

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

З дисципліни

**«Бази даних і засоби управління»**

Виконав:

Студент 3 курсу

ФПМ групи КВ-71

Бусь Б.Б.

Перевірив:

Київ – 2019

**Ознайомлення з базовими операціями СУБД PostgreSQL**

*Метою роботи* є здобуття практичних навичок проектування реляційних баз даних та створення прикладних програм щодо їх обробки.*Завдання роботи* полягає у наступному:

1. Виконати нормалізацію бази даних, що була створена у лабораторній роботі №1, до третьої нормальної форми.
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 базу.
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 заданого діапазону.
4. Забезпечити реалізацію повнотекстового пошуку за будь-яким текстовим атрибутом бази даних засобами PostgreSQL з виділенням знайденого фрагменту.

*Вимогу до інтерфейсу користувача:*

1. Використовувати консольний інтерфейс користувача.

*Вимоги до інструментарію*:

1. Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.
2. Мова програмування – Python 3.6-3.7.
3. Середовище розробки програмного забезпечення – PyCharm Community Edition 2019.

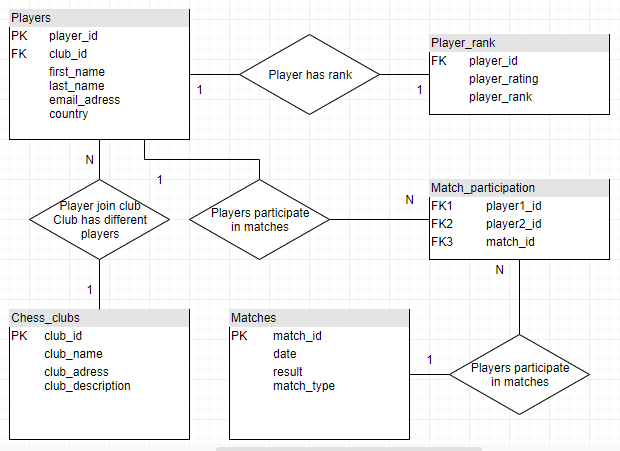
*Вимоги до структури програми:*

1. Програмний код організувати згідно з шаблоном MVC.
2. Для доступу до бази даних використовувати лише мову SQL.

*Варіант 1*:

|  |  |
| --- | --- |
| Пошук за атрибутами | Повнотекстовий пошук |
| діапазон чисел,  перелічення | обов’язкове входження  слова, ціла фраза |

**Нормалізація моделі даних**



В порівнянні з першою лабораторною роботою модель даних зазнала змін. Для забезпечення відповідності таблиць 3 нормальній формі із сутності Matches були видалені два атрибути player1\_id, player2\_id, що були зв’язуючими ідентифікаторами з гравцями, що уже і так є у сутності Matches\_participation.

Дана модель даних відповідає 1 нормальній формі, бо всі сутності мають скалярні атрибути.

Дана модель даних відповідає 2 нормальній формі, бо в усіх сутностях усі неключові атрибути залежать від первинних ключів повністю (тобто з первинного ключа не можна виділити таку підмножину, від якої можна вивести дану функціональну залежність).

Дана модель даних відповідає 3 нормальній формі, бо кожен неключовий атрибут кожної сутності залежить лише від первинного ключа, а не від іншого неключового атрибута.

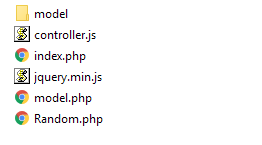
**Середовище розробки та налаштування підключення до бази даних**

Лабораторна робота виконувалась в середовищі розробки Sublime text 3 Мовою програмування була обрана php.

Для підключення до бази даних використовується бібліотека PDO, що є стандартною для php.

**Опис структури програми**

Програма реалізована за шаблоном Model-View-Controller (MVC).

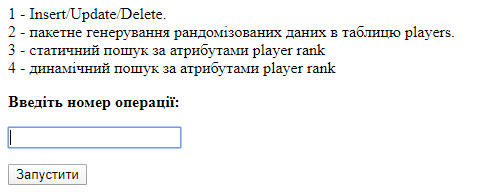


До складу компонента View входить файл index.php, що реалізує користувацький інтерфейс та виводить дані, що були вибрані з бази даних (SELECT).

До складу компонента Controller входить файл controller.js, що реалізує взаємозв’язок між Model та View. Тут знаходиться обробка вводу користувача ла логіка взаємодії між всіма компонентами.

До складу компонента Model входять файли model.php, random.php. model.php реалізує технічну частину взаємодії програми з базою даних (наприклад, підключення, надсилання запросів до бази даних і т.д.). Модуль random.php містить набір значень атрибутів, що можуть бути використані під час пакетної генерації рандомізованих даних. Модуль model.php працює з «логічним наповненням» бази даних. Цей модуль дозволяє створити об’єкт (що, по суті, є записом в таблиці) сутності, а також згенерувати відповідний SQL код для маніпуляції з новоствореним об’єктом та базою даних (наприклад, додати даний об’єкт до таблиці).

**Структура меню програми**



Пункт 1 – внесення, вилучення чи оновлення даних.

Пункт 2 – пакетне генерування рандомізованих даних в таблицю individual.

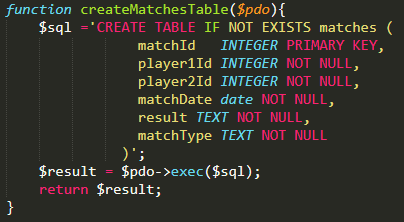
Пункт 3 – статичний пошук в 2 таблицях (individual, employment\_contract) одночасно по 2 атрибутам (individual.individual\_id, employment\_contract) за завчасно внесеними параметрами.

Пункт 4 – дивись пункт 3, але з відмінністю, що параметри пошуку вносяться користувачем через консольний інтерфейс.

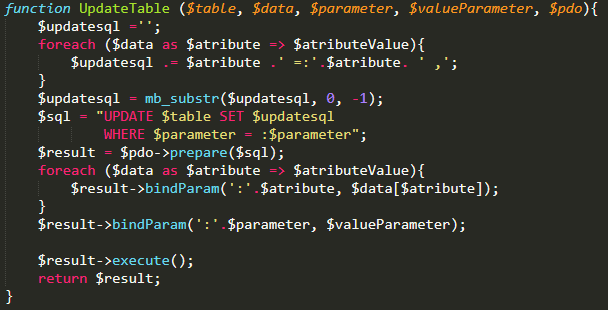
**Лістинги реалізації функції внесення, вилучення, оновлення та редагування даних у базі даних**

**Insert**

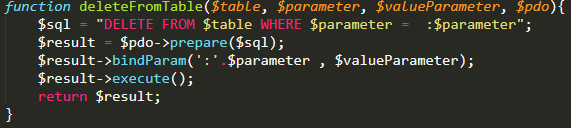




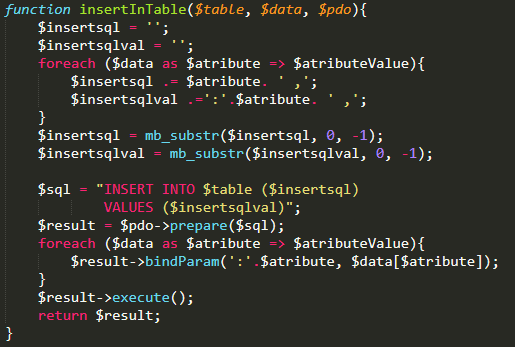
**Update**



**Delete**



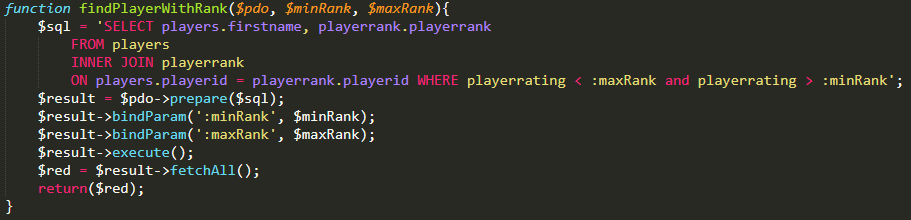
**Insert**



**Лістинги функцій з запитами на пошук**

Пошук виконується з 2 сутностей одночасно (individual, employment\_contract) та за 2 атрибутами (individual.male, employment\_contract.position).

**Статичний пошук**

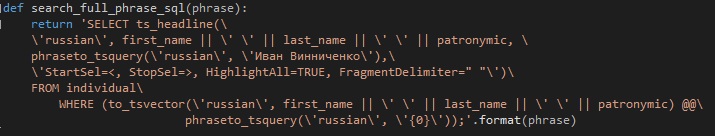
**Динамічний пошук**

Реалізація функції статичного пошуку не відрізняється від динамічного, окрім параметрів пошуку. Для статичного пошуку параметри задаються під час написання програми, а в динамічному вони вводяться користувачем.

**Лістинги функцій з повнотекстовим пошуком**

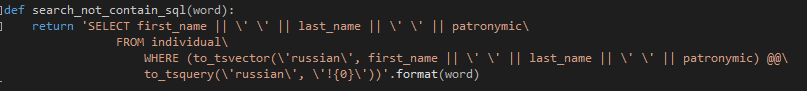
**Входить ціла фраза**

Цей запрос виводить рядок, в якому зустрілась фраза phrase, та виділяє її.

****

**Слово не входить**

Цей запрос виводить лише ті рядки, в яких дане слово не зустрічається.



**Лістинг компонента Controller**

**Controller.php**

function Operations(event){

event.preventDefault();

document.getElementById('menu').style.display = 'none';

var operation = document.getElementById("operation").value;

switch (operation) {

case '1':

document.getElementById('tableOperations').style.display = 'block';

break;

case "2":

$.post(

"Random.php",

onAjaxSuccess

);

function onAjaxSuccess(data)

{

showTable.insertAdjacentHTML('beforeend', data + '<br>');

showTable.insertAdjacentHTML('beforeend', '<button onclick="location.reload()">В главное меню</buttonon>' );

}

break;

case '3':

document.getElementById('ranks').style.display = 'block';

break;

}

}

function tableOperations(event){

event.preventDefault();

document.getElementById('tableOperations').style.display = 'none';

var table = document.getElementById("table").value;

var operation = document.getElementById("operationForTable").value;

var tables ={'1': 'Players', '2': 'Matches', '3': 'Player\_rank', '4': 'Chess\_club'}

var operations ={'1': 'Insert', '2': 'Update', '3': 'Delete', '4': 'Show'}

$.post(

"model.php",

{

table: tables[table],

operation: operations[operation]

},

onAjaxSuccess

);

function onAjaxSuccess(data)

{

data = JSON.parse(data);

data.forEach(function (element) {

for (var key in element) {

if(isNaN(key)){

showTable.insertAdjacentHTML('beforeend', key + ': '+ element[key] +'</br>' );

}

}

showTable.insertAdjacentHTML('beforeend', '</br>' );

})

showTable.insertAdjacentHTML('beforeend', '<button onclick="location.reload()">В главное меню</buttonon>' );

}

}

function FindWithRating(event){

event.preventDefault();

document.getElementById('ranks').style.display = 'none';

var minRating = document.getElementById("minRating").value;

var maxRating = document.getElementById("maxRating").value;

$.post(

"model.php",

{

operation: 'findDiap',

maxRank: minRating,

minRank: maxRating

},

onAjaxSuccess

);

function onAjaxSuccess(data)

{

data = JSON.parse(data);

data.forEach(function (element) {

for (var key in element) {

if(isNaN(key)){

showTable.insertAdjacentHTML('beforeend', key + ': '+ element[key] +'</br>' );

}

}

showTable.insertAdjacentHTML('beforeend', '</br>' );

})

showTable.insertAdjacentHTML('beforeend', '<button onclick="location.reload()">В главное меню</buttonon>' );

}

}

**Лістинг компонента View**

**Index.php**

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>DB</title>

<script src="jquery.min.js"></script>

<script src="controller.js"></script>

</head>

<body>

<div id = 'menu'>

1 - Insert/Update/Delete/Show.<br>

2 - пакетне генерування рандомізованих даних в таблицю players.<br>

3 - статичний пошук за атрибутами player rank<br>

4 - динамічний пошук за атрибутами player rank<br>

<form action="">

<p><strong>Введіть номер операції:</strong></p>

<p><input id='operation' ></p>

<button onclick="Operations(event)">Запустити</button>

</form>

</div>

<div id ='tableOperations' style="display: none;">

<form action="">

1 - players<br>

2 - matches<br>

3 - playerrank<br>

4 - chessclub<br>

<p><strong>Виберіть таблицю:</strong></p>

<p><input id='table' ></p>

1 - Insert<br>

2 - Update<br>

3 - Delete<br>

4 - Show<br>

<p><strong>Виберіть операцію:</strong></p>

<p><input id="operationForTable" ></p>

<button onclick="tableOperations(event)" id='submit'>Запустити</button>

</form>

</div>

<div id = "showTable">

</div>

<div id = "ranks" style="display: none;">

<p><strong>Видача гравців та їх звання по діапазону рейтингу:</strong></p>

<p>Mінімальний рейтинг:</p>

<p><input id="minRating" ></p>

<p>Максимальний рейтинг:</p>

<p><input id="maxRating"></p>

<button onclick="FindWithRating(event)" id='submit'>Запустити</button>

</div>

</body>

</html>

**Лістинг компонента Model**

**Model.php**

<?php

$host = 'localhost';

$port = '5432';

$dbname = 'chess\_db';

$user = 'postgres';

//create a pdo instance

$pdo = new PDO("pgsql:host=".$host.";dbname=".$dbname."", $user);

function createClubsTable($pdo){

$sql ='CREATE TABLE IF NOT EXISTS chessclubs (

clubId INTEGER PRIMARY KEY,

clubName TEXT NOT NULL,

clubAdress TEXT NOT NULL,

clubDescrption TEXT NOT NULL

)';

$result = $pdo->exec($sql);

return $result;

}

function createPlayersTable($pdo){

$sql ='CREATE TABLE IF NOT EXISTS players (

playerId INTEGER PRIMARY KEY ,

clubId INTEGER NOT NULL,

firstName TEXT NOT NULL,

lastName TEXT NOT NULL,

emailAdress TEXT NOT NULL,

country TEXT NOT NULL

)';

$result = $pdo->exec($sql);

return $result;

}

function createRankTable($pdo){

$sql ='CREATE TABLE IF NOT EXISTS playerRank (

playerId INTEGER PRIMARY KEY ,

playerRating INTEGER NOT NULL,

playerRank TEXT NOT NULL

)';

$result = $pdo->exec($sql);

return $result;

}

function createMatchesTable($pdo){

$sql ='CREATE TABLE IF NOT EXISTS matches (

matchId INTEGER PRIMARY KEY,

player1Id INTEGER NOT NULL,

player2Id INTEGER NOT NULL,

matchDate date NOT NULL,

result TEXT NOT NULL,

matchType TEXT NOT NULL

)';

$result = $pdo->exec($sql);

return $result;

}

function insertInTable($table, $data, $pdo){

$insertsql = '';

$insertsqlval = '';

foreach ($data as $atribute => $atributeValue){

$insertsql .= $atribute. ' ,';

$insertsqlval .=':'.$atribute. ' ,';

}

$insertsql = mb\_substr($insertsql, 0, -1);

$insertsqlval = mb\_substr($insertsqlval, 0, -1);

$sql = "INSERT INTO $table ($insertsql)

VALUES ($insertsqlval)";

$result = $pdo->prepare($sql);

foreach ($data as $atribute => $atributeValue){

$result->bindParam(':'.$atribute, $data[$atribute]);

}

$result->execute();

return $result;

}

function deleteFromTable($table, $parameter, $valueParameter, $pdo){

$sql = "DELETE FROM $table WHERE $parameter = :$parameter";

$result = $pdo->prepare($sql);

$result->bindParam(':'.$parameter , $valueParameter);

$result->execute();

return $result;

}

function UpdateTable ($table, $data, $parameter, $valueParameter, $pdo){

$updatesql ='';

foreach ($data as $atribute => $atributeValue){

$updatesql .= $atribute .' =:'.$atribute. ' ,';

}

$updatesql = mb\_substr($updatesql, 0, -1);

$sql = "UPDATE $table SET $updatesql

WHERE $parameter = :$parameter";

$result = $pdo->prepare($sql);

foreach ($data as $atribute => $atributeValue){

$result->bindParam(':'.$atribute, $data[$atribute]);

}

$result->bindParam(':'.$parameter, $valueParameter);

$result->execute();

return $result;

}

function SelectAllData($table, $pdo){

$sql = 'SELECT \* FROM players';

$result = $pdo->prepare($sql);

$result->execute();

$red = $result->fetchAll();

return ($red);

}

function findPlayerWithRank($pdo, $minRank, $maxRank){

$sql = 'SELECT players.firstname, playerrank.playerrank

FROM players

INNER JOIN playerrank

ON players.playerid = playerrank.playerid WHERE playerrating < :maxRank and playerrating > :minRank';

$result = $pdo->prepare($sql);

$result->bindParam(':minRank', $minRank);

$result->bindParam(':maxRank', $maxRank);

$result->execute();

$red = $result->fetchAll();

return($red);

}

if (isset($\_POST['operation'])){

switch ($\_POST['operation']){

case 'Insert':

$data= [

'playerid' => '12',

'clubid' => '33',

'firstname' => 'wq',

'lastname' => 'ccqc',

'emailadress' => 'cqwqcc',

'country' => 'ccc'

];

insertInTable('players', $data, $pdo);

break;

case 'Update':

$data= [

'playerid' => '12',

'clubid' => '33',

'firstname' => 'wq',

'lastname' => 'ccqc',

'emailadress' => 'cqwqcc',

];

UpdateTable ('players', $data, 'country', 'ccc', $pdo);

break;

case 'Delete':

deleteFromTable('players', 'country', 'ccc', $pdo);

break;

case 'Show':

$data = SelectAllData($\_POST['table'], $pdo);

echo(json\_encode($data));

break;

case 'findDiap':

$data = findPlayerWithRank($pdo, $\_POST['maxRank'],$\_POST['minRank'] );

echo(json\_encode($data));

break;

}

}

?>

**Random.php**

<?php

$host = 'localhost';

$port = '5432';

$dbname = 'chess\_db';

$user = 'postgres';

//create a pdo instance

$pdo = new PDO("pgsql:host=".$host.";dbname=".$dbname."", $user);

$firstname = ['Bodya', 'Stepa', 'Misha', 'Nina', 'Liza', 'Sanya', 'Vadim', 'Olya', 'Lilia', 'Ser\_gay', 'Donald'];

$lastname = ['Bodyanich', 'Stepanich', 'Mishin', 'Ninasiv', 'Lizin', 'Sanich', 'Vadimich', 'Olyasha', 'Lilianich', 'Ser\_gayevich', 'Donaldisimmo'];

$emailadress = ['Bodyanich@gmail.ru', 'Stepanich@gmail.ru', 'Mishin@gmail.ru', 'Ninasiv@gmail.ru', 'Lizin@gmail.ru', 'Sanich@gmail.ru', 'Vadimich@gmail.ru', 'Olyasha@gmail.ru', 'Lilianich@gmail.ru', 'Ser\_gayevich@gmail.ru', 'Donaldisimmo@gmail.ru'];

$country = ['ru', 'Ua', 'Uzb', 'Kz', 'Eng', 'Fr', 'Pl', 'Br', 'Italia', 'Esp', 'Col'];

$data= [

'playerid' => rand(1, 267),

'clubid' => rand(1, 267),

'firstname' => $firstname[rand(1, 10)],

'lastname' => $lastname[rand(1, 10)],

'emailadress' => $emailadress[rand(1, 10)],

'country' => $country[rand(1, 10)]

];

$sql = "INSERT INTO players (playerid, clubid, firstname, lastname, emailadress, country)

VALUES (:playerid, :clubid, :firstname, :lastname, :emailadress, :country)";

$result = $pdo->prepare($sql);

foreach ($data as $atribute => $atributeValue){

$result->bindParam(':'.$atribute, $data[$atribute]);

}

$result->execute();

if ($result)

echo ('Success');

else

echo ('ID уже занят');