НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра програмного забезпечення та комп'ютерних систем

Лабораторна робота №2

з дисципліни

«Бази даних і засоби управління»

Виконав: студент III курсу
ФПМ групи КП-73
Литвиненко Антон
Перевірив(ла):

Київ – 2019

Ознайомлення з базовими операціями СУБД PostgreSQL

Мета роботи: здобуття практичних навичок проектування та побудови реляційних баз даних та створення прикладних програм з базами даних

Завдання роботи полягає у наступному:

1. Виконати нормалізацію бази даних, яка була створена у лабораторній роботі №1, до третьої нормальної форми (3НФ);

2. Реалізувати функціональні вимоги, наведені нижче.

Функціональні вимоги:

1. Реалізувати внесення, редагування та вилучення даних у базі засобами консольного інтерфейсу;

2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі;

3. Забезпечити реалізацію пошуку за двома-трьома атрибутами з двох сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як перелічення, для логічного типу – значення True/False, для дат – у рамках діапазону дат;

4. Забезпечити реалізацію повнотекстового пошуку за будь-яким текстовим атрибутом бази даних засобами PostgreSQL з виділенням знайденого фрагменту.

Вимоги до інтерфейсу користувача:

1. Використовувати консольний інтерфейс користувача.

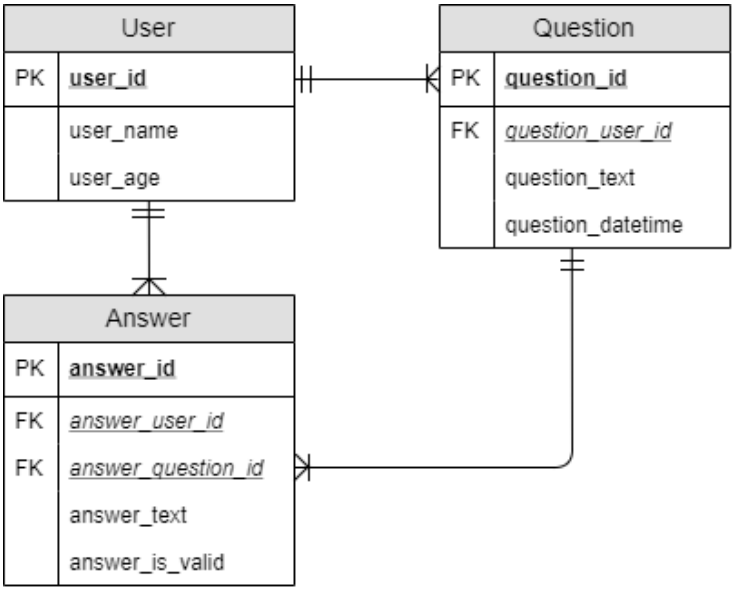
Варіант 13

Пошук за атрибутами має відбуватися по діапазону дат та логічному типу.

Повнотекстовий пошук зі «слово не входить» та за цілою фразою.

Нормалізована модель даних

(була у 3НФ ще у лаб1)



Таблиця User знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця User знаходиться у 2НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа user_id. Таблиця User знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Таблиця Question знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Question знаходиться у 2НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа question_id. Таблиця Question знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Таблиця Answer знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Answer знаходиться у 2 НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа answer_id. Таблиця Answer знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Опис програми

Програма створена за патерном MVC (Model-View-Controller). Складається відповідно з модулів model , view та controller.

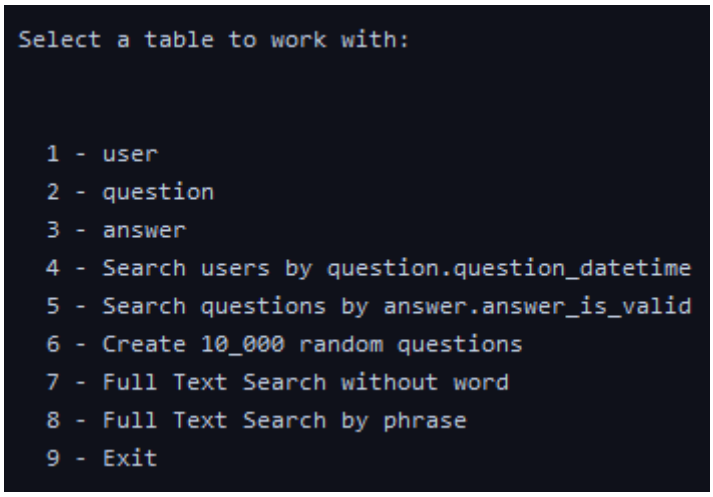
У модулі model реалізовані функції , що здійснюють SQL запити до Бази Даних.

У модулі view реалізовані функції виводу даних з таблиць та зчитування даних з клавіатури.

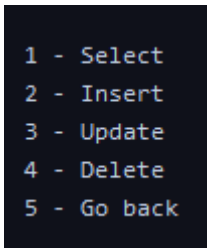
У модулі controller реалізовані функції для відповідних меню та допоміжні функції.

Опис структури меню програми

Меню програми можна розглядати як її концептуальну модель



Пункти 1-3:



Пункт 4-5: пошук по діапазону дат та по логічному типу відповідно.

Пункт 6: створення 10 000 випадкових Запитань.

Пункт 7: повнотексовий пошук без входження слова

Пункт 8: повнотекстовий пошук за фразою

Лістинг програми

```
from controller import display_main_menu

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

Controller

```
from consolemenu import SelectionMenu

import model
import view


def display_main_menu(err='', table=None):
    tables = list(model.TABLES.keys())

    menu = SelectionMenu(tables + ['Search users by question.question_datetime',
                                   'Search questions by answer.answer_is_valid',
                                   'Create 10_000 random questions',
                                   'Full Text Search without word',
                                   'Full Text Search by phrase'], subtitle=err,
                         title="Select a table to work with:")
    menu.show()

    index = menu.selected_option
    if index < len(tables):
        table = tables[index]
        display_secondary_menu(table)
    elif index == len(tables):
        search_users_by_question_date()
    elif index == len(tables) + 1:
        search_questions_by_answer_is_valid()
    elif index == len(tables) + 2:
        create_random_questions()
    elif index == len(tables) + 3:
        full_text_search_without_word()
    elif index == len(tables) + 4:
        full_text_search_by_phrase()
    else:
        print('Bye! Have a nice day!')


def display_secondary_menu(table, subtitle=''):
    opts = ['Select', 'Insert', 'Update', 'Delete']
    steps = [select, insert, update, delete, display_main_menu]

    menu = SelectionMenu(
        opts, subtitle=subtitle,
        title=f'Selected table "{table}"', exit_option_text='Go back',)
    menu.show()
    index = menu.selected_option
    steps[index](table=table)


def select(table):
    query = view.multiple_input(table, 'Enter requested fields:')
    data = model.get(table, query)
    view.print_entities(table, data)
    view.press_enter()
    display_secondary_menu(table)
```

```
def insert(table):
    data = view.multiple_input(table, 'Enter new fields values:')
    model.insert(table, data)
    display_secondary_menu(table, 'Insertion was made successfully')


def update(table):
    condition = view.single_input(
        table, 'Enter requirement of row to be changed:')
    query = view.multiple_input(table, 'Enter new fields values:')

    model.update(table, condition, query)
    display_secondary_menu(table, 'Update was made successfully')


def delete(table):
    query = view.multiple_input(
        table, 'Enter requirement of row to be deleted:')

    model.delete(table, query)
    display_secondary_menu(table, 'Deletion was made successfully')


def search_users_by_question_date():
    dates = view.specified_input(msg='Enter datetime range divided in format
<dd/mm/yyyy hh:mm:ss>-<dd/mm/yyyy hh:mm:ss>:').split('-')
    data = model.search_users_by_question_date(dates)
    view.print_entities(f'User who wrote questions in date range={dates}', data)
    view.press_enter()
    display_main_menu()


def search_questions_by_answer_is_valid():
    query = view.specified_input(
        'is_admin', 'Enter answer.is_valid value:'
    ).lower() in ['true', 't', 'yes', 'y', '+']

    data = model.search_questions_by_answer_is_valid(query)
    view.print_entities(
        f'Questions which where answered with answer.is_valid={query}:', data)
    view.press_enter()
    display_main_menu()


def full_text_search_without_word():
    query = view.specified_input(
        'word', 'Enter your word to be absent in the document:')
    data = model.full_text_search_without_word(query)
    view.print_entities(
        f'Documents without word=`{query}`', data)
    view.press_enter()
    display_main_menu()


def full_text_search_by_phrase():
    query = view.specified_input(
        'phrase', 'Enter your phrase to be found in document:')
    data = model.full_text_search_by_phrase(query)
    view.print_entities(
        f'Documents with phrase=`{query}`', data)
    view.press_enter()
```

```
    display_main_menu()


def create_random_questions():
    model.create_random_questions()
    display_main_menu('10_000 random questions were successfully added')
```

View

```
import model

COLUMN_WIDTH = 25


def print_entities(table, data):
    entities, cols = data

    separator_line = '-' * COLUMN_WIDTH * len(cols)

    print(f'Working with table "{table}"', end='\n\n')
    print(separator_line)
    print(''.join([f'{col}     |'.rjust(COLUMN_WIDTH, ' ') for col in cols]))
    print(separator_line)

    for entity in entities:
        print(''.join([f'{col}     |'.rjust(COLUMN_WIDTH, ' ') for col in entity]))
    print(separator_line)


def specified_input(colname=None, msg=None):
    if msg:
        print(msg)
    if colname:
        print(f'{colname}=', end='')
    return input()


def single_input(tname, msg):
    print(msg)
    print('(use format <attribute>=<value>)')
    print(f'({"/".join(model.TABLES[tname])})', end='\n\n')

    while True:
        data = input()
        if not data or data.count('=') != 1:
            print('Invalid input, try one more time')
            continue

        data = data.split('=')
        col, val = data[0].strip(), data[1].strip()
        if col.lower() in [tcol.lower() for tcol in model.TABLES[tname]]:
            return col, val
        else:
            print(f'Invalid column name "{col}" for table "{tname}"')


def multiple_input(tname, msg):
    print(msg)
    print('(use format <attribute>=<value>)')
    print(f'({"/".join(model.TABLES[tname])})', end='\n\n')
```

```
    res = {}
    while True:
        data = input()
        if not data:
            break
        if data.count('=') != 1:
            print('Invalid input')
            continue

        data = data.split('=')
        col, val = data[0].strip(), data[1].strip()
        if col.lower() in [tcol.lower() for tcol in model.TABLES[tname]]:
            res[col] = val
        else:
            print(f'Invalid column name "{col}" for table "{tname}"')

    return res


def multiline_input(msg):
    print(msg, end='\n\n')

    lines = []
    while True:
        line = input()
        if not line:
            break
        lines.append(line)

    return '\n'.join(lines)


def press_enter():
    input()
```

Model

```
import psycopg2

conn = psycopg2.connect("dbname='labs' user='anton'"
                        "host='localhost' password='password'")
cursor = conn.cursor()

TABLES = {
    'user': ('user_id', 'user_name', 'user_age'),
    'question': ('question_id', 'question_text', 'question_datetime',
'question_user_id'),
    'answer': ('answer_id', 'answer_text', 'answer_is_valid', 'answer_user_id',
'answer_question_id')
}


def insert(table_name, opts):
    try:
        cols = opts.keys()
        vals = [f"'{val}'" for val in opts.values()]
        command = f'insert into "{table_name}" ({", ".join(cols)}) ' + \
            f'values ({", ".join(vals)})'
        cursor.execute(command)
    finally:
        conn.commit()
```

```python
def get(table_name, opts=None):
    command = f'select * from "{table_name}"'

    if opts:
        conditions = [f"{col}='{opts[col]}'" for col in opts]
        command = f'{command} where {" and ".join(conditions)}'

    cursor.execute(command)
    return cursor.fetchall(), TABLES[table_name]


def update(table_name, condition, opts):
    try:
        column, value = condition
        updates = ', '.join([f"{col} = '{opts[col]}'" for col in opts])
        command = f'update "{table_name}" set {updates} where {column}=\'{value}\''

        cursor.execute(command)
        conn.commit()
    finally:
        conn.commit()


def delete(table_name, opts):
    try:
        conditions = [f"{col}='{opts[col]}'" for col in opts]
        command = f'delete from "{table_name}" where {" and ".join(conditions)}'
        cursor.execute(command)
    finally:
        conn.commit()


def search_questions_by_answer_is_valid(is_valid):
    command = f'''
    select question_id, question_text, question_datetime, question_user_id from
question
    join answer a on question.question_id = a.answer_question_id
    where answer_is_valid={is_valid}
    group by question_id;'''

    cursor.execute(command)
    return cursor.fetchall(), TABLES['question']


def search_users_by_question_date(dates):
    start, end = dates
    command = f"""
    select "user".user_id, user_name, user_age from "user"
    join question q on "user".user_id = q.question_user_id
    where question_datetime between '{start}' and '{end}'
    group by user_id;"""

    cursor.execute(command)
    return cursor.fetchall(), TABLES['user']


def full_text_search_without_word(word):
    command = f"""
    select question_text, answer_text, answer_is_valid from question
    join answer a on question.question_id = a.answer_question_id
```

```
    where to_tsvector(question_text) || to_tsvector(answer_text) @@
to_tsquery('!{word}');"""
    cursor.execute(command)
    return cursor.fetchall(), ('question_text', 'answer_text')


def full_text_search_by_phrase(phrase):
    command = f"""
    select ts_headline(question_text, phraseto_tsquery('{phrase}'),
'StartSel=\033[91m, StopSel=\033[0m'),
          ts_headline(answer_text, phraseto_tsquery('{phrase}'), 'StartSel=\033[91m,
StopSel=\033[0m')
    from question join answer a on question.question_id = a.answer_question_id
    where to_tsvector(question_text) || to_tsvector(answer_text) @@
phraseto_tsquery('{phrase}');"""
    cursor.execute(command)
    return cursor.fetchall(), ('question_text', 'answer_text')


def create_random_questions():
    try:
        with open('sql/random.sql', 'r') as file:
            sql = file.read()
            cursor.execute(sql)
    finally:
        conn.commit()
```

Скріншот результатів виконання операції вилучення

Спроба вилучення Користувача на який є посилання

```
Working with table "user"

-------------------------------------------------------------------------
            user_id     |          user_name     |           user_age     |
-------------------------------------------------------------------------
                  1     |             lolkek     |                  20    |
                  2     |               user     |                  30    |
                  3     |           user1234     |                  15    |
                  4     |           nagibator    |                  20    |
                  5     |              pro100    |                  10    |
                  6     |              sample    |                  11    |
                  7     |                   1    |                None    |
                  8     |                  22    |                None    |
-------------------------------------------------------------------------
```

```
Working with table "answer"

-----------------------------------------------------------------------------------------------------------------
        answer_id     |       answer_text     |    answer_is_valid     |     answer_user_id     | answer_question_id     |
-----------------------------------------------------------------------------------------------------------------
                1     |             Human     |             True     |                5     |                1     |
                2     |                No     |            False     |                4     |                2     |
                3     |               Yep     |             True     |                1     |                2     |
                4     |              Sure     |             True     |                1     |                3     |
                5     |           I`m fine    |            False     |                5     |                4     |
                6     |         I feel sad    |             True     |                6     |                4     |
-----------------------------------------------------------------------------------------------------------------
```

Видаляємо користувача на якого є посилання:

```
Enter requirement of row to be deleted:
(use format <attribute>=<value>)
(user_id/user_name/user_age)

user_id=1
```

```
psycopg2.errors.ForeignKeyViolation: update or delete on table "user" violates foreign key
constraint "question_question_user_id_fkey" on table "question"
DETAIL:  Key (user_id)=(1) is still referenced from table "question".
```

Вдале видалення і демонстрація результату вилучення

```
Enter requirement of row to be deleted:
(use format <attribute>=<value>)
(user_id/user_name/user_age)

user_id=8
```

```
Working with table "user"

-------------------------------------------------------------------------
           user_id     |          user_name     |           user_age     |
-------------------------------------------------------------------------
                 1     |             lolkek     |                 20     |
                 2     |               user     |                 30     |
                 3     |           user1234     |                 15     |
                 4     |          nagibator     |                 20     |
                 5     |             pro100     |                 10     |
                 6     |             sample     |                 11     |
                 7     |                  1     |               None     |
-------------------------------------------------------------------------
```