УДК 004.94
Рябенко А.Є.[1] , Гавришко А.Ю.[2], Широкорад Д.В.[3], Терещенко Е.В.[1]
[1] доц. НУ «Запорізька політехніка»
[2] студ. гр. КНТ-819 НУ «Запорізька політехніка»
[3] старш. викл. НУ «Запорізька політехніка»

РОЗРОБКА ПРОГРАМНОЇ БІБЛІОТЕКИ ДЛЯ ОБ'ЄКТНО-РЕЛЯЦІЙНОГО ВІДОБРАЖЕННЯ БАЗ ДАНИХ

У створенні веб-додатків істотну роль відіграють засоби для роботи з базами даних, як правило, їх реляційними видами. Така технологія як ORM (Object-Relational Mapping) дозволяє застосовувати принципи та підходи об'єктно-орієнтованого програмування для взаємодії з базами даних. Як відомо, основною програмною виразкою для створення запитів до баз даних є SQL. ORM є надбудовою вищого рівня абстракції, при реалізації якої немає необхідності писати рутинні запити SQL, застосовуючи замість них властивості та методи спеціальних класів, структура яких відповідає структурі таблиць та полів бази даних, а набір функцій дозволяє автоматично генерувати SQL запити для створення, модифікації, видалення та читання даних. ORM застосовується, зокрема, у таких бібліотеках Python для веб-розробки як Django та Flask-SQLAlchemy.

В рамках цієї роботи був розроблений програмний продукт MagicSQL, що реалізує засобами мови програмування Python автоматизацію дій з базами даних SQLite в об'єктно-орієнтованому стилі. Зокрема застосування цієї технології дозволяє створити модель даних у вигляді класів-спадкоємців класу Table з бібліотеки. Статичні поля класів визначають структуру таблиць бази даних. Після створення моделі метод фізично create_all створює базу даних та таблиці у ній відповідної структури. Крім полів, визначених користувачем, кожна таблиця отримує цілий первинний ключ (поле id). При цьому, як і при виконанні інших дій, немає необхідності створювати самостійно SQL запити, вони генеруються і запускаються автоматично, при цьому одна з налаштувань дозволяє користувачеві бачити в консолі запити, що виконуються.

Після створення бази даних та таблиць природним продовженням буде заповнення таблиць даними. Тут також застосовується ООП підхід, у якого створення об'єкта класу, що представляє таблицю, означає автоматичне внесення рядка даних у таблицю. Наприклад, ми маємо таблицю Language, що зберігає дані про мови програмування з полями name, year, rating (назва, рік створення, рейтинг). Для додавання запису достатньо створити об'єкт класу Language та викликати його метод add:

```
python = Language(name='Python', year=1991, rating=8.5)
```

```
python.add()
```
при цьому генерується та виконується SQL-запит:

```
INSERT INTO Language (name, year, rating)
VALUES
('Python', 1991, 8.5);
```

Більш складною з програмної точки зору була реліз зв'язків «one-to many» між таблицями. Наприклад, якщо таблиця з даними про проекти Project пов'язана за зовнішнім ключем із таблицею Language, то на рівні окремих об'єктів має бути можливість встановлення зв'язків за допомогою методів цих об'єктів. Так, якщо ми хочемо встановити зв'язок між проектом «web-site» та мовою Python, то в коді нам достатньо викликати метод relate:

```
site = Project(name='company web-site')
site.relate(python)
```

при цьому таблиці автоматично вносяться відповідні значення зовнішніх ключів для реалізації зв'язків між рядками.

Також реалізовано вибірку даних. Наприклад, метод filter дозволяє читати записи таблиці з укзанними обмеженнями. Наприклад, якщо нам потрібні мови програмування з рейтингом не нижче 8.0, це виконується таким чином:

```
Language.objects().filter(rating = '>= 8.0')
```

Для вибірки також реалізовані методи all і order_by дозволяють отримати неупорядковані записи і впорядковані за яким-небудь полем або групою полів записи відповідно. Як подальше вдосконалення проекту заплановано реалізацію зв'язку «many-to-many».