Katedra
Informatyki i Automatyki
Politechniki Rzeszowskiej

Bazy Danych

2. Zadanie domowe

Opracował: Maciej Penar

Spis treści

1. Zadanie domowe

Wybrać dowolne zadanie i zrealizować:

ZALEŻNOŚCI FUNKCYJNE – TOP-DOWN

Mała pomoc: Link

Zależność funkcyjna oznacza że dla pewnych wartości atrybutów (kolumn) tabeli, pewne inne atrybuty **zawsze** mają określoną wartość.

Zależność funkcyjną (ZF) zapisujemy jako: $X \rightarrow Y$ np. $PESEL \rightarrow IMIE$ co oznacza że każdy Pesel jednoznacznie wyznacza imię.

Ogólnie rzecz biorąc ZF przyjmują postać: $X_1X_2 \dots X_n \rightarrow Y_1Y_2 \dots Y_m$

Co oznacza że każda kombinacja kolumn $X_1X_2 \dots X_n$ wyznacza dokładnie wartości kolumn $Y_1Y_2 \dots Y_m$

ZF mają kilka cech:

1) Prawą stronę ZF możemy dekomponować:

$$X_1X_2 \dots X_n \rightarrow Y_1Y_2 \dots Y_m = X_1X_2 \dots X_n \rightarrow Y_1,\quad X_1X_2 \dots X_n \rightarrow Y_2,\ \dots\quad X_1X_2 \dots X_n \rightarrow Y_m$$

2) **Lewej strony ZF nie wolno dekomponować**

3) Są przechodnie, tj.:

Jeśli wiemy, że $X \rightarrow Y$ i $Y \rightarrow Z$, to też $X \rightarrow Z$

4) Są zwrotne: $X \rightarrow X$ oraz $X_1X_2 \dots X_n \rightarrow X_1X_2 \dots X_n$

Jeśli w schemacie $R(X, Y_1 \dots Y_m)$ atrybut X (lub zbiór atrybutów) wyznacza pozostałe kolumny to ZF: $X \rightarrow Y_1Y_2 \dots Y_m$, to jest nazywany **kluczem** (więzy UNIQUE).

Zazwyczaj w tabeli o schemacie wybierany jest **klucz główny** – tylko jeden.

Jedna z metodologii projektowania Baz Danych polega na utworzeniu A) schematu uniwersalnego czyli bazy danych z jedną tabelą oraz B) sformułowania zbioru ZF dzięki którym nastąpi dekompozycja. Na przykład:

A) **Baza Danych(PESEL, Imie, Nazwisko, Miasto, Ulica, Numer Domu, Kod-Pocztowy, Numer Zamówienia, Adres Wysyłki, Pozycja Paragonu, Nazwa Produku, Ilość na Paragonie, …. itp.)**

B) $PESEL \rightarrow IMIE$, $PESEL \rightarrow NAZWISKO$, $KOD-POCZTOWY \rightarrow MIASTO$, $MIASTO, ULICA, NUMER-DOMU \rightarrow ADRES\_WYSYŁKI$

Dekompozycja jest wykonywana w oparciu o pewne reguł – zbiory tych reguł nazywamy postaciami normalnymi. Jeśli wszystkie tabele spełniają $X-t$ą postać normalną, to mówimy że baza danych jest w $X-tej$ postaci normalnej.

Przykładowe postacie normalne:

1. 1PN – gdy wartości kolumn są atomowe (nie są kolekcjami) oraz istnieje klucz główny
2. 2PN – gdy kolumny niekluczowe są w pełni zależne funkcyjnie, czyli dla ZF: $AB \rightarrow CD$ i kolumn kluczowych AB, nie istnieje ZF: $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow C$, $B \rightarrow D$
3. 3PN – gdy kolumny niekluczowe nie posiadają zależności przechodnich, czyli jeśli A jest kolumną klucza i ZF: $A \rightarrow BC$, to albo:
    a. nie istnieje $B \rightarrow C$
    b. nie istnieje $C \rightarrow B$
    c. istnieje $B \rightarrow C$ i $C \rightarrow B$ (wszystkie kolumny są kluczami - tricky part)
4. BCNF – gdy kolumny niekluczowe nie posiadają zależności przechodnich, czyli jeśli A jest kolumną klucza **głównego** i ZF: $A \rightarrow BC$, to nie istnieje $B \rightarrow C$ i nie istnieje $C \rightarrow B$

Napisać program który na wejściu przyjmuje:

- zbiór atrybutów
- zbiór zależności funkcyjnych
- poziom normalizacji (1PN/2PN/3PN/BCNF)

I dzieli atrybuty na podzbiory (tabele) spełniające kryteria wg. wybranego poziomu normalizacji.

Jakieś przypadki testowe:

1. $R(A,B,C,D), ZF: AB \rightarrow C, C \rightarrow D, D \rightarrow A$
   Dekompozycja do: $R(A,B,C), X(C,D), Z(D,A)$
2. $R(A,B,C,D), ZF: B \rightarrow C, B \rightarrow D$
   Dekompozycja do: $R(A,B), X(B,C,D)$
3. $R(A,B,C,D,E), ZF: AB \rightarrow C, C \rightarrow D, D \rightarrow E$
   Dekompozycja do: $R(A,B,C), X(C,D), Z(D,E)$

ORM

Napisać prosty ORM – Code First - w Javie lub C# wykorzystujący:

- Atrybuty (C#)
- Adnotacje (Java)

ORM powinien umożliwiać:

- CREATE TABLE ze wsparciem dla kluczy obcych i ograniczeniami ON DELETE/ON UPDATE
- DROP TABLE
- INSERT
- UPDATE
- DELETE
- Wykonanie dowolnego SQL'a (RawSQL)

Jeśli chcecie zaimponować to rozwiązać którykolwiek z tych problemów:

- opracować sposób zmiany schematu: np. dodawania kolumn, usuwania kolumn
- dodać wsparcie do SELECT WHERE

PROXY CACHE'UJĄCE

Spiąć się z dowolnym ORM'em oraz:

- Backend
- Redis
- Bazę danych - dowolną

I napisać kawałek kodu w sposób **możliwie ogólny** dzięki któremu można wykonywać operacje odczytu na bazie danych które w pierwszej kolejności pobierane są z Redisa.

W momencie w którym użytkownik formułuje zapytanie np. **SELECT * FROM tableX;** Wykonywany jest następujący algorytm:

1. Dla zapytania wejściowego Z biblioteka ustala klucz cache'owania k oraz dopuszczalny termin przedawnienia t
2. Bibliotek idzie z $<k, t>$ do **Redisa**
    a. Jeśli Redis posiada dane odpowiadające kluczowi k spełniające t to następuje zwrot danych
    b. Jeśli Redis nie posiada danych:
        i. Biblioteka wykonuje zapytanie Z na bazie danych
        ii. Biblioteka umieszcza wynikowe w Redisie pod kluczem k
        iii. Biblioteka zwraca użytkownikowi dane

Wyobrażam sobie że taki framework byłby parametryzowany np.:

- Dopuszczalnym czasem nieświeżości danych
- Adresem Redisa
- Adresem BD

2. … i wykonać

Przygotować sprawozdanie zawierające:

1. Napisać kod, spakować do zip
2. Napisać dokumentację omawiającą problem wraz z omówieniem przypadków testowych

3. Na kiedy

Wysłać na maila mpenar[at]kia.prz.edu.pl do 25 maja 2019.