Szkoła Gospodarstwa Wiejskiego
Wydział Zastosowań Informatyki i Matematyki

PARADYGMATY PROGRAMOWANIA

Jednostka 8 – Paradygmat deklaratywny: SQL, logika i Prolog

Kierunek: Informatyka, semestr 5
Prowadzący: dr inż. Krzysztof Malczewski

1. Cel i zakres zajęć

Celem zajęć jest:

- zrozumienie **paradygmatu deklaratywnego** i jego różnic względem podejść imperatywnych i funkcyjnych,
- poznanie podstaw **języka SQL** oraz zapytań deklaratywnych,
- zapoznanie z podstawami logiki programowania (Prolog lub równoważne),
- nauczenie się formułowania **reguł i zapytań logicznych**, a nie kroków algorytmicznych.

Zakres tematyczny:

- definicja paradygmatu deklaratywnego,
- języki deklaratywne: SQL, Prolog, zapytania logiczne,
- formułowanie żądań („co") zamiast instrukcji („jak"),
- proste przykłady baz danych i faktów logicznych,
- reguły, relacje, zapytania i wzorce.

---

2. Wprowadzenie teoretyczne

2.1 Paradygmat deklaratywny – definicja

W paradygmacie deklaratywnym **programista określa, *co* ma zostać obliczone**, a nie *jak* to zrobić.
Silnik wykonawczy (np. optymalizator zapytań SQL, maszyna Prolog) sam decyduje o sposobie osiągnięcia wyniku.

Cechy:

- brak jawnych pętli, instrukcji sterujących i przypisań,
- operowanie na zbiorach, relacjach lub formułach logicznych,
- nacisk na opis relacji i zależności.

2.2 SQL jako język deklaratywny

SQL (Structured Query Language) służy do definiowania, modyfikowania i **odpytywania danych** w relacyjnych bazach danych.
Przykład:

```
SELECT imie, nazwisko
FROM studenci
WHERE rok = 3;
```

Powyższe zapytanie **opisuje, co chcemy uzyskać**, ale nie mówi, jak silnik bazy ma znaleźć te dane (plan wykonania tworzony jest automatycznie).

2.3 Logika programowania i Prolog

W językach logicznych (np. Prolog):

- baza wiedzy składa się z **faktów** i **reguł**,
- zapytania są formułowane jako cele logiczne,
- silnik wyszukiwania (backtracking) sam znajduje dopasowania.

Przykład:

```
rodzic(jan, anna).
rodzic(anna, piotr).

przodek(X,Y) :- rodzic(X,Y).
przodek(X,Y) :- rodzic(X,Z), przodek(Z,Y).
```

Zapytanie:

```
?- przodek(jan, piotr).
```

Silnik automatycznie znajduje dowód, że Jan jest przodkiem Piotra.

---

3. Tutorial 1 – SQL krok po kroku

3.1 Tworzenie przykładowej tabeli

```
CREATE TABLE Studenci (
    id INT PRIMARY KEY,
    imie TEXT,
    nazwisko TEXT,
    rok INT,
    kierunek TEXT
);

INSERT INTO Studenci VALUES
(1, 'Anna', 'Kowalska', 1, 'Informatyka'),
(2, 'Jan', 'Nowak', 3, 'Informatyka'),
(3, 'Maria', 'Wiśniewska', 2, 'Matematyka'),
(4, 'Krzysztof', 'Malinowski', 3, 'Informatyka');
```

3.2 Podstawowe zapytania

Selekcja danych

```
SELECT imie, nazwisko FROM Studenci;
```

Warunki

```
SELECT imie, nazwisko FROM Studenci WHERE rok = 3;
```

Sortowanie i agregacja

```
SELECT rok, COUNT(*) as liczba
FROM Studenci
GROUP BY rok
ORDER BY rok;
```

3.3 Złączenia

```
SELECT s.imie, s.nazwisko, k.nazwa
FROM Studenci s
JOIN Kierunki k ON s.kierunek = k.kod;
```

👉 SQL umożliwia **opisanie relacji** między tabelami bez wskazywania sposobu iteracji.

---

4. Tutorial 2 – Prolog krok po kroku (lub analogiczna logika)

4.1 Fakty i reguły

```
student(anna, informatyka, 1).
student(jan, informatyka, 3).
student(maria, matematyka, 2).

na_roku(Imie, Rok) :- student(Imie, _, Rok).
```

4.2 Zapytania

```
?- na_roku(jan, 3).
true.

?- student(Imie, informatyka, Rok).
Imie = anna, Rok = 1 ;
Imie = jan, Rok = 3.
```

👉 Widzimy, że nie mówimy, jak przeszukiwać fakty — **opisujemy zależności**.

---

5. Zadania do samodzielnego wykonania

Zadanie 1 (obowiązkowe, SQL)

Utwórz tabelę `Przedmioty` i `Oceny`.
Napisz zapytania SQL, które:

- zwracają wszystkich studentów z danego roku,
- obliczają średnią ocen dla każdego studenta,
- wypisują nazwiska studentów, którzy mają średnią > 4.0.

Zadanie 2 (obowiązkowe, Prolog / logika)

Zdefiniuj fakty i reguły reprezentujące relacje rodzinne.
Zaimplementuj reguły `rodzic/2`, `dziecko/2`, `przodek/2`.
Zadaj zapytania sprawdzające różne relacje.

Zadanie 3 (dodatkowe)

Stwórz prostą bazę wiedzy w Prologu opisującą studentów, kierunki i prowadzących, a następnie napisz regułę, która odpowiada na pytanie: „Kto jest prowadzącym studenta X?”.

---

6. Pytania kontrolne

1. Czym różni się program deklaratywny od imperatywnego?
2. Dlaczego w SQL nie opisujemy kroków algorytmu?
3. Czym są fakty i reguły w Prologu?
4. Jak działa mechanizm dopasowywania i backtrackingu?
5. Jakie zalety ma deklaratywność przy przetwarzaniu danych?

---

7. Checklisty

SQL

- Tabele utworzone poprawnie.
- Zapytania selekcji, filtrowania i agregacji działają.
- Wyniki są poprawne i odpowiadają zadaniu.

Prolog

- Fakty i reguły zdefiniowane poprawnie.
- Zapytania zwracają oczekiwane odpowiedzi.
- Użyto reguł złożonych (rekurencyjnych) przynajmniej w jednym zadaniu.

---

8. Wzór sprawozdania

Dodatkowo:

- Fragmenty zapytań SQL i ich wyniki.
- Fakty i reguły Prologa + przykładowe zapytania i odpowiedzi.
- Porównanie stylu deklaratywnego z wcześniejszymi (imperatywny/funkcyjny).

Zakończenie

Po wykonaniu ćwiczenia student powinien:

- rozumieć ideę paradygmatu deklaratywnego,
- potrafić formułować zapytania SQL i logiczne,
- znać różnicę między opisem *co* a *jak*,
- umieć zapisać proste relacje w postaci faktów i reguł,
- rozumieć podstawy działania maszyn logicznych i optymalizatorów zapytań.