Метапрограмування

Лабораторна робота № 3

Проектування та розробка засобів об'єктно-реляційного відображення

Завдання

Використовуючи мову програмування Python, розробити пакет для `pip`, який реалізує повну або часткову трансляцію об'єктів екземплярів, об'єктів класів та класових ієрархій в термінах мови Python в один з обраних діалектів мови SQL^*, реалізуючи повне або часткове об'єктно-реляційне відображення (ORM), де:

— **Повне відображення** $Python \leftrightarrows SQL^*$ – це відображення об'єктного коду в термінах мови Python в обраний діалект мови SQL^* та зворотнє відображення SQL^*-запитів в об'єктний код в термінах мови Python.

— **Часткове відображення** $Python \rightarrow SQL^*/Python \leftarrow SQL^*$ – це відображення об'єктного коду в термінах мови Python в обраний діалект мови SQL^* або зворотнє відображення SQL^*-запитів в об'єктний код в термінах мови Python.

Розроблений пакет має інтегруватися в інші модулі та пакети Python шляхом імпортування та повинен бути опублікований на pypi.org. В рамках пакету мають бути реалізовані наступні можливості:

1. **Трансляція об'єктів екземплярів.** В залежності від обраного варіанту об'єктно-реляційного відображення, система має вміти трансформувати:

   (a) об'єкти екземплярів в термінах мови Python $\longrightarrow$ у певні SQL^*-запити, які після виконання створюватимуть або певним чином змінюватимуть вже існуюче представлення цих об'єктів екземплярів у відповідній реляційній базі даних ($Python \rightarrow SQL^*$);

(б) інформацію отриману шляхом виконання певних SQL^*-запитів до обраної реляційної бази даних $\longrightarrow$ в об'єкти екземплярів в термінах мови Python шляхом генерації відповідного програмного коду в термінах мови Python із подальшим його збереженням у відповідні модулі проєкту та використанням під час виконання програми ($Python \leftarrow SQL^*$).

2. **Трансляція об'єктів класів.** В залежності від обраного варіанту об'єктно-реляційного відображення, система має вміти трансформувати:

   (а) об'єкти класів в термінах мови Python $\longrightarrow$ у певні SQL^*-запити, які після виконання створюватимуть або певним чином змінюватимуть вже існуюче представлення цих об'єктів класів у відповідній реляційній базі даних ($Python \rightarrow SQL^*$);

   (б) інформацію отриману шляхом виконання певних SQL^*-запитів до обраної реляційної бази даних $\longrightarrow$ в об'єкти класів в термінах мови Python шляхом генерації відповідного програмного коду в термінах мови Python із подальшим його збереженням у відповідні модулі проєкту та використання під час виконання програми ($Python \leftarrow SQL^*$).

3. **Трансляція класових ієрархій.** В залежності від обраного варіанту об'єктно-реляційного відображення, система має вміти трансформувати:

   (а) класові ієрархії в термінах мови Python $\longrightarrow$ у певні SQL^*-запити, які після виконання створюватимуть або певним чином змінюватимуть вже існуюче представлення цих ієрархій у відповідній реляційній базі даних ($Python \rightarrow SQL^*$);

   (б) інформацію отриману шляхом виконання певних SQL^*-запитів до обраної бази даних $\longrightarrow$ у класові ієрархії в термінах мови Python шляхом генерації відповідного програмного коду в термінах мови Python із подальшим його збереженням у відповідні модулі проєкту та використання під час виконання програми ($Python \leftarrow SQL^*$).

Розробка системи має здійснюватися командою студентів:

— для варіантів N.(а) – 4 студенти,

— для варіантів N.(б) – 3 студенти,

— для варіантів N.(в) – 2 студенти,

де N – номер обраного варіанту. Загальний обсяг задач в рамках реалізації системи повинен бути рівномірно розподілений між усіма членами команди.

Функціональні вимоги до системи

Забороняється використовувати будь-які сторонні інструменти об'єктно-реляційного відображення (готові ORM, тощо) як частини системи.

Потрібно реалізувати:

1. Пакет для мови Python, який реалізує повну або часткову трансляцію об'єктів екземплярів, об'єктів класів та класових ієрархій на мові Python в один з обраних діалектів мови SQL^* та виконує пошук/читання/зберігання/модифікацію/видалення інформації через взаємодію з певною реляційною базою даних (визначеною умовою обраного варіанту).

2. Відображення між системою стандартних типів мови Python та системою типів діалекту мови SQL^* (в залежності від обраного варіанту).

3. Клас `Py2SQL` в рамках якого реалізувати набір спеціалізованих методів для роботи з відповідною реляційною базою даних та різними варіантами об'єктно-реляційного відображення:

   — $\boldsymbol{Python \rightleftarrows SQL^*/Python \leftarrow SQL^*/Python \rightarrow SQL^*}$:

   1. `Py2SQL.db_connect(db)` – встановлює з'єднання із базою даних через об'єкт `db`, якій містить всю необхідну інформацію для встановлення з'єднання.
   2. `Py2SQL.db_disconnect()` – розриває поточне з'єднання із базою даних.
   3. `Py2SQL.db_engine` – повертає назву та версію СУБД.
   4. `Py2SQL.db_name` – повертає назву бази даних.
   5. `Py2SQL.db_size` – повертає розмір бази даних у Mb.
   6. `Py2SQL.db_tables` – повертає список таблиць бази даних.
   7. `Py2SQL.db_table_structure(table)` – повертає упорядкований список кортежів виду `(id, name, type)`, де `id` – номер атрибуту таблиці `table`, `name` – ім'я атрибуту, `type` – тип атрибуту.
   8. `Py2SQL.db_table_size(table)` – повертає розмір таблиці `table` бази даних у Mb.

   — $\boldsymbol{Python \rightleftarrows SQL^*}$:

   1. `Py2SQL.find_object(table, py_object)` – повертає запис таблиці `table` бази даних, який є еквівалентним об'єкту `py_object` у вигляді списку кортежів виду `(attribute, type, value)`, якщо такий запис існує.
   2. `Py2SQL.find_objects_by(table, *attributes)` – повертає упорядкований список записів таблиці `table` бази даних, що містять атрибути вказані у послідовності `*attributes=[(name, value)]`. Кожен запис представляється у вигляді списку кортежів виду `(attribute, type, value)`.
   3. `Py2SQL.find_class(py_class)` – повертає упорядкований список структурних елементів таблиці `table` бази даних, якщо вона є структурно еквівалентною класу `py_class`, при умові що така таблиця існує. Структурні елементи таблиці представляються у вигляді кортежу `(attribute, type, value)`.
   4. `Py2SQL.find_classes_by(*attributes)` – повертає упорядкований список списків структурних елементів таблиць бази даних, що містять атрибути

вказані у послідовності `*attributes=[(attribute1_name)]`, при умові що такі таблиці існують. Кожен структурний елемент представляється у вигляді кортежу виду `(attribute, type)`.

5. `Py2SQL.find_hierarches()` – повертає усі табличні ієрархії бази даних у вигляді списку списків кортежів виду `(parent_table, child_table)`, де кожен окремий список відповідає окремій табличній ієрархії.

6. `Py2SQL.create_object(table, id)` – динамічно створює новий об'єкт на основі запису з номером `id` із таблиці `table` бази даних.

7. `Py2SQL.create_objects(table, fid, lid)` – динамічно створює список нових об'єктів, які створюються на основі записів таблиці `table` бази даних починаючи із запису із номером `fid` завершуючи записом з номером `lid` включно.

8. `Py2SQL.create_class(table, module)` – динамічно створює на основі таблиці `table` програмний код нового класу та зберігає його в модулі `module`, після чого динамічно імпортує новостворений клас із нового створеного модуля до глобальної області видимості запущеної програми.

9. `Py2SQL.create_hierarchy(table, package)` – динамічно створює на основі таблиці `table` та усіх зв'язаних із нею таблиць (у тому числі і транзитивно зв'язаних) програмний код нової класової ієрархії та зберігає окремі її класи у відповідні модулі пакету `package`, після чого динамічно імпортує новостворені класи із новостворених модулів пакету `package` до глобальної області видимості запущеної програми.

10. `Py2SQL.save_object(object)` – створює представлення об'єкта `object` у базі даних або оновлює його якщо воно вже існує.

11. `Py2SQL.save_class(class)` – створює представлення класу `class` у базі даних або оновлює його якщо воно вже існує.

12. `Py2SQL.save_hierarchy(root_class)` – створює представлення класової ієрархії `hierarchy`, де `root_class` – це корінь ієрархії, у базі даних або оновлює його якщо воно вже існує.

13. `Py2SQL.delete_object(object)` – знаходить та видаляє з бази даних представлення об'єкта `object`, якщо воно існувало.

14. `Py2SQL.delete_class(class)` – знаходить та видаляє з бази даних представлення класу `class`, якщо воно існувало.

15. `Py2SQL.delete_hierarchy(root_class)` – знаходить та видаляє з бази даних представлення класової ієрархії `hierarchy`, де `root_class` – це корінь ієрархії, якщо воно існувало.

— ***Python*** $\leftarrow$ ***SQL****:

1. `Py2SQL.find_object(table, py_object)` – повертає запис таблиці `table` бази даних, який є еквівалентним об'єкту `py_object` у вигляді списку корте-

жів виду `(attribute, type, value)`, якщо такий запис існує.

2. `Py2SQL.find_objects_by(table, *attributes)` – повертає упорядкований список записів таблиці `table` бази даних, що містять атрибути вказані у послідовності `*attributes=[(name, value)]`. Кожен запис представляється у вигляді списку кортежів виду `(attribute, type, value)`.

3. `Py2SQL.find_class(py_class)` – повертає упорядкований список структурних елементів таблиці `table` бази даних, якщо вона є структурно еквівалентною класу `py_class`, при умові що така таблиця існує. Структурні елементи таблиці представляються у вигляді кортежу `(attribute, type, value)`.

4. `Py2SQL.find_classes_by(*attributes)` – повертає упорядкований список списків структурних елементів таблиць бази даних, що містять атрибути вказані у послідовності `*attributes=[(attribute1_name,...)]`, при умові що такі таблиці існують. Кожен структурний елемент представляється у вигляді кортежу виду `(attribute, type)`

5. `Py2SQL.create_object(table, id)` – динамічно створює новий об'єкт на основі запису з номером `id` із таблиці `table` бази даних.

6. `Py2SQL.create_objects(table, fid, lid)` – динамічно створює список нових об'єктів, які створюються на основі записів таблиці `table` бази даних починаючи із запису із номером `fid` завершуючи записом з номером `lid` включно.

7. `Py2SQL.create_class(table, module)` – динамічно створює на основі таблиці `table` програмний код нового класу та зберігає його в модулі `module`, після чого динамічно імпортує новостворений клас із нового створеного модуля до глобальної області видимості запущеної програми.

8. `Py2SQL.create_hierarchy(table, package)` – динамічно створює на основі таблиці `table` та усіх зв'язаних із нею таблиць (у тому числі і транзитивно зв'язаних) програмний код нової класової ієрархії та зберігає окремі її класи у відповідні модулі пакету `package`, після чого динамічно імпортує новостворені класи із новостворених модулів пакету `package` до глобальної області видимості запущеної програми.

— ***Python → SQL****:

1. `Py2SQL.save_object(object)` – створює представлення об'єкта `object` у базі даних або оновлює його якщо воно вже існує.

2. `Py2SQL.save_class(class)` – створює представлення класу `class` у базі даних або оновлює його якщо воно вже існує.

3. `Py2SQL.save_hierarchy(root_class)` – створює представлення класової ієрархії `hierarchy`, де `root_class` – це корінь ієрархії, у базі даних або оновлює його якщо воно вже існує.

4. `Py2SQL.delete_object(object)` – знаходить та видаляє з бази даних пред-

ставлення об'єкта `object`, якщо воно існувало.

5. `Py2SQL.delete_class(class)` – знаходить та видаляє з бази даних представлення класу `class`, якщо воно існувало.

6. `Py2SQL.delete_hierarchy(root_class)` – знаходить та видаляє з бази даних представлення класової ієрархії `hierarchy`, де `root_class` – це корінь ієрархії, якщо воно існувало.

4. Підтримку типів `int`, `float`, `str`, `list`, `tuple`, `dict`, `set`, `frozenset`, `array`, `object` в якості атрибутів об'єктів екземплярів та об'єктів класів. Колекції `list`, `tuple`, `dict`, `set`, `frozenset` можуть містити лише об'єкти стандартних типів даних.

5. Повну `html`-документацію пакету згенеровану за допомогою модуля `pydoc`.

Реалізація будь-яких додаткових функціональних можливостей для обраної, відповідно до варіанту, конфігурації пакету буде оцінюватися у вигляді додаткових балів в залежності від складності та кількості реалізованих додаткових функцій кожним учасником команди.

Демонстрація

Потрібно продемонструвати усі функціональні можливості пакету. Демонстрація роботи розробленого пакету повинна відбуватися з використанням:

— класових ієрархій із стандартної бібліотеки мови Python та/або спеціалізованих програмних пакетів доступних на GitHub чи GitLab;

— наповнених інформацією реляційних баз даних, що відповідають обраному варіанту.

Варіанти

1. **Apache Derby:**

   1. *Python* 3 $\rightleftarrows$ *Apache Derby SQL* (**16 балів**).
   2. *Python* 3 $\leftarrow$ *Apache Derby SQL* (**14 балів**).
   3. *Python* 3 $\rightarrow$ *Apache Derby SQL* (**11 балів**).

   Дистрибутив та документація на офіційному вебсайті Apache Derby 10.15.x.

2. **CUBRID:**

   1. *Python* 3 $\rightleftarrows$ *CUBRID SQL* (**16 балів**).
   2. *Python* 3 $\leftarrow$ *CUBRID SQL* (**14 балів**).
   3. *Python* 3 $\rightarrow$ *CUBRID SQL* (**11 балів**).

   Дистрибутив та документація на офіційному вебсайті CUBRID 10.x.

3. **Firebird:**

1. *Python* 3 $\rightleftarrows$ *Firebird SQL* (**16 балів**).
2. *Python* 3 $\leftarrow$ *Firebird SQL* (**14 балів**)
3. *Python* 3 $\rightarrow$ *Firebird SQL* (**11 балів**).

Дистрибутив та документація на офіційному вебсайті Firebird 3.0.x.

4. **H2:**
    1. *Python* 3 $\rightleftarrows$ *H2 SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *H2 SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *H2 SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті H2 1.4.x.

5. **HSQLDB:**
    1. *Python* 3 $\rightleftarrows$ *HSQLDB SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *HSQLDB SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *HSQLDB SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті HSQLDB 2.5.x.

6. **IBM Db2:**
    1. *Python* 3 $\rightleftarrows$ *IBM Db2 SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *IBM Db2 SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *IBM Db2 SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті IBM Db2 Database 11.5.x CE.

7. **MariaDB:**
    1. *Python* 3 $\rightleftarrows$ *MariaDB SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *MariaDB SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *MariaDB SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті MariaDB 10.5.x CS.

8. **Microsoft SQL Server:**
    1. *Python* 3 $\rightleftarrows$ *Microsoft SQL Server SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *Microsoft SQL Server SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *Microsoft SQL Server SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті Microsoft SQL Server 2019 Express.

9. **Microsoft SQL Server Compact:**

1. *Python* 3 $\rightleftarrows$ *Microsoft SQL Server Compact SQL* (**16 балів**).
2. *Python* 3 $\leftarrow$ *Microsoft SQL Server Compact SQL* (**14 балів**).
3. *Python* 3 $\rightarrow$ *Microsoft SQL Server Compact SQL* (**11 балів**).

Дистрибутив та документація на офіційному вебсайті Microsoft SQL Server Compact 4.0 SP1.

10. **MonetDB:**
    1. *Python* 3 $\rightleftarrows$ *MonetDB SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *MonetDB SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *MonetDB SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті MonetDB 11.37.x.

11. **MySQL:**
    1. *Python* 3 $\rightleftarrows$ *MySQL SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *MySQL SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *MySQL SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті MySQL Community Server 8.0.x.

12. **OmniSciDB:**
    1. *Python* 3 $\rightleftarrows$ *OmniSciDB SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *OmniSciDB SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *OmniSciDB SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті OmniSciDB Open Source 5.4.x.

13. **Oracle:**
    1. *Python* 3 $\rightleftarrows$ *Oracle SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *Oracle SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *Oracle SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті Oracle Database 18c XE.

14. **PostgreSQL:**
    1. *Python* 3 $\rightleftarrows$ *PostgreSQL SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *PostgreSQL SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *PostgreSQL SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті PostgreSQL 12.

15. **SQLite:**

    1. *Python* 3 $\rightleftarrows$ *SQLite SQL* (**16 балів**).
    2. *Python* 3 $\leftarrow$ *SQLite SQL* (**14 балів**).
    3. *Python* 3 $\rightarrow$ *SQLite SQL* (**11 балів**).

    Дистрибутив та документація на офіційному вебсайті SQLite 3.x.