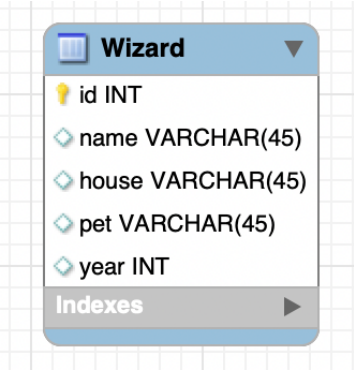
SQL Queries to ORM

Let's do a little refresher on SQL queries, and understand what's going on behind the scenes when we use our ORM commands.

First, create the following table schema in MySQL workbench and forward-engineer your database.



Now, in a new Django project, open up your models.py file and put the following model there:

```
class Wizard(models.Model):
    name = models.CharField(max_length=45)
    house = models.CharField(max_length=45)
    pet = models.CharField(max_length=45)
    year = models.IntegerField()
```

Once you have the model, run the two migration commands in your terminal and open the shell:

```
python manage.py makemigrations
python manage.py migrate
python manage.py shell
```

You should now be ready to run SQL queries in MySQLWorkbench and run ORM commands in the shell.

SQL to ORM

For each of the following SQL queries, submit the corresponding ORM command that would do the same thing.

Note: You may need to change your database name to get these queries to run in MySQLWorkbench. i.e. INSERT INTO hogwarts_db.Wizard

```
INSERT INTO hogwarts.wizard (name, house, pet, year) VALUES ('Harry Potter', 'Gryffindor', 'Hedwig', '5');
INSERT INTO hogwarts.wizard (name, house, pet, year) VALUES ('Hermione Granger', 'Gryffindor', 'Crookshanks', '5');
SELECT * FROM hogwarts.wizard WHERE id = 1;
SELECT * FROM hogwarts.wizard WHERE house = 'Gryffindor';
UPDATE hogwarts.wizard SET year = '6' WHERE id = 1;
```

ORM to SQL

For each of the following ORM command segments, submit the corresponding SQL query that would do the same thing.

```
Wizard.objects.create(name="Luna Lovegood", house="Ravenclaw", pet="None", year="4")
Wizard.objects.create(name="Padma Patil", house="Ravenclaw", pet="None", year="5")
ravenclaws = Wizard.objects.filter(house="Ravenclaw")
luna = Wizard.objects.get(name="Luna Lovegood")
luna.year = 5
luna.save()
```

Try running your SQL queries in MySQLWorkbench and your ORM commands in the shell and see if you were correct!