УДК 004

Глинчук Людмила Ярославівна
кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук та кібербезпеки
Волинський національний університет імені Лесі Українки, м. Луцьк
Hlynchuk.Ludmila@vnu.edu.ua

ПОБУДОВА ПРИКЛАДНИХ ПРОГРАМ МОВОЮ ЛОГІКИ: ПРИКЛАД СТВОРЕННЯ БАЗИ ДАНИХ ТА РОБОТА З НЕЮ

Анотація. В роботі пропонується аналіз та опис прикладу побудови прикладної програми мовою логіки. У даному випадку описано проектування та реалізацію на мові логічного програмування Prolog створення бази даних та роботу з нею. Стаття складається з декількох стандартних частин, де вказано: причини використання логічної мови програмування та у яких сферах, чому здобувачі освіти повинні розглядати під час свого навчання парадигму логічного програмування; коротко проаналізовано останні дослідження та публікації в сфері даної тематики; спроектована БД, яка містить інформацію про викладачів, детально описаний код для кожного предикату та правила. Зібрана за даним описом програма чітко та правильно функціонує в онлайн-компіляторі, що дає можливість демонструвати її як приклад при вивченні відповідної теми.

Ключові слова: мова програмування Prolog; база даних; код; предикат; правила.

Hlynchuk L. Construction of example programs in the language of logic: an example of creating a database and working with it. The paper offers an analysis and description of an example of building an application program in the language of logic. In this case, the design and implementation in the logical programming language Prolog of creating a database and working with it is described. The article consists of several standard parts, which indicate: the reasons for using a logical programming language and in which areas, why students should consider the logical programming paradigm during their studies; the latest research and publications in the field of this topic are briefly analyzed; designed database containing information about teachers, detailed code for each predicate and rule. The program compiled according to this description functions clearly and correctly in the online compiler, which makes it possible to demonstrate it as an example when studying the relevant topic.

Keywords: Prolog programming language; Database; code; predicate; rules.

АКТУАЛЬНІСТЬ ПРОБЛЕМИ. Побудова прикладних програм мовою логіки, зокрема використання мови програмування Prolog, залишається актуальною темою в сучасному світі програмування з кількох причин, опишемо їх детальніше.

Перша та дуже важлива причина використання – ефективність при роботі з базами даних. Prolog є потужним інструментом для обробки та розв'язання складних завдань у галузі обробки і аналізу даних. Він може бути використаний для створення та роботи з базами даних, що дозволяє ефективно виконувати різноманітні завдання, такі як пошук, фільтрація, злиття даних тощо. Розроблений приклад саме і демонструється в даній роботі.

Логічне програмування використовується також і для представлення знань. Оскільки, мова Prolog базується на логіці першого порядку і надає можливості для легкого та зрозумілого представлення знань та фактів. Це робить Prolog ідеальним вибором для створення баз знань і експертних систем, які вимагають дедуктивного мислення та логічного виводу.

Наступна особливість – можливість автоматизації в інтелектуальних системах. Prolog дозволяє реалізувати інтелектуальні системи, які можуть автоматизувати рішення в складних областях, таких як медицина, фінанси, штучний інтелект та інші. Використання мови логіки дозволяє створювати системи, які можуть робити висновки та приймати рішення на основі логічних правил і знань. [1]

Ще одна особливість для можливого використання – це галузь машинного навчання. Програми, написані на Prolog, можуть бути використані для реалізації алгоритмів машинного навчання, таких як класифікація, кластеризація та прогнозування. Також Prolog може бути використаний для аналізу та обробки даних в різних областях, включаючи біоінформатику, соціальні науки та інші.

Як бачимо, усі ці фактори роблять побудову прикладних програм мовою логіки, зокрема з використанням Prolog, актуальною темою, особливо в контексті розвитку інтелектуальних систем та обробки даних.

Що стосується актуальності вивчення логічної мови програмування для здобувачів освіти спеціальності 122 «Комп'ютерні науки» галузі знань 12 «Інформаційні технології», то потрібно звернути увагу на відповідний стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр». У стандарті спеціальна (фахова) компетентність 8 формулюється наступним чином «Здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.» [2]. Як бачимо компетентність каже про набуття здатності застосовувати різні парадигми програмування серед яких є логічне. А відповідний програмний результат 9 у стандарті каже «Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв'язання задач в галузі комп'ютерних наук.» [2]. А як можна обрати з позиції зручності та якості застосування відповідну парадигму, якщо здобувач освіти не знає про неї та не вміє використовувати. Звідси і висновок, що враховуючи фахову компетентність 8 та програмний результат 9 для здобувачів освіти спеціальності 122 «Комп'ютерні науки» потрібно не тільки розказувати про теоретичні основи в цьому напрямі, але і показувати конкретні практичні застосування мови логіки.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ. Ось деякі з останніх досліджень і публікацій, які вдалось відстежити, пов'язаних із даною темою:

1) Міжнародна конференція з логічного програмування (ICLP 2023).

Конференція ICLP 2023 включала численні доповіді та презентації про застосування логічного програмування в різних сферах, включаючи управління базами даних. Теми охоплювали вдосконалення теоретичних основ, проектування мов, аналіз програм та методології реалізації, що стосуються додатків логічного програмування. Конференція також висвітлила нові застосування в інтеграції даних, інженерії програмного забезпечення та штучному інтелекті, що мають відношення до створення та управління базами даних за допомогою логічного програмування (ICLP 2023). [3]

2) Журнал «Theory and Practice of Logic Programming» (TPLP).

Журнал TPLP публікує статті, які наголошують як на теоретичних, так і на практичних аспектах логічного програмування. Нещодавні публікації в журналі досліджували такі теми, як немонотонні міркування, представлення знань і логічне програмування обмежень, які є ключовими для розробки надійних програм баз даних. Відомі статті включають дослідження з індуктивного логічного програмування та мультиреляційного аналізу даних, які безпосередньо застосовуються до створення та керування базами даних (Cambridge University Press & Assessment). [4]

Останні статті в TPLP:

"IASCAR: Incremental Answer Set Counting by Anytime Refinement" та "Unit Testing in ASP Revisited: Language and Test-Driven Development Environment" є прикладами останніх досліджень, зосереджених на застосуванні логічного програмування в штучному інтелекті та системах баз даних. Ці дослідження пропонують уявлення про підвищення ефективності та надійності систем баз даних за допомогою методик логічного програмування [4]

Ці джерела надають всебічний огляд сучасних досягнень і напрямків досліджень у галузі логічного програмування, особливо його застосування у створенні та управлінні базами даних. Для більш детальної інформації можна ознайомитися з повними статтями та матеріалами конференції ICLP 2023 та журналу TPLP.

Мета дослідження – розглянути приклад побудови програми для створення та роботи з базою даних за допомогою мови логіки, в даному випадку мовою програмування Prolog.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ. База даних (БД), записана на Prolog, представляться набором фактів і правил, оформлених у синтаксично правильній формі. Ця впорядкована сукупність і є змістом бази даних.

Існують три добре відомі моделі організації БД: ієрархічна, мережева та реляційна моделі. У Prolog легко реалізувати реляційну модель БД, тому ми розглянемо саме її.

Нагадаємо, що дані в реляційних моделях подаються у вигляді таблиць, які складаються з рядків і стовпців. У Prolog є спеціальні засоби для організації баз даних, розраховані на роботу з реляційними базами. Пролог особливо зручний для створення діалогових систем для реляційних БД, оскільки його внутрішні процедури автоматично вибирають факти з потрібними значеннями відомих параметрів і присвоюють значення ще не визначеним. Це дозволяє знаходити всі наявні відповіді на зроблений запит. [5]

Процес побудови можна розділити на 2 кроки: проектування БД та програмування БД. Розглянемо та опишемо детальніше кожен з кроків.

Створення бази даних у Prolog починається з етапу проектування бази. Потрібно врахувати такі чинники: розмір бази даних; оорганізацію елементів бази даних; способи роботи і вміст бази даних.

Використання баз даних в оперативній пам'яті виправдано, якщо БД має невеликий обсяг. Спочатку необхідно задати початкові дані і створити базу. Потім йде черга системи керування базою даних (СКБД), орієнтованої на діалог з користувачем. Будь-яка система такого типу має як мінімум такі можливості:

- занесення в базу нових даних;
- ввилучення даних з бази;
- вибірка і виведення даних з бази.

Ці вимоги передбачають наявність у системі меню, яке дозволяє користувачеві легко орієнтуватися та використовувати стандартні функції СКБД. [5]

Найкращий спосіб розпочати проектування програми – це створити її діаграму потоків даних. Для початку сформулюємо завдання: реалізувати базу даних, в якій зібрана інформація про викладачів певного університету, та роботу з нею. Структурна схема, що відповідає цій БД, показана на рисунку 1. Вона демонструє, що присутній модуль меню, який дозволяє користувачеві вибирати між п’ятьма пунктами: пункт(1) – додати нового викладача у базу даних, пункт(2) – для вилучення даних, пункт(3) – переглянути (знайти) інформацію про викладача у базі даних, пункт(4) – вивести інформацію про всіх викладачів, що є у базі даних, пункт(5) – завершити сеанс роботи з базою даних.

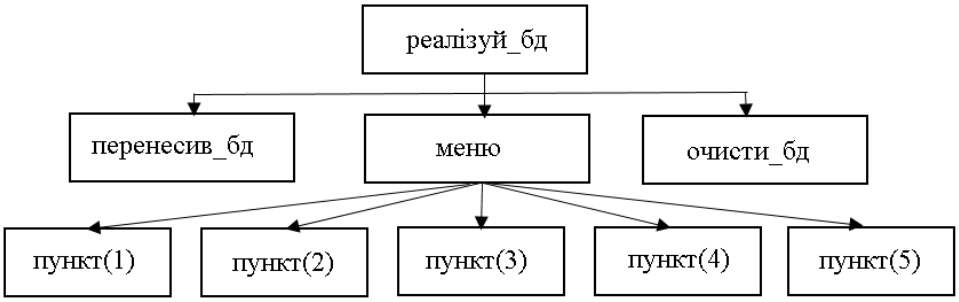

Рисунок 1. Структурна схема БД

Розглянемо операцію запису в базу нових даних. Користувач, що працює за клавіатурою, запускає програму. Для запису нових даних потрібно вибрати пункт(1). Керування в такий спосіб передається від основного модуля реалізуй_бд до модуля меню, а потім до пункт(1). Оскільки введення здійснюється з клавіатури, то саме від неї і бере початок потік даних, а закінчується на моніторі і базі даних.

Щоб написати програму БД мовою Prolog використовують дані, що є інформацією про шістьох вкладачів певного університету. База включає імена викладачів, назви факультетів, де вони працюють, назви кафедр, вчене звання, кандидатський ступінь, досвід роботи. Вся інформація міститься в таблиці 1. [5]

Для роботи з нею необхідний предикат, що зберігає цю інформацію.

```
teacher(t_name,          /*  прізвище та ім'я викладача (string)  */
f_name,                  /* назва факультету (string)  */
k_name,                  /* назва кафедри (string)  */
```

```
ac_title,                /* вчене звання (string)  */
cand_degree,             /* науковий ступінь (string)  */
work_exper)              /* досвід роботи (integer)  */
```

Таблиця 1.

Дані про викладачів

Прізвище та ім'я викладача	Назва факультету	Назва кафедри	Вчене звання	Науковий ступінь	Досвід роботи, кількість років
Кропивець Євген	Юридичний факультет	Кримінально-правових дисциплін	Доцент	Кандидат	12
Світлинський Віталій	Міжнародно-правовий факультет	Міжнародних стандартів	Професор	Доктор наук	23
Волошинський Іван	Факультет суспільних наук	Кафедра філософії	Доцент	Кандидат	18
Кавольчук Валентина	Факультет міжнародних відносин	Міжнароднх комунікацій та політичного аналізу	Доцент	Кандидат	15
Карпенко Василина	Факультет психології	Загальної психології	Професор	Доктор наук	26
Весела Катерина	Історичний факультет	Новітньої історії України	Професор	Доктор наук	20

Об'єктам предиката присвоєні імена, що пояснюють суть справи, і тому запам'ятовуються легко. Об'єкт T_name позначає повне ім'я викладача, F_name – назву факультету і т.д. Цей предикат є основою для подальшої побудови бази даних.

Prolog вимагає, щоб усі твердження того самого предиката були згруповані в одному місці. Відповідно до цієї вимоги групу предиката teacher записують у вигляді:

teacher("Кропивець Євген", "Юридичний факультет", "Кримінально-правових дисциплін", "Доцент", "Кандидат", 12).

teacher("Світлинський Віталій", "Міжнародно-правовий факультет", "Міжнародних стандартів", "Професор", "Доктор наук ", 23).

teacher("Волошинський Іван", "Факультет суспільних наук", " Кафедра філософії ", "Доцент", "Кандидат", 18).

teacher("Кавольчук Валентина", "Факультет міжнародних відносин", " Міжнароднх комунікацій та політичного аналізу ", "Доцент", "Кандидат", 15).

teacher("Карпенко Василина", "Факультет психології", "Загальної психології", "Професор", "Доктор наук ", 26).

teacher("Весела Катерина", "Історичний факультет", "Новітньої історії України", "Професор", "Доктор наук ", 20).

Зауважимо, що якщо об'єкти тверджень є рядками і починаються з великих букв, то їх беруть в лапки.

На початку роботи програми необхідно занести в динамічну БД призначену для неї інформацію, що міститься в статичній БД. Це завдання виконує предикат перенесив_бд. Предикат очисти_бд призначений для розв'язання суміжної задачі: очищення БД перед закінченням роботи програми. Насправді, ця робота є зайвою, але ми включили цей предикат у нашу програму, тому що очищення БД потрібне в деяких додатках.

Завданням предиката помилка є реагування на введення неправильної вхідної

інформації.

Предикат teacher призначений для задання початкового вмісту бази даних, тієї інформації, що є в таблиці 1. Коли програма розпочинає роботу, ця інформація заноситься у твердження предиката dteacher.

Предикат реалізуй_бд є головним правилом (модулем) програми. Предикат меню визначає правило, що здійснює інтерфейс із користувачем за допомогою меню. Предикати пункт(номер) визначають різні правила, що виконують усі можливі операції над БД. [5]

Опишемо детально реалізацію програмних модулів.

Після закінчення етапу проектування можна розпочати стадію реалізації проекту. Тепер нашим завданням є написати основні і допоміжні правила, що будуть потрібні основним модулям.

На початку програми потрібно вказати, що предикат, який ми використовуємо буде динамічним, для того щоб робити зміни у БД. Предикат буде мати вигляд:

:- dynamic dteacher /6.

Далі потрібно записати факти статичної БД і продовжити роботу за логікою рисунку 1.

Реалізація головного модуля.

Головний модуль програми реалізуй_бд є одночасно і метою програми (запитом), та має вигляд як на рисунку 2.

```
22 реалізуй_бд :-
23     перенесив_бд,
24     меню,
25     очисти_бд.
```

Рисунок 2. – Код для реалізуй_бд

Модуль посилає в базу інформацію з тверджень teacher, показує меню й очищає БД після закінчення роботи програми.

Вхідні дані БД задають за допомогою тверджень з використанням статичних предикатів. Правило для занесення в базу цієї інформації таке, як на рисунку 3.

```
11 перенесив_бд :-
12 teacher(T_name,F_name,K_name,Ac_title, Cand_degree, Work_exper),
13 assertz(dteacher(T_name,F_name,K_name,Ac_title, Cand_degree, Work_exper)),
14 fail.
15 перенесив_бд :- !.
```

Рисунок 3. – Код для перенесив_бд

Предикат очищення бази даних – рисунок 4.

```
17 очисти_бд :-
18     retract(dteacher(_,_,_,_,_,_,_,_)),
19     fail.
20 очисти_бд :- !.
```

Рисунок 4. – Код для очисти_бд

Об'єкти тверджень dteacher у цьому правилі не становлять інтересу, тому використовуються анонімні змінні.

Меню призначено, щоб користувачеві було зручно вибирати програмні функції. Модуль меню показує п'ять доступних користувачеві опцій.

Дані опції:

1. Додати нового гравця у базу даних
2. Вилучити гравця з бази даних.
3. Переглянути (знайти) інформацію про гравця у базі даних
4. Вивести інформацію про всіх гравців, що є у базі даних
5. Завершити сеанс роботи з базою даних

Модуль меню переважно складається з предикатів write, які виводять на екран перераховані вище опції. Зірочки використовують для виділення простору, що містить меню. В модулі наявні предикати write, що створюють бордюр із зірок, та предикат, що запитує в користувача ціле число в діапазоні від 1 до 5.

Модуль меню цілком відповідає поставленим до нього в проекті програми вимогам, рисунок 5.

```
27 меню :-
28     repeat,
29     write("* * * * * * * * * * * * * * * * * * * *"), nl,
30     write(" 1. Додати нового викладача у базу даних"),nl,
31     write(" 2. Вилучити викладача з бази даних"),nl,
32     write(" 3. Переглянути (знайти) інформацію про викладача у базі даних"),nl,
33     write(" 4. Вивести інформацію про всіх викладачів, що є у базі даних"),nl,
34     write(" 5. Завершити сеанс роботи з базою даних"),nl,
35     write(" * * * * * * * * * * * * * * * * * * * "),nl,
36     nl,
37     write("Вибери номер пункту 1, 2, 3, 4 або 5 :"),
38     read(Choice),
39     пункт(Choice),
40     Choice = 5,
41     !.
```

Рисунок 5. – Код для меню

Якщо користувач введе число, що дорівнює 5 (5 викликає закінчення програми), вибір Choice = 5 викликає повернення до предиката repeat.

Розглянемо реалізацію правил пункт.

Правило пункт(1) призначено для занесення нових даних в БД. Цей модуль просить користувача ввести дані з клавіатури, зчитує їх і записє в БД нове твердження dteacher в кінець бази. Після цього модуль повертає керування головному меню.

Предикати write і read інформують користувача про те, які дані він повинен увести, і зчитують ці дані з клавіатури, код на рисунку 6.

```
43 /* Додати нового викладача у БД */
44 пункт(1) :-
45 write("Введи прізвище та імя: "), read(T_name),
46 write("Назва факультету: "), read(F_name),
47 write("Назва кафедри: "), read(K_name),
48 write("Вчене звання: "), read(Ac_title),
49 write("Науковий ступінь: "), read(Cand_degree),
50 write("Досвід роботи: "), read(Work_exper),
51 assertz(dteacher(T_name,F_name,K_name,Ac_title, Cand_degree, Work_exper)),
52 write(T_name), write(" було додано до БД."), nl,!.
```

Рисунок 6. – Код для пункт(1)

Призначенням модуля пункт(2) є вилучення інформації з БД. Це правило, як і правило пункт(1), запитує в користувача ім'я викладача і вилучає з БД твердження, що містить інформацію про нього. Потім керування знову передається головному меню. Код для пункт(2) на рисунку 7.

```
54 /* Вилучити викладача з БД */
55 пункт(2) :-
56 write("Введи імя для вилучення: "), read(T_name),
57 retract(dteacher(T_name,_,_,_,_,_)),
58 write(T_name), write(" був вилучений з БД."), nl, !.
```

Рисунок 7. – Код для пункт(2)

Призначенням модуля пункт(3) є пошук даних, що містяться в базі. Цей модуль запитує ім'я викладача. Якщо в БД є твердження, що містить введене ім'я, модуль робить вибірку даних і виводить їх на екран у зручному форматі.

Предикат dteacher шукає потрібне твердження в базі даних і вибирає рядкові і цілі значення з кожного запитуваного пункту. Зверніть увагу, що для пошуку за певним іменем дані потрібно вводити в подвійних лапках, інакше результат буде видаватися не вірно. Далі ціла низка предикатів write виводить отримані значення, рисунок 8.

```
/* Переглянути (знайти) інформацію про викладача у базі даних */
пункт(3) :-
write("Введи імя для пошуку: "), read(T_name),
dteacher(T_name,F_name,K_name,Ac_title, Cand_degree, Work_exper), nl,
write(" Знайдено шуканого викладача:"),nl,
nl, write(" Прізвище та імя :       "),write(T_name),
nl, write(" Назва факультету :  "),write(F_name),
nl, write(" Назва кафедри : "),write(K_name),
nl, write(" Вчене звання :      "),write(Ac_title),
nl, write(" Науковий ступінь :  "), write(Cand_degree),
nl, write(" Досвід роботи: "), write(Work_exper), write(" роки(ів)"),
nl, nl, !.
```

Рисунок 8. – Код для пункт(3)

Якщо в БД відсутнє твердження з введеним користувачем ім'ям гравця, програма видає повідомлення про помилку. Варіант пункт(3), відповідальний за видавання повідомлення про помилку, виглядає так:

```
пункт(3) :-
write("Не знайдено інформації в БД!"),
nl, !.
```

Рисунок 9. – Код для пункт(3), інший варіант

Модуль пункт(4) дозволяє вивести всі записи БД. Код виглядає наступним чином:

```
/* Вивести інформацію про всіх викладачів, що є у базі даних */
пункт(4) :-
dteacher(T_name,F_name,K_name,Ac_title, Cand_degree, Work_exper),nl,
write(" Інформація про всіх викладачів з бази:"),nl,
nl, write(" Прізвище та імя :       "),write(T_name),
nl, write(" Назва факультету :  "),write(F_name),
nl, write(" Назва кафедри : "),write(K_name),
nl, write(" Вчене звання :      "),write(Ac_title),
nl, write(" Науковий ступінь :  "), write(Cand_degree),
nl, write(" Досвід роботи: "), write(Work_exper), write(" роки(ів)"),
nl, nl, fail.
```

Рисунок 10. – Код для пункт(4)

Останній предикат fail дає можливість вивести інформацію про одного гравця, тоді дає відмову і процес повертається знову, тобто і так далі виводиться інформація про усіх гравців, які на момент виведення є у БД.

Модуль пункт(5) забезпечує нормальне закінчення сеансу роботи з базою даних. Модуль вимагає від користувача чіткої відповіді на запитання, чи хоче він закінчити роботу з програмою:

```
/* Завершити сеанс роботи з БД */
пункт(5) :-
    write("Хочеш продовжити роботу з БД? (y/n)"),
    read(Answer),
    sub_atom(Answer, 0, 1, _, 'n'),
    !.
```

Рисунок 11. – Код для пункт(5)

Предикат sub_atom успішний тільки якщо відповідь користувача на запит програми починається з літери у. Якщо вводиться інша літера, предикат неуспішний, тому відбувається повернення до предиката repeat модуля menu.

Програма повинна належно реагувати на допущені користувачем помилки під час введення. Якщо користувач введе число, менше за 1 або більше за 5, буде виконуватись одне з правил, як на рисунку 12. Ці правила викликають модуль помилка:

```
пункт(Choice) :- Choice < 1, помилка.
пункт(Choice) :- Choice > 5, помилка.
помилка :-
write("Не вірно вибрали номер пункту."), nl,
write("Натисни клавішу для продовження"),read(_).
```

Рисунок 12. – Код реагування на допущені помилки

Код було реалізовано у потужному онлайн-компіляторі https://swish.swi-prolog.org/, тобто зібрана з описаних частин програма. Робота з БД за допомогою меню виглядає як на рисунку 13.

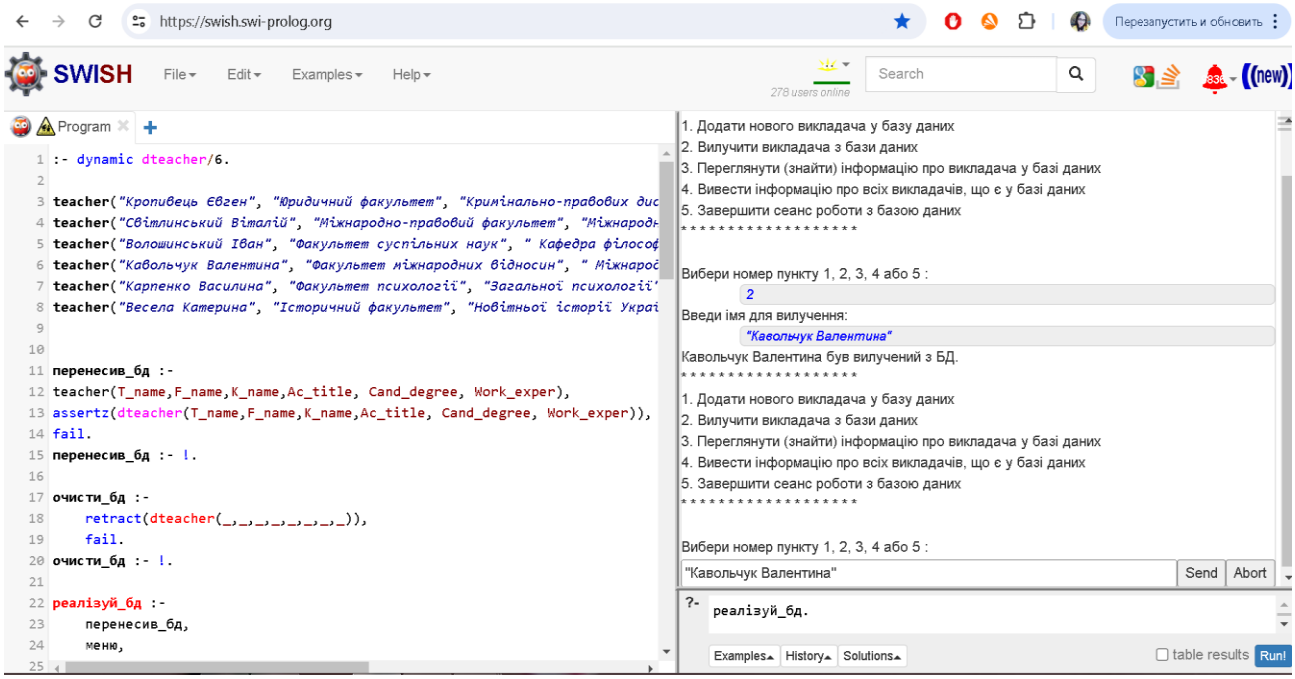

Рисунок 13. – Робота з БД в онлайн-компіляторі

ВИСНОВКИ. В даній роботі були описані особливості та сфери застосування мови логіки. В тому ж пункті вказано, що згідно джерела [2] здобувачів освіти спеціальності 122 «Комп'ютерні науки» галузі знань 12 «Інформаційні технології» потрібно знайомити з основними парадигмами програмування, серед яких є парадигма логічного програмування. У пункті «ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ» спроектована БД, яка містить інформацію про певних викладачів, проаналізовано та побудовано код для роботи з даною БД. Зібрана програма в онлайн-компіляторі дозволяє працювати користувачу за 5-ма, описаними вище, пунктами меню та видавати відповідний результат.

Створену БД та описаний код можна і корисно використовувати для здобувачів освіти, для навчання у програмуванні, та науково-педагогічному персоналу при поясненні відповідної теми. А саме, як приклад демонстрації побудови прикладних програм мовою логіки (для створення та роботи з БД).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Логічне програмування: принципи та застосування. *FoxmindEd*. URL: https://foxminded.ua/lohichne-prohramuvannia/.
2. Стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп'ютерні науки». Чинний (вводиться в дію) з 2019-2020 рр. Вид. офіц. Міністерство освіти і науки України, Київ, 2019. URL: https://mon.gov.ua/storage/app/media/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyut.nauk.bakalavr-1.pdf.
3. International Conference on Logic Programming (ICLP 2023). *International Conference on Logic Programming (ICLP 2023)*. URL: https://iclp2023.imperial.ac.uk/
4. Theory and Practice of Logic Programming | Cambridge Core. *Cambridge Core*. URL: https://www.cambridge.org/core/journals/theory-and-practice-of-logic-programming.
5. Заяць В. М. Логічне і функціональне програмування. Системний підхід : підручник / В. М. Заяць, М. М. Заяць. Рівне : НУВГП, 2018. 422 с.