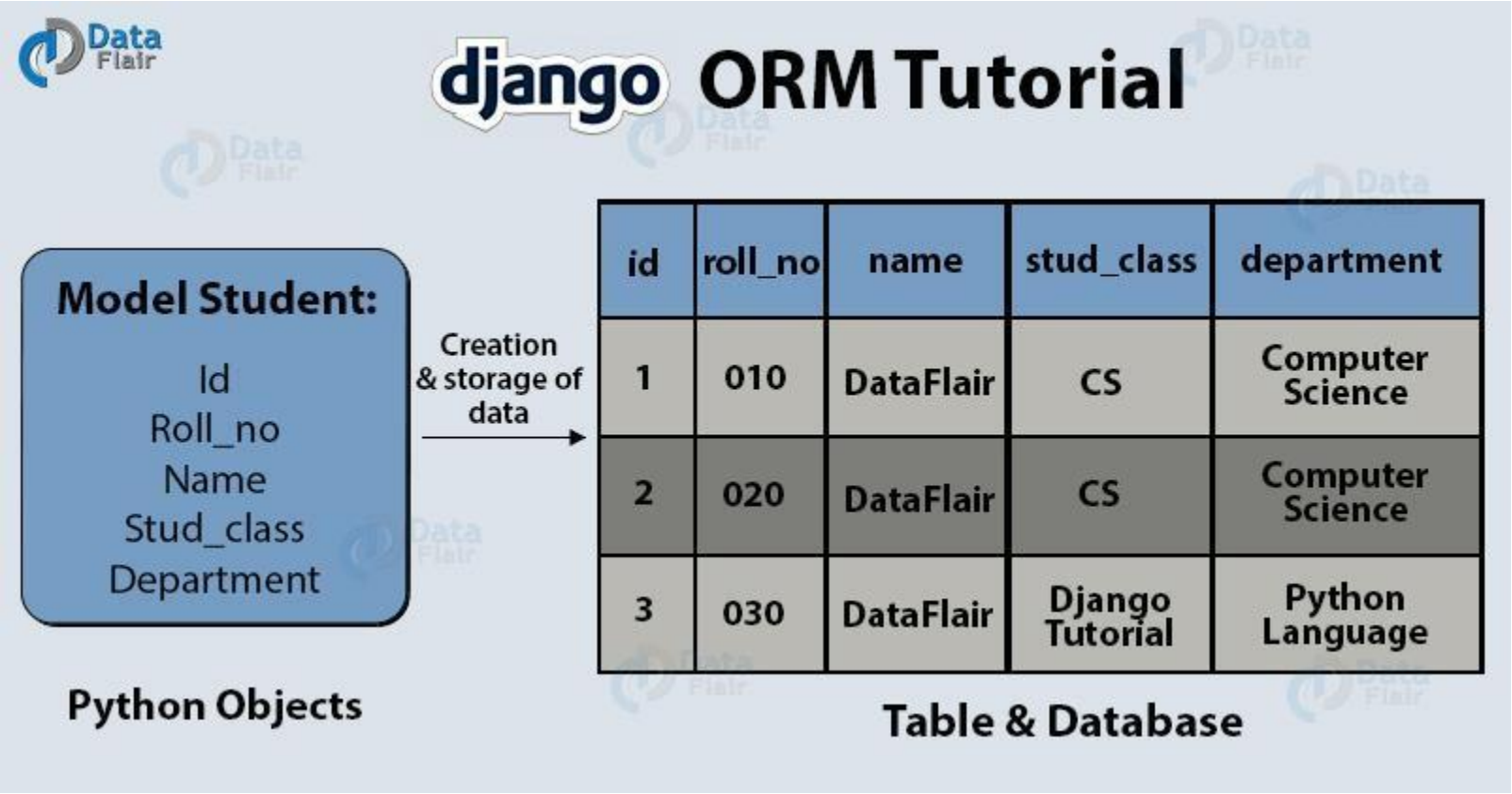
  python manage.py makemigrations <<project_name>>

  python manage.py migrate

- To create superuser (admin)

  python manage.py createsuperuser

Django Object Relation Mapper



An ORM will automatically create and store your object data in the database. You don’t have to write any SQL for the same.

Post Model

blog > models.py

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User

# Create your models here.
class Post(models.Model):
        title = models.CharField(max_length=100)
        content = models.TextField()
        date_posted = models.DateTimeField(default=timezone.now)

        #auto_now=True or auto_now_add=True
        author = models.ForeignKey(User, on_delete=models.CASCADE)
```

Django Model Fields

Field Name	Description
AutoField	It An IntegerField that automatically increments.
BigAutoField	It is a 64-bit integer, much like an AutoField except that it is guaranteed to fit numbers from 1 to 9223372036854775807.
BigIntegerField	It is a 64-bit integer, much like an IntegerField except that it is guaranteed to fit numbers from -9223372036854775808 to 9223372036854775807.
BinaryField	A field to store raw binary data.

Read more:

- https://docs.djangoproject.com/en/3.1/ref/models/fields/
- https://www.geeksforgeeks.org/django-model-data-types-and-fields-list/

Migrating into DB

Commands to save the model in database:

- python manage.py makemigrations

  (Optional) To check the SQL query :

  python manage.py sqlmigrate <appname> <migrate_no>

  [Note: The migration number will be inside the **migrations** folder of your app]

- python manage.py migrate

Migrations allow us to make changes into our database even if there is data in the table without worrying about the internal structure.

Register Model in Admin panel

blog > admin.py

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Methods in a Django model

Note: To access django from shell, use the command-

python manage.py shell

Suppose we are using User model, since we don't have anything in Post model.

- To get all entries:

  User.objects.all()

- To select an entry:
  - u1 = User.objects.first()
  - u1 = User.objects.filter(username='onix.hoque').first()
  - u1 = User.objects.get(id=1)
- To create a Post:

  p1 = Post(title='Post1', content='First post content',

  author=u1) p1.save()

Django ORM Shortcuts

Django ORM provides a lot of shortcut methods.

- To create a Post using user id:

  p1 = Post(title='Post1', content='First post content', author_id=u1.id)

  p1.save()

- To see all the post a user (u1) has created, use u1.<table_name>_set

  u1.post_set.all()

  - To create a post directly from user:

    u1.post_set.create(title='Post3', content='Third Post')

Linking the Post model to View

blog > views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Post

def home(request):
        context = {
                'posts': Post.objects.all()
        }
        return render(request, 'blog/home.html', context)

def about(request):
        return render(request, 'blog/about.html', {'title' :
'CSE-456 Project About page'})
```

Reference

Python Django Tutorial (Corey Schafer)

https://www.youtube.com/playlist?list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p