МГТУ им. Н.Э. Баумана

Отчёт по рубежному контролю №1
по курсу «Базовые компоненты и интернет-технологии»
Вариант 12.

Руководитель
Гапанюк Ю.Е.
28.10.2022

Студент группы ИУ5-33Б
Поляков Д.Д.
28.10.2022

2022 г.

Полученное задание:

Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

Необходимо разработать запросы в соответствии с Вашим вариантом.

Предметная область: класс_1 – Язык программирования, класс_2 – Среда разработки, вариант запросов: В.

Запросы:

1. «Язык программирования» и «Инструмент разработки» связаны соотношением один-ко-многим. Выведите список всех инструментов разработки, у которых название начинается с буквы «А», и названия их языков программирования.
2. «Язык программирования» и «Инструмент разработки» связаны соотношением один-ко-многим. Выведите список языков с минимальным количеством пользователей инструмента в каждом языке, отсортированные по минимальным пользователям.
3. «Язык программирования» и «Инструмент разработки» связаны соотношением многие-ко-многим. Выведите список всех связанных языков и инструментов, отсортированный по инструментам, сортировка по языкам произвольная.

Текст программы:

```
from operator import itemgetter


class Programming_language:
    """Сотрудник"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class Development_tool:
    """Отдел"""

    def __init__(self, id, name, users_count, programming_language_id):
        self.id = id
        self.name = name
        self.users_count = users_count
        self.programming_language_id = programming_language_id


class Programming_languageDevelopment_tool:
```

```
    """
    'Сотрудники отдела' для реализации
    связи многие-ко-многим
    """

    def __init__(self, Programming_language_id, Development_tool_id):
        self.Development_tool_id = Development_tool_id
        self.Programming_language_id = Programming_language_id


# Отделы
Development_tools = [
    Development_tool(1, 'CLion', 75000, 1),
    Development_tool(2, 'gdb', 100000, 1),

    Development_tool(3, 'Visual Studio', 85000, 2),
    Development_tool(4, '.NET core', 50000, 2),
    Development_tool(5, 'ASP.NET', 45000, 2),
    Development_tool(6, 'Entity Framework', 35000, 2),

    Development_tool(7, 'Angular', 100000, 3),
    Development_tool(8, 'React', 120000, 3),
    Development_tool(9, 'NodeJS', 150000, 3),
    Development_tool(10, 'WebStorm', 75000, 3),

    Development_tool(11, 'PyCharm', 87500, 4),
    Development_tool(12, 'Django', 50000, 4),
    Development_tool(13, 'Tensor Flow', 35000, 4),

    Development_tool(14, 'Visual Studio Code', 187000, 1),

    Development_tool(15, 'A+ PROGRAMMING IDE', 10, 5),
]

# Сотрудники
Programming_languages = [
    Programming_language(1, 'C++', ),
    Programming_language(2, 'C#', ),
    Programming_language(3, 'JavaScript', ),
    Programming_language(4, 'Python', ),
    Programming_language(5, 'A+', ),

]

Programming_languages_Development_tools = [
    Programming_languageDevelopment_tool(1, 1),
    Programming_languageDevelopment_tool(1, 2),
    Programming_languageDevelopment_tool(1, 14),

    Programming_languageDevelopment_tool(2, 3),
    Programming_languageDevelopment_tool(2, 4),
    Programming_languageDevelopment_tool(2, 5),
    Programming_languageDevelopment_tool(2, 6),
    Programming_languageDevelopment_tool(2, 14),

    Programming_languageDevelopment_tool(3, 7),
    Programming_languageDevelopment_tool(3, 8),
    Programming_languageDevelopment_tool(3, 9),
    Programming_languageDevelopment_tool(3, 10),
    Programming_languageDevelopment_tool(3, 14),

    Programming_languageDevelopment_tool(4, 11),
    Programming_languageDevelopment_tool(4, 12),
    Programming_languageDevelopment_tool(4, 13),
```

```
    Programming_languageDevelopment_tool(4, 14),

    Programming_languageDevelopment_tool(5, 14),
    Programming_languageDevelopment_tool(5, 15),
]


def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(d.name, e.name, e.users_count)
                   for d in Programming_languages
                   for e in Development_tools
                   if e.programming_language_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, pl.Programming_language_id,
pl.Development_tool_id)
                         for d in Programming_languages
                         for pl in Programming_languages_Development_tools
                         if d.id == pl.Programming_language_id]

    many_to_many = [(programming_language_name, e.name)
                    for programming_language_name, Programming_language_id,
Development_tool_id in many_to_many_temp
                    for e in Development_tools if e.id ==
Development_tool_id]

    print('Задание B1')
    res_11 = list(filter(lambda i: (i[0][0] == 'A' or i[0][0] == 'a'),
one_to_many))
    print(res_11)

    print('\nЗадание B2')
    res_2 = []
    for pl in Programming_languages:
        d_tmp = list((pl.name, d.users_count) for d in Development_tools if
d.programming_language_id == pl.id)
        if len(d_tmp) > 0:
            res_2.append(min(d_tmp, key=lambda i: i[1]))
    tmp = sorted(res_2, key=itemgetter(1))
    print(tmp)

    print('\nЗадание B3')
    # Перебираем все отделы
    many_to_many.sort(key=lambda i: i[1])
    print(many_to_many)


if __name__ == '__main__':
    main()
```

Результаты выполнения:

```
Задание B1
[('A+', 'A+ PROGRAMMING IDE', 10)]

Задание B2
[('A+', 10), ('C#', 35000), ('Python', 35000), ('C++', 75000), ('JavaScript', 75000)]

Задание B3
[('C#', '.NET core'), ('A+', 'A+ PROGRAMMING IDE'), ('C#', 'ASP.NET'), ('JavaScript', 'Angular'), ('C++', 'CLion'), ('Python', 'Django'), ('C#', 'Entity Framework'),
```