Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

ЛАБОРАТОРНА РОБОТА №2

із дисципліни

“Бази даних та засоби управління”

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав
студент ІІІ курсу
ФПМ групи КВ-03
Недашківський Д. О.

Перевірив(ла)

Київ – 2022

Завдання

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

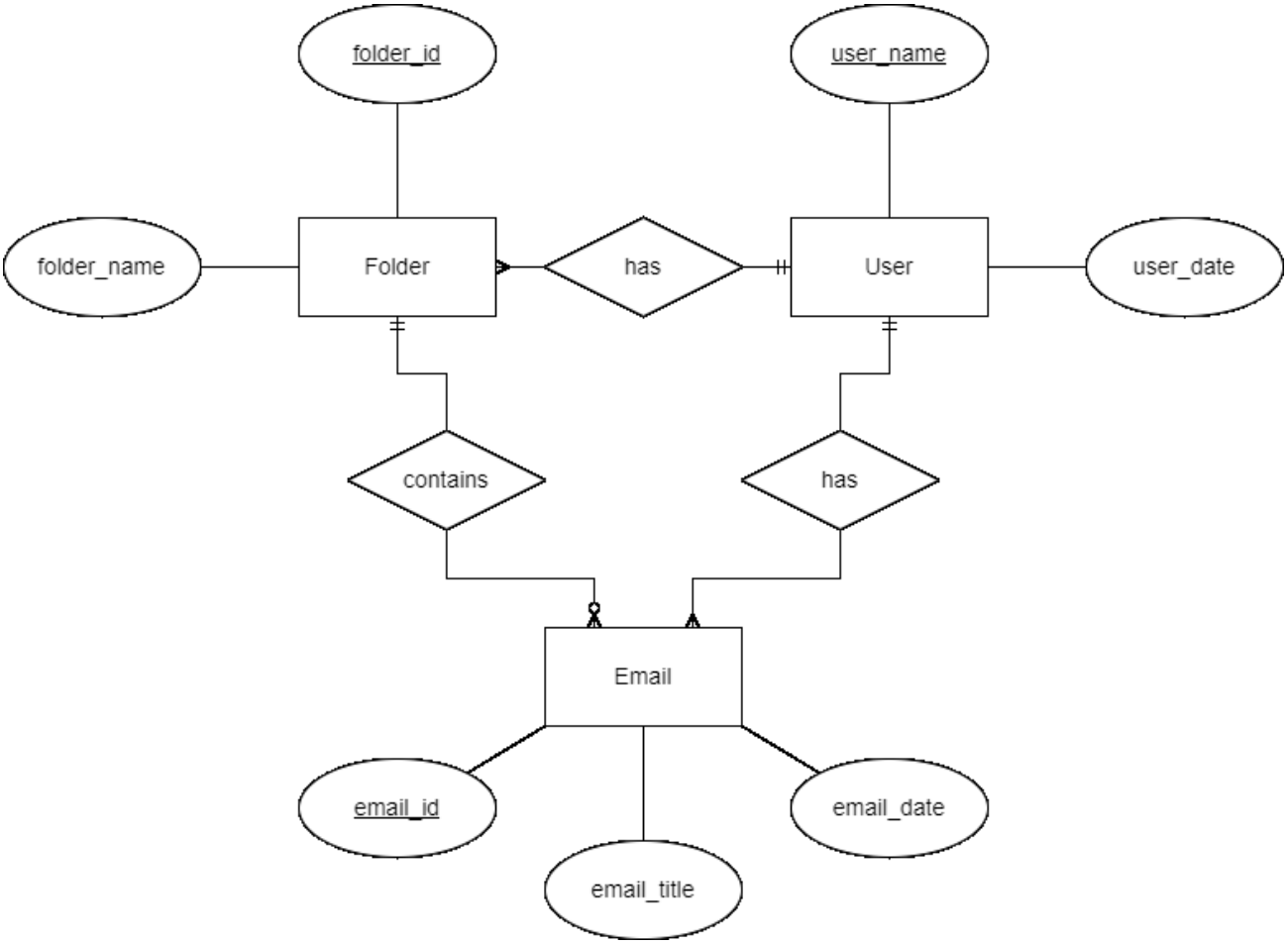
Деталізоване завдання

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв'язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом**!
3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Хід роботи

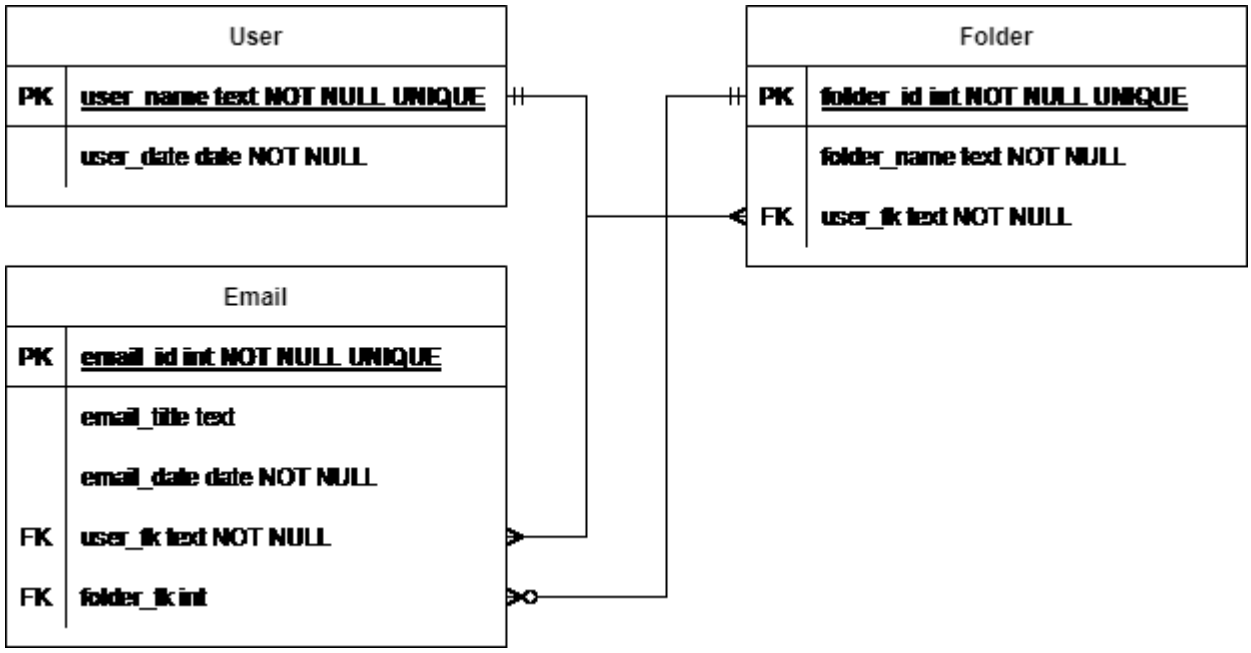
Діаграма ER-моделі умовної електронної пошти:



Наявні сутності:

- Сутність User – описує користувачів електронної пошти. Кожен користувач має унікальне ім’я та дату створення облікового запису. Один користувач може мати багато папок та листів.
- Сутність Folder – описує папки, які мають користувачі та містять листи. Кожна папка має ім’я та унікальний ID-номер. Одна папка може містити багато листів.
- Сутність Email – описує електронні листи, які мають користувачі, та які можуть лежати в папках. Кожен лист має заголовок, дату створення та унікальний ID-номер. Один лист знаходиться в одного користувача, і може знаходитися в папці.

ER-модель, перетворена в схему бази даних PostgreSQL для pgAdmin 4:



Середовище для відлагодження SQL-запитів до бази даних – PgAdmin 4;

Мова програмування – Python 3.9, використана бібліотека psycopg2;

Середовище розробки – Visual Studio Code;

Схема меню програми:

```
1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit
```

Опис функціональності кожного пункту:

1. Показує вміст таблиці, обраної користувачем;
2. Показує вміст усіх наявних таблиць;
3. Виконує введення даних з клавіатури до одного рядка таблиці, обраної користувачем;
4. Виконує редагування одного рядка таблиці, обраної користувачем;
5. Виконує видалення одного рядка таблиці, обраної користувачем;
6. Виконує пошук рядків із двох таблиць у трьох можливих конфігураціях;
7. Виконує введення згенерованих псевдовипадкових даних до таблиці;

Завдання №1

Операції вилучення та виведення

```
1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input: 5

        1 - Email
        2 - Folder
        3 - User

Choose a table: 3
Input the name/ID of the entry that you wish to delete: XE
Entry successfully deleted.

1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input: 1

        1 - Email
        2 - Folder
        3 - User

Choose a table: 2
===== TABLE Folder =====
* {'folder_name': 'TV', 'folder_id': 3, 'user_fk': 'MF'}
* {'folder_name': 'MN', 'folder_id': 4, 'user_fk': 'MF'}
* {'folder_name': 'LJ', 'folder_id': 5, 'user_fk': 'MF'}
* {'folder_name': 'SM', 'folder_id': 6, 'user_fk': 'MF'}
* {'folder_name': 'BV', 'folder_id': 7, 'user_fk': 'MF'}
* {'folder_name': 'VL', 'folder_id': 8, 'user_fk': 'MF'}
* {'folder_name': 'II', 'folder_id': 9, 'user_fk': 'MF'}
* {'folder_name': 'KE', 'folder_id': 10, 'user_fk': 'MF'}
* {'folder_name': 'EK', 'folder_id': 11, 'user_fk': 'MF'}
* {'folder_name': 'HU', 'folder_id': 12, 'user_fk': 'MF'}
```

Операція вставки

```
1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input: 3

        1 - Email
        2 - Folder
        3 - User

Choose a table: 1
Input data that you wish to insert:
ID = 17
Title = about gaming
Arrival Date = 22-10-1999
Assoc. Folder = MN
Assoc. User =
Invalid data - couldn't create entry
```

Завдання №2

Генерація псевдовипадкових даних

```
1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input: 7

        1 - Email
        2 - Folder
        3 - User

Choose a table: 1
Input number of rows to be generated: 200

1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input:
```

```
* {'email_id': 187, 'email_title': 'VU', 'email_date': datetime.date(2014, 1, 11), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 188, 'email_title': 'JH', 'email_date': datetime.date(2014, 1, 12), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 189, 'email_title': 'RS', 'email_date': datetime.date(2014, 1, 13), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 190, 'email_title': 'PG', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 191, 'email_title': 'WR', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 192, 'email_title': 'KM', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 193, 'email_title': 'RI', 'email_date': datetime.date(2014, 1, 15), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 194, 'email_title': 'TS', 'email_date': datetime.date(2014, 1, 12), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 195, 'email_title': 'MD', 'email_date': datetime.date(2014, 1, 20), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 196, 'email_title': 'TN', 'email_date': datetime.date(2014, 1, 18), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 197, 'email_title': 'KL', 'email_date': datetime.date(2014, 1, 10), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 198, 'email_title': 'MJ', 'email_date': datetime.date(2014, 1, 16), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 199, 'email_title': 'EB', 'email_date': datetime.date(2014, 1, 15), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 200, 'email_title': 'HQ', 'email_date': datetime.date(2014, 1, 12), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 201, 'email_title': 'FR', 'email_date': datetime.date(2014, 1, 11), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 202, 'email_title': 'SW', 'email_date': datetime.date(2014, 1, 10), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 203, 'email_title': 'PQ', 'email_date': datetime.date(2014, 1, 20), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 204, 'email_title': 'AV', 'email_date': datetime.date(2014, 1, 20), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 205, 'email_title': 'UL', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 206, 'email_title': 'AV', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 207, 'email_title': 'FW', 'email_date': datetime.date(2014, 1, 15), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 208, 'email_title': 'JA', 'email_date': datetime.date(2014, 1, 13), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 209, 'email_title': 'LE', 'email_date': datetime.date(2014, 1, 19), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 210, 'email_title': 'JF', 'email_date': datetime.date(2014, 1, 20), 'folder_fk': 3, 'user_fk': 'UP'}
* {'email_id': 211, 'email_title': 'AU', 'email_date': datetime.date(2014, 1, 14), 'folder_fk': 3, 'user_fk': 'UP'}
```

SQL-запити

```
INSERT INTO public."Email" (email_id,email_title,email_date,folder_fk,user_fk)
SELECT nextval('email_seq'),
chr(trunc(65+random()*25)::int)||chr(trunc(65+random()*25)::int),
timestamp '2014-01-10 20:00:00'+random()*(timestamp '2014-01-20 20:00:00'-
timestamp '2014-01-10 10:00:00'),
(SELECT * FROM (SELECT folder_id FROM public."Folder" ORDER BY random()) as foo
LIMIT 1) as bar,
(SELECT * FROM (SELECT user_name FROM public."User" ORDER BY random()) as baz
LIMIT 1) as qux
FROM generate_series(1,{})

INSERT INTO public."Folder" (folder_name,folder_id,user_fk)
SELECT chr(trunc(65+random()*25)::int)||chr(trunc(65+random()*25)::int),
nextval('folder_seq'),
(SELECT * FROM (SELECT user_name FROM public."User" ORDER BY random()) as foo
LIMIT 1) as bar
FROM generate_series(1,{})

INSERT INTO public."User" (user_name,user_date)
SELECT chr(trunc(65+random()*25)::int)||chr(trunc(65+random()*25)::int),
timestamp '2014-01-10 20:00:00'+random()*(timestamp '2014-01-20 20:00:00'-
timestamp '2014-01-10 10:00:00')
FROM generate_series(1,{})
```

Завдання №3

Пошук даних

```
1 - Show contents of a table
2 - Show contents of all tables
3 - Insert record
4 - Modify record
5 - Delete record
6 - Find record
7 - Randomize data
8 - Exit

Input: 6

Choose the relation:
1 - Email by User
2 - Folder by User
3 - Email by Folder

Input: 1
Input the username/folder name related to the entry that you wish to find: U%
Time elapsed searching: 3 ms
Entries found:
* ('MF', datetime.date(2014, 1, 14), 'UP')
* ('LH', datetime.date(2014, 1, 11), 'UP')
* ('WA', datetime.date(2014, 1, 19), 'UP')
* ('VT', datetime.date(2014, 1, 15), 'UP')
* ('BE', datetime.date(2014, 1, 20), 'UP')
* ('RI', datetime.date(2014, 1, 15), 'UP')
* ('AJ', datetime.date(2014, 1, 19), 'UP')
* ('LE', datetime.date(2014, 1, 19), 'UP')
```

SQL-запити

```
SELECT email_title,email_date,user_name FROM
(SELECT L.email_title,L.email_date,R.user_name FROM
public."Email" L LEFT JOIN public."User" R on L.user_fk=R.user_name
WHERE R.user_name LIKE '{}' GROUP BY L.email_title,L.email_date,R.user_name) as
foo

SELECT folder_name,folder_id,user_name FROM
(SELECT L.folder_name,L.folder_id,R.user_name FROM
public."Folder" L LEFT JOIN public."User" R on L.user_fk=R.user_name
WHERE R.user_name LIKE '{}' GROUP BY L.folder_name,L.folder_id,R.user_name) as
foo

SELECT email_title,email_date,folder_name FROM
(SELECT L.email_title,L.email_date,R.folder_name FROM
public."Email" L LEFT JOIN public."Folder" R on L.folder_fk=R.folder_id
WHERE R.folder_name LIKE '{}' GROUP BY L.email_title,L.email_date,R.folder_name)
as foo
```

Завдання №4

Код

```
import backend

class Model():
    def __init__(self, table_type):
        self._conn = backend.connect_db()
        self._table_type = table_type
        self._primkey = backend.primkeys[table_type]

    def __del__(self):
        self._conn.commit()
        self._conn.close()

    @property
    def connection(self):
        return self._conn

    @property
    def table_type(self):
        return self._table_type

    def table_type(self, new_table_type):
        if new_table_type not in backend.tables:
            return True
        self._table_type = new_table_type
        return False

    def create_entry(self, args):
        backend.insert_one(self._conn, self._table_type, args)

    def read_entry(self, item):
        return backend.select_one(self._conn, self._table_type, self._primkey,
item)

    def read_entries(self):
        return backend.select_all(self._conn, self._table_type)

    def update_entry(self, item, args):
        backend.update_one(self._conn, self._table_type, item, args)

    def delete_entry(self, item):
        backend.delete_one(self._conn, self._table_type, item)

    def find_entries(self, item, rel):
        return backend.find(self._conn, item, rel)

    def randomize(self, n):
        backend.randomize(self._conn, self._table_type, n)
```

Модуль “Model” відповідає за:

- створення зв’язку із базою даних, та його закриття під час завершення роботи;

- виклик відповідних функцій модуля backend, що безпосередньо формує запит та вносить зміни до бази даних;
- перевірку на відповідність введеного користувачем номеру таблиці списку наявних таблиць;