Informacje o autorze

Autor: Wojciech Krzos

Indeks: 276264 Data: 24.04.2024

Algorytm Bernsteina - Dekompozycja relacji

Dany jest schemat relacji R(P,I,O,E,G,S) oraz zbiór zależności funkcjonalnych F. Rozpoczniemy od analizy i zastosowania algorytmu Bernsteina do tego schematu.

Krok 1: Minimalizacja zbioru zależności funkcjonalnych (F)

Najpierw sprawdzamy, czy w zbiorze zależności funkcjonalnych występują zależności nadmiarowe:

$$F = \{P \to \{G, S\}, \{G, S\} \to P, \{P, I\} \to O, \{G, I\} \to P, \{G, I\} \to S, \{P, G, S\} \to E, \{G, S, I\} \to O\}$$

Analiza:

- $P \to \{G,S\}$ i $\{G,S\} \to P$ tworzą zależność pełną, co implikuje, że P jest kluczem kandydującym oraz $\{G,S\} \to P$ jest kluczem kandydującym.
- $\{P,I\} \rightarrow O$ jest unikalną zależnością i nie jest nadmiarowa.
- $\{G,I\} \to \{P,S\}$ wynika z połączenia $\{G,I\} \to P$ i $\{G,I\} \to S.$
- $\{P,G,S\} \to E$ i $\{G,S,I\} \to O$ mogą być niezbędne do utrzymania, ale ich minimalizacja jest ograniczona przez dostępność kluczy.

Krok 2: Tworzenie relacji z minimalnych zależności funkjonalnych

Przekształcenia znajdują się na końcu dokumentu. Tworzymy relacje dla każdej minimalnej zależności funkjonalnej:

- $R_1(P,G) z P \rightarrow G$
- $R_2(P,S)$ z $P \to S$
- $R_3(G, S, P)$ z $\{G, S\} \rightarrow P$
- $R_4(P, I, O) z \{P, I\} \to O$
- $R_5(G, I, S) z \{G, I\} \to S$
- $R_6(G, I, O) z \{G, I\} \to O$
- $R_7(P,E)$ z $P \to E$

Krok 3: Usuwanie zbędnych relacji

Niektóre z relacji mogą być nadmiarowe lub mogą być połączone:

- R_1 , R_2 i R_7 mogą być połączone, ponieważ zawierają tożsame RHS (Right Hand Side).
- R_5 i R_6 mogą być połączone, ponieważ zawierają tożsame RHS (Right Hand Side).

Krok 4: Relacje końcowe

Dochodzimy do dwóch relacji:

- $S_1(PGSE)$ klucz to P oraz $\{G,S\}$
- $S_2(GISO)$ klucz to $\{GI\}$
- $S_3(PIO)$ klucz to $\{PI\}$

 ${\bf W}$ ten sposób, żadne zleżności funkcyjne nie zostały stracone względem oryginalnego zbioru.

Przykładowa reprezentacja i kod SQL dla relacji S_1 i S_2 , bez uwzględnienia typów danych

```
Definicja tabeli S_1
CREATE TABLE S1 (
   P char(10) NOT NULL,
   G char(10) NOT NULL,
   S char(10) NOT NULL,
   E char(10) NOT NULL,
   PRIMARY KEY (P),
   UNIQUE (G, S)
);
Definicja tabeli S_2
CREATE TABLE S2 (
   G char(10) NOT NULL,
   I char(10) NOT NULL,
   S char(10),
   0 char(10),
   PRIMARY KEY (G, I)
);
Definicja tabeli S_3
CREATE TABLE S3 (
   P char(10) NOT NULL,
   I char(10) NOT NULL,
   O char(10) NOT NULL,
   PRIMARY KEY (P, I)
);
Wstawianie danych do tabeli S_1
INSERT INTO S1 (P, G, S, E)
VALUES ('P001', 'G001', 'S001', 'E001'),
       ('P002', 'G002', 'S002', 'E002');
Wstawianie danych do tabeli S_2
INSERT INTO S2 (G, I, S, 0)
VALUES ('G001', 'I001', 'S001', '0001'),
       ('G002', 'I002', 'S002', '0002');
```

Wstawianie danych do tabeli S_3

```
INSERT INTO S3 (P, I, 0) VALUES ('P001', 'I001', '0001'), ('P002', 'I002', '0002');  
Zapytanie o dane z S_1
```

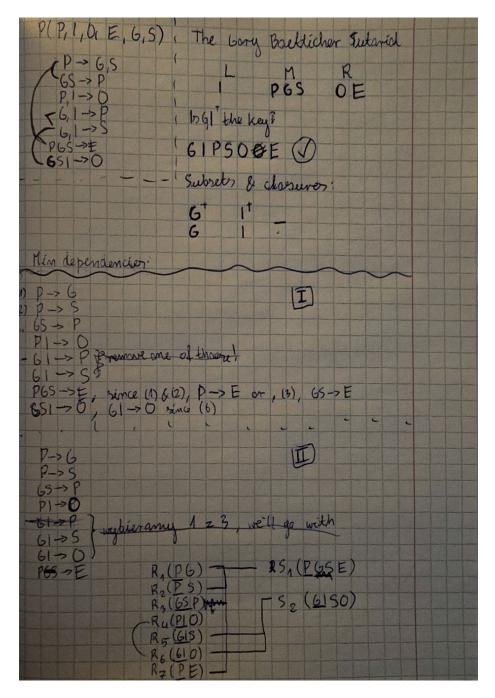
Zapytanie o dane z S_2

SELECT * FROM S2;

SELECT * FROM S1;

Zapytanie o dane z S_3

SELECT * FROM S3;



Rysunek 1: Plik roboczy z przekształceniami