**Outline**

1. ORM = Object Relational Mapping
2. Model
3. **What is ORM?**

Object-Relational Mapper is a programming technique that helps application to interact with database such as SQLite, MySQL, PostgreSQL, Oracle.

• create a database schema from defined classes or models.

• generate SQL from Python code for a particular database which

means developer do not need to write SQL Code.

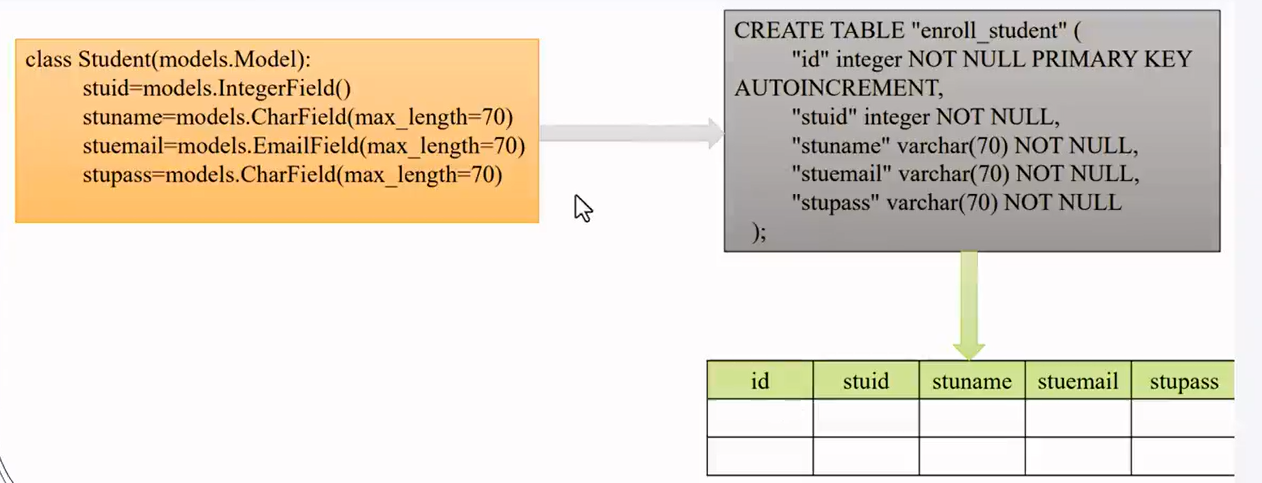
• helps to change the database easily

to connect databases with a web application.

• use connectors

Banglai:

|  |
| --- |
| ***📒 ORM (Object Relational Mapping) – Note***  ***🔹 ORM কী?***   * ***ORM = Object Relational Mapping*** * ***ডাটাবেজের টেবিল ↔ Python ক্লাস/অবজেক্ট এর মধ্যে সংযোগ।*** * ***SQL না লিখেই Python কোড দিয়ে CRUD করা যায়।***   ***🔹 কেন ORM ব্যবহার করব?***   * ***❌ Raw SQL না লিখেও কাজ করা যায়।*** * ***✅ কোড সহজ ও readable হয়।*** * ***✅ Database independent (MySQL, PostgreSQL, SQLite সাপোর্ট করে)।*** * ***✅ SQL Injection কমায় → নিরাপদ।*** * ***✅ Maintain ও Update করা সহজ।***   ***🔹 Django ORM (Models.py)***  from django.db import models    class Student(models.Model):  name = models.CharField(max\_length=100)  roll = models.IntegerField()  marks = models.FloatField()    ***👉 ORM এই ক্লাস থেকে SQL টেবিল বানাবে:***  CREATE TABLE Student (  id INTEGER PRIMARY KEY AUTOINCREMENT,  name VARCHAR(100),  roll INTEGER,  marks FLOAT  );    ***🔹 ORM Operations (CRUD)***  ***1. Create (INSERT)***  student = Student(name="Farhan", roll=1, marks=95.5)  student.save()    ***2. Read (SELECT)***  students = Student.objects.all() *# সব ডাটা*  student = Student.objects.get(roll=1) *# নির্দিষ্ট ডাটা*  top\_students = Student.objects.filter(marks\_\_gt=90) *# শর্ত দিয়ে ডাটা*    ***3. Update (UPDATE)***  student = Student.objects.get(roll=1)  student.marks = 99  student.save()    ***4. Delete (DELETE)***  student = Student.objects.get(roll=1)  student.delete()    ***🔹 সুবিধা***   * ***SQL ছাড়াই Database এ কাজ করা যায়।*** * ***Database পরিবর্তন হলেও কোডে বেশি পরিবর্তন লাগে না।*** * ***কোড clean ও সহজবোধ্য হয়।*** * ***Security বেশি।***   ***👉 Short Line: ORM = Database টেবিল কে Python class দিয়ে কন্ট্রোল করার টেকনিক।*** |



**QuerySet**

|  |
| --- |
| ***📒 QuerySet – Note***  ***🔹 QuerySet কী?***   * ***QuerySet হলো ডাটাবেজ থেকে ডাটা আনার, ফিল্টার করার, আপডেট করার বা ডিলিট করার জন্য Django ORM এর মাধ্যমে তৈরি হওয়া object-এর collection।*** * ***এক কথায়: ডাটাবেজ query এর result = QuerySet***   ***🔹 QuerySet কিভাবে কাজ করে?***  ***যখন তুমি Model.objects ব্যবহার করো → ORM ভিতরে ভিতরে SQL query চালায় এবং তার result কে QuerySet object আকারে ফেরত দেয়।***  ***Example:***  students = Student.objects.all()    ***👉 এখানে students হলো একটি QuerySet → যেখানে সব student data আছে।***  ***Internally ORM SQL বানাচ্ছে:***  SELECT \* FROM Student;    ***🔹 সাধারণ QuerySet Methods***  ***1. সব ডাটা আনা***  Student.objects.all()    ***2. নির্দিষ্ট ডাটা আনা***  Student.objects.get(roll=1)    ***# একটিমাত্র ডাটা ফেরত দেবে***  ***3. শর্ত দিয়ে ডাটা আনা***  Student.objects.filter(marks\_\_gte=80)  *# যাদের নম্বর ≥ 80*    ***4. এক্সক্লুড করা (exclude)***  Student.objects.exclude(marks\_\_lt=50)    *# ৫০ এর নিচে যারা নেই*    ***5. অর্ডার করা***  Student.objects.all().order\_by('marks') *# ascending*  Student.objects.all().order\_by('-marks') *# descending*    ***6. কতগুলো row আছে (count)***  Student.objects.filter(marks\_\_gte=80).count()    ***7. প্রথম ও শেষ ডাটা***  Student.objects.first()  Student.objects.last()    ***8. লিমিট করা***  Student.objects.all()[:5] *# প্রথম ৫ জন*    ***🔹 QuerySet এর বৈশিষ্ট্য***   1. ***Lazy → QuerySet সাথে সাথে SQL চালায় না। যখন তুমি result ব্যবহার করো তখন SQL রান হয়।***   students = Student.objects.all() *# এখানে query এখনো রান হয়নি*  list(students) *# এখানে query রান হবে*     1. ***Chainable → একাধিক method একসাথে ব্যবহার করা যায়।***   Student.objects.filter(marks\_\_gte=80).order\_by('-marks')[:3]    ***👉 মানে: ৮০ এর উপরে যারা → তাদের মার্কস descending এ সাজাও → প্রথম ৩ জন নাও।***  ***✅ সংক্ষেপে:***   * ***QuerySet = ডাটাবেজ থেকে আনা ডাটার collection*** * ***এটা Lazy & Chainable*** * ***এটা ORM এর মাধ্যমে SQL query চালায়, কিন্তু Python object আকারে result দেয়***   ***চাও কি আমি এখন একটা QuerySet vs SQL তুলনা টেবিল বানিয়ে দিই? এতে তুমি একসাথে দেখতে পারবে Django কোড আর SQL query কিভাবে একই কাজ করে ✨*** |

**Models**

|  |
| --- |
| 📒 Django Model – Note  🔹 Model কী?   * Django তে Model হলো Python Class * Model → Database Table define করে * Class এর Attribute (Variable) → Table এর Column * ORM (Object Relational Mapper) Model থেকে SQL Table তৈরি করে   🔹 Model লেখার ধাপ  Example – models.py  from django.db import models  class Student(models.Model):  name = models.CharField(max\_length=100) *# String field*  roll = models.IntegerField(unique=True) *# Integer field, must be unique*  marks = models.FloatField() *# Decimal/Float number*  passed = models.BooleanField(default=True) *# True/False field*  🔹 Model → SQL Table  উপরের Model কে ORM এই SQL এ রূপান্তর করবে ⬇️  CREATE TABLE Student (  id INTEGER PRIMARY KEY AUTOINCREMENT,  name VARCHAR(100),  roll INTEGER UNIQUE,  marks FLOAT,  passed BOOLEAN DEFAULT TRUE  );  🔹 Database এ Model Create করার কমান্ড  python manage.py makemigrations *# Model → Migration তৈরি করবে*  python manage.py migrate *# Migration → Database এ টেবিল বানাবে*  🔹 Model ব্যবহার (CRUD)  1. Create (Insert)  s1 = Student(name="Farhan", roll=1, marks=90, passed=True)  s1.save()  2. Read (Select)  Student.objects.all() *# সব ডাটা*  Student.objects.get(roll=1) *# নির্দিষ্ট ডাটা*  Student.objects.filter(marks\_\_gte=80) *# শর্ত দিয়ে ডাটা*  3. Update  s1 = Student.objects.get(roll=1)  s1.marks = 95  s1.save()  4. Delete  s1 = Student.objects.get(roll=1)  s1.delete()  🔹 Extra Example – Teacher Model  class Teacher(models.Model):  name = models.CharField(max\_length=100)  subject = models.CharField(max\_length=50)  join\_date = models.DateField()  ✅ Summary   * Model = Python Class → Database Table * Attribute = Class Variable → Table Column * ORM Model কে SQL এ কনভার্ট করে টেবিল বানায় * CRUD কাজ Model এর মাধ্যমে করা যায় (SQL লাগবে না) |

**Migration**

|  |
| --- |
| **Model লিখলে সেটা Database এ টেবিলে রূপান্তর করতে Migration লাগে।**  **📒 Migration – Note**  **🔹 Migration কী?**   * **Migration হলো Django-তে Model → Database Table বানানোর প্রক্রিয়া।** * **Model এ যেকোনো পরিবর্তন হলে Migration এর মাধ্যমে Database এ সেই পরিবর্তন Apply হয়।** * **Migration basically → Model change history**   **🔹 Migration করার ধাপ**  **1. Model লেখো (models.py তে)**  from django.db import models    class Student(models.Model):  name = models.CharField(max\_length=100)  roll = models.IntegerField(unique=True)  marks = models.FloatField()    **2. Migration তৈরি করা টার্মিনালে লিখো ⬇️**  python manage.py makemigrations    **👉 এটা migrations/ ফোল্ডারে ফাইল বানাবে, যেমন:**  0001\_initial.py    **3. Migration ফাইল (Example: 0001\_initial.py)**  from django.db import migrations, models    class Migration(migrations.Migration):    initial = True    dependencies = [  ]    operations = [  migrations.CreateModel(  name='Student',  fields=[  ('id', models.BigAutoField(auto\_created=True,  primary\_key=True,  serialize=False,  verbose\_name='ID')),  ('name', models.CharField(max\_length=100)),  ('roll', models.IntegerField(unique=True)),  ('marks', models.FloatField()),  ],  ),  ]    **👉 এখানে Django বলছে → একটা Student নামের টেবিল বানাও**  **4. Migration Apply করা (Database এ Table তৈরি হবে)**  python manage.py migrate    **👉 এখন Database এ Student টেবিল তৈরি হয়ে যাবে।**  **🔹 Migration Commands (খুব দরকারি)**   * **Migration বানানো:**   python manage.py makemigrations     * **Migration Apply করা:**   python manage.py migrate     * **Migration List দেখা:**   python manage.py showmigrations     * **নির্দিষ্ট Migration Apply করা:**   python manage.py migrate appname 0001     * **SQL Query দেখা (Migration এর জন্য):**   python manage.py sqlmigrate appname 0001    **✅ Summary**   * **Model → Database Table বানাতে Migration দরকার** * **makemigrations → Migration ফাইল বানায়** * **migrate → Database এ Apply করে** * **Migration = Database এর change history** |

**Build-in Field Option**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **📒 Django Field Options — Step by Step Guide**  **Step 1) null vs blank — কখন কোনটা?**   * **null=True → DB level: কলামে NULL রাখা যাবে।** * **blank=True → Form/Admin level: ফাঁকা রাখলেও validation পাস করবে।** * **Thumb rule:**   + **CharField/TextField → সাধারণত null=False, blank=True (empty string "" ভালো; NULL এভয়েড)**   + **Numeric/Date/ForeignKey → প্রয়োজনে null=True**   name = models.CharField(max\_length=100, blank=True) *# ✅*  age = models.IntegerField(null=True, blank=True) *# ✅*  about = models.TextField(null=False, blank=True) *# ✅*    **Step 2) Uniqueness & Primary Key**   * **unique=True → কলামে একই ভ্যালু একাধিকবার রাখা যাবে না (DB-level constraint + index)** * **primary\_key=True → এই ফিল্ডটাই PK হবে (তাহলে Django আর id বানাবে না)** * **একাধিক ফিল্ডে যৌথ ইউনিক দরকার? UniqueConstraint (Model Meta-তে)**   roll = models.IntegerField(unique=True) *# ইউনিক কলাম*  slug = models.SlugField(unique=True, db\_index=True) *# ইউনিক + ইনডেক্স*  *# যৌথ ইউনিক:*  class Meta:  constraints = [  models.UniqueConstraint(fields=['class\_name', 'roll'], name='uniq\_class\_roll')  ]    **Step 3) Default Values**   * **default= এ স্ট্যাটিক ভ্যালু বা callable (ফাংশন) ব্যবহার করা যায়।** * **সময়/তারিখের জন্য default=timezone.now ভালো; auto\_now\* আলাদা (নিচে আছে)।**   from django.utils import timezone  status = models.CharField(max\_length=20, default='active')  joined\_at= models.DateTimeField(default=timezone.now) *# callable*    **Step 4) Auto Timestamps**   * **auto\_now\_add=True → create সময় একবার সেট** * **auto\_now=True → প্রতি save()-এ আপডেট** * **এগুলো সাধারণত editable=False হয়ে যায়—ম্যানুয়ালি সেট করতে হয় না।**   created\_at = models.DateTimeField(auto\_now\_add=True) *# first create*  updated\_at = models.DateTimeField(auto\_now=True) *# every save*    **ম্যানুয়ালি কন্ট্রোল করতে চাইলে auto\_now\*/default না দিয়ে নিজেই সেট করো।**  **Step 5) Choices (Fixed options)**   * **ফর্মে ড্রপডাউন, ডাটায় নিয়ন্ত্রিত ভ্যালু।** * **TextChoices/IntegerChoices ব্যবহার করলে কোড পরিষ্কার হয়।**   class Grade(models.TextChoices):  A = 'A', 'Excellent'  B = 'B', 'Good'  C = 'C', 'Average'    grade = models.CharField(max\_length=1, choices=Grade.choices, default=Grade.C)  *# ব্যবহার: obj.grade == Grade.A*    **Step 6) Indexing (Performance)**   * **db\_index=True দিলে ওই কলামে ইনডেক্স তৈরি হয় → ফিল্টার/সোর্ট দ্রুত।** * **unique=True দিলে স্বয়ংক্রিয়ভাবেই ইনডেক্স থাকে।**   email = models.EmailField(db\_index=True)    **Heavy সার্চ/সোর্টে ইনডেক্স দরকার—অন্যথায় অযথা ইনডেক্স এভয়েড (write ধীর হয়)।**  **Step 7) Validation (clean data in, clean data out)**   * **validators=[...] দিয়ে ফিল্ড-লেভেল চেক।** * **Ready-made: MinValueValidator, MaxValueValidator, RegexValidator ইত্যাদি।** * **কাস্টম validator: ভুল হলে ValidationError raise করতে হবে।**   from django.core.validators import MinValueValidator, RegexValidator  from django.core.exceptions import ValidationError    def even\_only(value):  if value % 2:  raise ValidationError("Must be an even number.")    age = models.IntegerField(validators=[MinValueValidator(0)])  roll = models.IntegerField(validators=[even\_only])  phone = models.CharField(  max\_length=14,  validators=[RegexValidator(r'^\+?[0-9]{10,14}$', "Invalid phone number")]  )    **Step 8) Human-friendly labels**   * **verbose\_name → অ্যাডমিন/ফর্মে সুন্দর নাম** * **help\_text → ইনপুটের নিচে গাইড টেক্সট**   full\_name = models.CharField(  max\_length=100,  verbose\_name="Student Full Name",  help\_text="As on admission form"  )    **Step 9) Editability & Display**   * **editable=False → ফর্ম/অ্যাডমিনে এডিট করা যাবে না (DB তে থাকবে)** * **choices + get\_FOO\_display() → লেবেল দেখাতে সুবিধা**   created\_at = models.DateTimeField(auto\_now\_add=True, editable=False)  *# টেমপ্লেটে: {{ obj.get\_grade\_display }} → "Excellent"*    **Step 10) ForeignKey/Relation-specific Options**  **এগুলোও ফিল্ডের options—খুব কাজে লাগে।**   * **on\_delete= (Required): models.CASCADE / PROTECT / SET\_NULL / DO\_NOTHING / SET\_DEFAULT** * **related\_name=: reverse relation এর নাম** * **related\_query\_name=: filter()-এ রিভার্স নাম** * **limit\_choices\_to=: ফর্মে রিলেটেড অবজেক্ট লিস্ট ফিল্টার**   class Teacher(models.Model):  name = models.CharField(max\_length=100)    class Student(models.Model):  teacher = models.ForeignKey(  Teacher,  on\_delete=models.SET\_NULL, null=True, blank=True,  related\_name='students',  limit\_choices\_to={'name\_\_startswith': 'A'}  )    **Step 11) Error messages & DB details**   * **error\_messages={...} → কাস্টম এরর টেক্সট** * **db\_column='...' → কাস্টম কলাম নাম** * **unique\_for\_date/month/year → ModelForm-level ইউনিকনেস (DB constraint নয়)**   slug = models.SlugField(  unique=True,  error\_messages={'unique': "This slug is already used."},  db\_column='post\_slug'  )    title = models.CharField(max\_length=200, unique\_for\_date='published\_at')  published\_at = models.DateField()  *# Note: unique\_for\_\* is form/clean-time check, DB-level নয়।*    **Step 12) সম্পূর্ণ উদাহরণ (Student)**  from django.db import models  from django.utils import timezone  from django.core.validators import MinValueValidator, RegexValidator    class Grade(models.TextChoices):  A = 'A', 'Excellent'  B = 'B', 'Good'  C = 'C', 'Average'    class Student(models.Model):  full\_name = models.CharField(max\_length=100, blank=True,  verbose\_name="Student Full Name",  help\_text="As on admission form")  roll = models.IntegerField(unique=True,  validators=[MinValueValidator(1)])  email = models.EmailField(db\_index=True, blank=True)  phone = models.CharField(max\_length=14, blank=True,  validators=[RegexValidator(r'^\+?[0-9]{10,14}$')])  grade = models.CharField(max\_length=1, choices=Grade.choices, default=Grade.C)  marks = models.FloatField(null=True, blank=True, default=0)  joined\_at = models.DateTimeField(default=timezone.now) *# ম্যানুয়ালি কন্ট্রোলযোগ্য*  created\_at = models.DateTimeField(auto\_now\_add=True, editable=False)  updated\_at = models.DateTimeField(auto\_now=True, editable=False)    class Meta:  constraints = [  models.UniqueConstraint(fields=['email', 'roll'], name='uniq\_email\_roll')  ]    def \_\_str\_\_(self):  return f"{self.full\_name or 'Unnamed'} ({self.roll})"    **Step 13) Migration flow (Options বদলালে)**  python manage.py makemigrations *# নতুন/বদলানো অপশন ধরবে*  python manage.py migrate *# DB-তে apply*  python manage.py showmigrations *# কোনগুলো পেন্ডিং/অ্যাপ্লাইড দেখো*    **Step 14) Quick Recipes**   * **Optional text: CharField(null=False, blank=True)** * **Optional number: IntegerField(null=True, blank=True)** * **Slug/username: SlugField(unique=True, db\_index=True)** * **Timestamps: created\_at=auto\_now\_add=True, updated\_at=auto\_now=True** * **Controlled list: choices=Enum.choices, দেখাতে get\_FOO\_display()** * **Speed up search: db\_index=True (যেখানে অনেক ফিল্টার/অর্ডার হয়)** * **Safe date default: default=timezone.now (not auto\_now\_add যদি ম্যানুয়াল দরকার হয়)**  | **Field Type** | **ব্যবহার** | | --- | --- | | CharField | ছোট টেক্সট | | TextField | বড় টেক্সট | | IntegerField | পূর্ণ সংখ্যা | | FloatField | দশমিক সংখ্যা | | DecimalField | নির্দিষ্ট precision দশমিক | | PositiveIntegerField | শুধু ধনাত্মক সংখ্যা | | BooleanField | True/False | | DateField | তারিখ | | TimeField | সময় | | DateTimeField | তারিখ + সময় | | FileField | ফাইল আপলোড | | ImageField | ছবি আপলোড | | URLField | ওয়েবসাইট লিংক | | EmailField | ইমেইল | | ForeignKey | One-to-Many সম্পর্ক | | OneToOneField | One-to-One সম্পর্ক | | ManyToManyField | Many-to-Many সম্পর্ক | | SlugField | URL slug | | UUIDField | Unique ID | | GenericIPAddressField | IP Address | | DurationField | সময়ের ব্যবধান | |