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
Булаєвський Ігор Олегович
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:

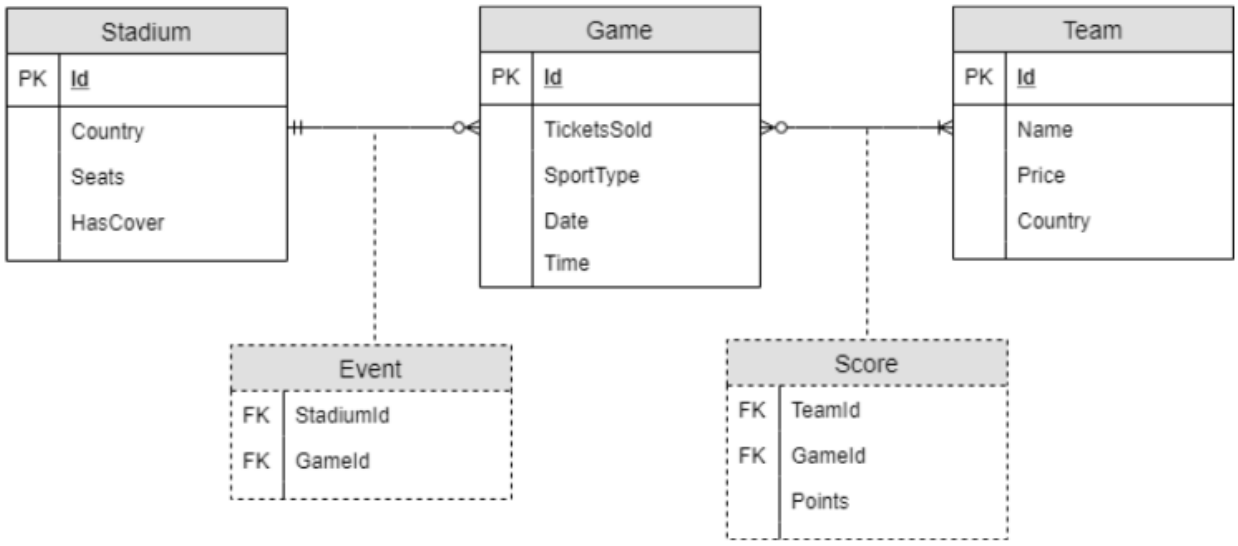
1. Використовувати консольний інтерфейс користувача.

Варіант 3

Пошук за атрибутами має відбуватися по переліченню строк та логічному типу.

Повнотекстовий пошук з обов’язковим входженням цілого слова або для випадку, коли вказане слово не входить в документ.

Нормалізована модель даних



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

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

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

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

Таблиця Score знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Score знаходиться у 2НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинних ключів GameId, TeamId. Таблиця Score 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.

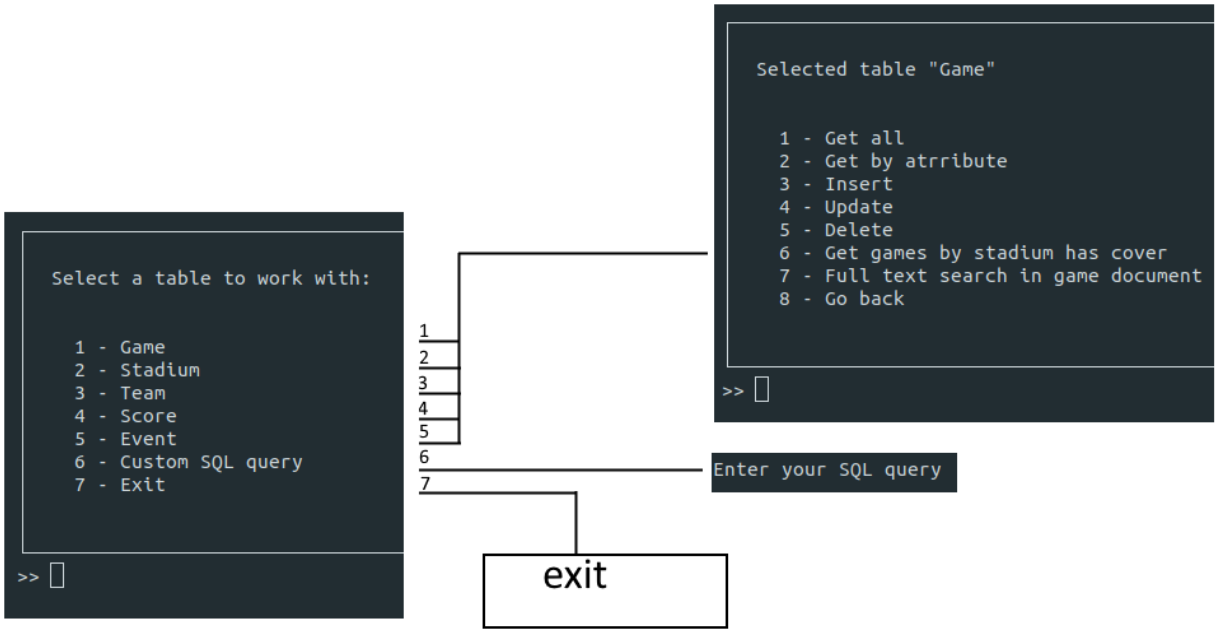
У модулі model реалізовані функції , що здійснюють SQL запити до Бази Даних.

У модулі view реалізовані функції виводу даних з таблиць.

У класі Controller реалізовані функції для відповідних меню та допоміжні функції.

Опис структури меню програми

Меню програми можна розглядати як її концептуальну модель



Лістинг програми

```
if __name__ == '__main__':
    import controller
    controller.show_start_menu()
```

Controller

```
from consolemenu import SelectionMenu

import model
import view
import reader


def handle_error(func):
    def wrapper(tname):
        try:
            func(tname)
        except Exception as e:
            show_table_menu(tname, str(e))
    return wrapper


def show_start_menu(tname='', err=''):
    tables = list(model.TABLES.keys())

    menu = SelectionMenu(tables + ['Custom SQL query'], subtitle=err,
                         title="Select a table to work with:")
    menu.show()

    index = menu.selected_option
    if index < len(tables):
        tname = tables[index]
        show_table_menu(tname)
    elif index == len(tables):
        custom_query()
    else:
        print('Bye! Have a nice day!')


def show_table_menu(tname, subtitle=''):
    opts = ['Get all', 'Get by atrribute', 'Insert', 'Update', 'Delete']
    steps = [get_all, get_by_attr, insert,
             update, delete]

    if tname == 'Game':
        opts += ['Get games by stadium has cover',
                 'Full text search in game document']
        steps += [get_games_hascover, fts]
```

```
    elif tname == 'Team':
        opts += ['Get teams by sport type', 'Create 10_000 random teams']
        steps += [get_team_sporttype, create_random_team]
    steps += [show_start_menu]

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


@handle_error
def get_all(tname):
    data = model.get(tname)
    view.print_entities(tname, data)
    reader.press_enter()
    show_table_menu(tname)


@handle_error
def get_by_attr(tname):
    query = reader.multiple_input(tname, 'Enter requested fields:')
    data = model.get(tname, query)
    view.print_entities(tname, data)
    reader.press_enter()
    show_table_menu(tname)


@handle_error
def insert(tname):
    data = reader.multiple_input(tname, 'Enter new fields values:')
    model.insert(tname, data)
    show_table_menu(tname, 'Insertion was made successfully')


@handle_error
def update(tname):
    condition = reader.single_input(
        tname, 'Enter requirement of row to be changed:')
    query = reader.multiple_input(tname, 'Enter new fields values:')

    model.update(tname, condition, query)
    show_table_menu(tname, 'Update was made successfully')


@handle_error
def delete(tname):
    query = reader.multiple_input(
        tname, 'Enter requirement of row to be deleted:')
```

```
    model.delete(tname, query)
    show_table_menu(tname, 'Deletion was made successfully')


@handle_error
def get_games_hascover(tname):
    query = reader.specified_input(
        'hasCover', 'Enter hasCover value:'
    ).lower() in ['true', 't', 'yes', 'y', '+']
    data = model.get_games_by_stadium_hascover(query)
    view.print_entities(f'Games with hasCover={query}', data)
    reader.press_enter()
    show_table_menu(tname)


@handle_error
def get_team_sporttype(tname):
    types = []
    query = reader.specified_input(
        msg='Enter your queries for sportype:').lower()
    while query:
        types.append(query)
        query = reader.specified_input().lower()

    data = model.get_teams_by_sporttype(types)
    view.print_entities(
        f'Teams which played at least one of this games: {types}', data)
    reader.press_enter()
    show_table_menu(tname)


@handle_error
def fts(tname):
    query = reader.specified_input(
        'query', 'Enter your query to search in document:')
    contains = reader.specified_input(
        msg='Query word shoul be in document?'
    ).lower() in ['true', 't', 'yes', 'y', '+']
    data = model.fts(query, contains)
    view.print_entities(
        f'Documents corresponding to query={query} ({"" if contains else "not "}c
ontains)', data)
    reader.press_enter()
    show_table_menu(tname)


@handle_error
def create_random_team(tname):
    model.create_random_teams()
    show_table_menu(tname, '10_000 random teams were successfully added')
```

```
def custom_query():
    try:
        sql = reader.multiline_input('Enter your SQL query')
        data = model.execute(sql)
        view.print_entities('Custom query result', data)
        reader.press_enter()
        show_start_menu()
    except Exception as e:
        show_start_menu(err=str(e))
```

View

```
COLUMN_WIDTH = 30


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

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

    print(f'Working with table "{tname}"', end='\n\n')
    print(separator_line)
    print(''.join([f'{col}     |'.rjust(30, ' ') for col in cols]))
    print(separator_line)

    for entity in entities:
        print(''.join([f'{col}     |'.rjust(30, ' ') for col in entity]))
    print(separator_line)
```

Model

```
import psycopg2

conn = psycopg2.connect("dbname='kpi' user='admin'"
                        "host='localhost' password='admin'")
cursor = conn.cursor()

TABLES = {
    'Game': ('Id', 'TicketSold', 'SportType', 'Date'),
    'Stadium': ('Id', 'Country', 'Seats', 'Hascover'),
    'Team': ('Id', 'Price', 'Country', 'Name'),
    'Score': ('GameId', 'TeamId', 'Points'),
    'Event': ('GameId', 'StadiumId')
}


def create_tables():
```

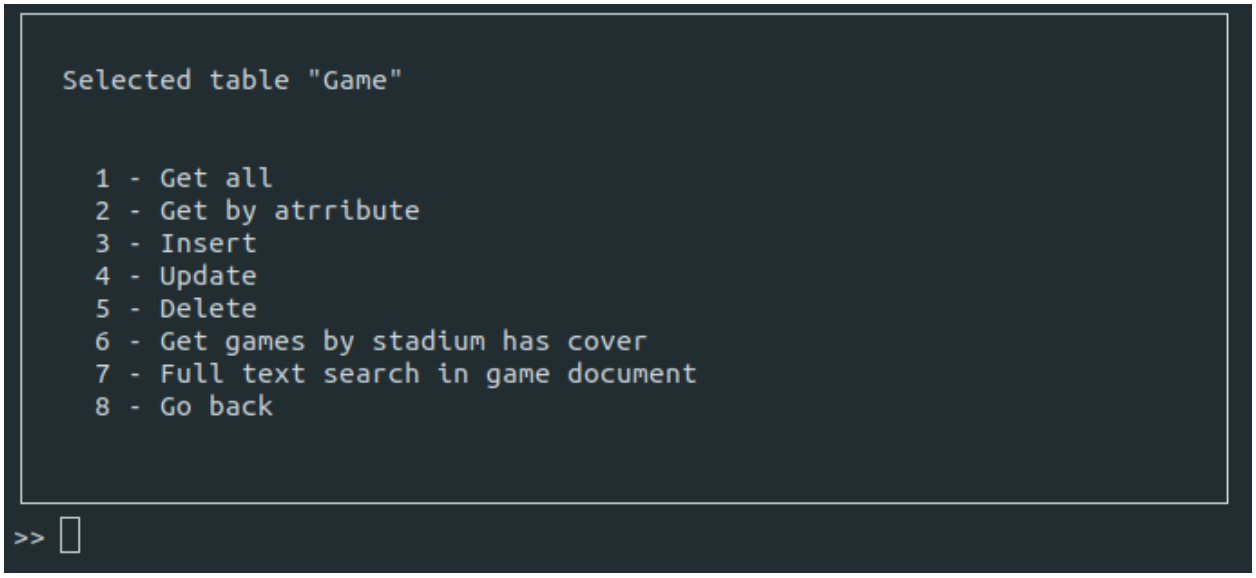
```
    with open('scripts/create.sql') as file:
        command = file.read()
        cursor.execute(command)
        conn.commit()


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


def get(tname, opts=None):
    comand = f'select * from {tname}'

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

    cursor.execute(comand)
    return cursor.fetchall(), TABLES[tname]


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

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


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


def get_teams_by_sporttype(sporttypes):
    sporttypes = [f"'{stype}'" for stype in sporttypes]
```

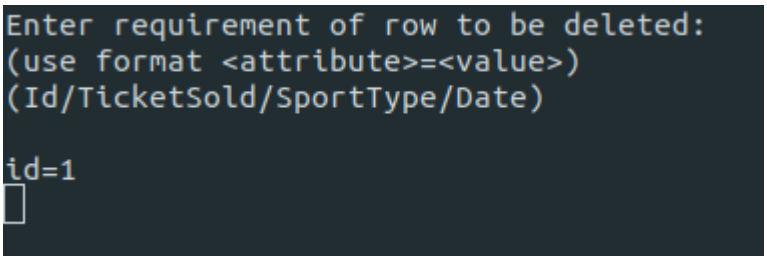
```
    comand = f'''
    select name from team
    join score on team.id=score.teamid
    join game on game.id=score.gameid
    where lower(sporttype) in ({", ".join(sporttypes)})'''

    cursor.execute(comand)
    return cursor.fetchall(), ('TeamName',)


def get_games_by_stadium_hascover(hascover):
    comand = f'''
    select * from game
    where id in (select gameid from event
    join stadium on stadium.id=event.stadiumid
    where hascover={hascover})'''

    cursor.execute(comand)
    return cursor.fetchall(), TABLES['Game']


def fts(query, contains):
    sql = f'''
    select gameid, name, sporttype from (
        select
            gameid,
            name,
            sporttype,
            to_tsvector(name) ||
            to_tsvector(sporttype) as document
        from score
        join game g on score.gameid = g.id
        join team t on score.teamid = t.id) search
    where search.document @@ to_tsquery('{'' if contains else '!'}{query}')'''
    cursor.execute(sql)
    return cursor.fetchall(), ('GameId', 'TeamName', 'SportType')


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


def execute(sql):
    try:
        cursor.execute(sql)
        return cursor.fetchall(), [desc[0] for desc in cursor.description]
```

```
    finally:
        conn.commit()
```

Скріншот результатів виконання операції вилучення

Спроба вилучення кортежу на який є посилання





Вдале видалення і деомонстрація результату вилучення

```
Working with table "Game"

-------------------------------------------------------------------------------------------------------------
|                  Id  |           TicketSold  |           SportType  |                  Date  |
-------------------------------------------------------------------------------------------------------------
|                   2  |                  850  |               Chess  |  2008-10-10 17:00:00  |
|                   3  |                   50  |             Restling  |  2005-09-09 16:00:00  |
|                   4  |                  999  |       Extrasensoric  |  2006-08-08 15:00:00  |
|                   5  |                  590  |              Racing  |  2012-07-07 14:00:00  |
|                   6  |                 1500  |              Diving  |  2015-08-08 13:00:00  |
|                   1  |                  222  |            Football  |  1999-12-12 18:00:00  |
|                  10  |                 1500  |             shashki  |                  None  |
-------------------------------------------------------------------------------------------------------------
```

```
Enter requirement of row to be deleted:
(use format <attribute>=<value>)
(Id/TicketSold/SportType/Date)

id=10
```

```
Working with table "Game"

-------------------------------------------------------------------------------------------------------------
|                  Id  |           TicketSold  |           SportType  |                  Date  |
-------------------------------------------------------------------------------------------------------------
|                   2  |                  850  |               Chess  |  2008-10-10 17:00:00  |
|                   3  |                   50  |             Restling  |  2005-09-09 16:00:00  |
|                   4  |                  999  |       Extrasensoric  |  2006-08-08 15:00:00  |
|                   5  |                  590  |              Racing  |  2012-07-07 14:00:00  |
|                   6  |                 1500  |              Diving  |  2015-08-08 13:00:00  |
|                   1  |                  222  |            Football  |  1999-12-12 18:00:00  |
-------------------------------------------------------------------------------------------------------------
```