

Deterministyczna metoda przetwarzania ciągów danych

Michał Widera*

Instytut Techniki i Aparatury Medycznej ITAM,
ul. Roosevelta 118, 41-800 Zabrze, Polska
michal@widera.int.pl,
strona WWW: <http://widera.int.pl>

Streszczenie W typowych systemach przetwarzania informacji system zarządzania danymi może zostać wydzielony. Niestety, relacyjne systemy zarządzania bazą danych nie są na tyle wydajne, aby podołać zadaniu bieżącego przetwarzania sygnałów w systemie monitorującym. Główny nurt badań nad stworzeniem modelowego systemu zarządzania danymi dla potrzeb systemów monitorujących jest związany ze strumieniowym modelem danych. W większości systemy te funkcjonują jednak niedeterministycznie. W artykule przedstawiono opracowane w ramach prac badawczych deterministyczne metody przetwarzania strumieni danych dla potrzeb zadań przetwarzania sygnałów w medycznych systemach zarządzania danymi. Przedstawiono opracowane w ramach prac badawczych twierdzenia opracowanej algebry ciągów danych (strumieni) wraz z wyprowadzonymi formalnymi dowodami. Udowodniono bezpośredni związek niektórych operatorów z odkrytymi w połowie zeszłego wieku twierdzeniami Beatty oraz Fraenkela.

1 Wstęp

Obecnie dostępne systemy zarządzania danymi przetwarzają informacje w oparciu o reguły rachunku relacji. Ten sposób przetwarzania informacji umożliwia opis dowolnych obiektów (Encji) oraz związku pomiędzy nimi (Relacji). Opis struktur danych przy pomocy diagramów Encji-Relacji [1] stał się obowiązkowym standardem w dziedzinie projektowania systemów zarządzania danymi.

W typowych systemach przetwarzania informacji system zarządzania danymi może zostać wydzielony. W przypadku systemów, do których dane napływają z zewnętrznych źródeł (np. system monitorujący) relacyjne systemy zarządzania danymi okazały się niedostatecznie wydajne [2]. Poszukiwania aktywnych lub sekwencyjnych [3] rozszerzeń poszerzyły dziedzinę wiedzy nie zapewniając jednak dostatecznie efektywnego rozwiązania problemu. Na przeszkodzie w podniesieniu wydajności stoi utrwalony paradygmat relacyjny [1].

Obecnie główny nurt badań nad stworzeniem modelowego systemu zarządzania danymi dla potrzeb systemów monitorujących jest związany ze strumieniowym modelem danych oraz systemami zarządzania strumieniami danych [4]. Oczekuje się, że zastosowanie go w zadaniu monitorowania pozwoli na prostsze i bardziej eleganckie wyrażenie algorytmów oraz dzięki uproszczeniu konstrukcji uzyskany zostanie znaczący wzrost efektywności przetwarzania informacji.

Podstawową różnicą pomiędzy systemem relacyjnym a strumieniowym jest istnienie w systemie strumieniowym tzw. Ciągłych zapytań [5]. Przez ciągle zapytanie rozumiemy zapytanie, którego plan realizacji zamknięty jest w martwej pętli. W systemie relacyjnym tego typu funkcjonalność osiąga się poprzez stosunkowo skomplikowany i mało efektywny proces odpytywania bazy danych przez aplikację lub zastosowanie wielu wyzwalaczy. Kolejną różnicą jest sposób wprowadzania danych do systemu. W systemie relacyjnym dane są wprowadzane i zatwierdzane przez użytkownika, natomiast w systemie strumieniowym dane wprowadzane są z zewnętrznych, zazwyczaj autonomicznych podsystemów. Dodatkowo strumień danych ma charakter nieskończony. Tradycyjne systemy zarządzania danymi operują z reguły na danych, dla których oszacowano górną granicę koniecznych zasobów. Operowanie na zbiorach nieskończonych wymaga więc modyfikacji niektórych operatorów.

1.1 System zarządzania danymi

Informatyczne systemy monitorujące stosowane w medycynie na bieżąco rejestrują, analizują i prezentują informacje dostarczane przez aparaturę [6]. Celem prowadzonych w prac badawczych było opracowanie systemu zarządzania danymi oraz języka zapytań umożliwiającego formułowanie i realizację zadań

* Praca finansowana ze środków Ministerstwa Nauki i Informatyzacji w latach 2003-2005 jako projekt badawczy 4 T11F 002 25

przetwarzania sygnałów. Opracowano algebrę [7] oraz zaimplementowano język zapytań w systemie zarządzania danymi [8]. Powstałe rozwiązanie umożliwia efektywne przetwarzanie sygnałów dla potrzeb medycznych systemów monitorujących oraz umożliwia prostą realizację i zapis równoległych algorytmów przetwarzania sygnałów.

1.2 Projekty systemów strumieniowych

Można obecnie wskazać kilka wiodących projektów badawczych obejmujących zagadnienia systemów strumieniowych [9, 10]. Projekt STREAM [1] jest prowadzony na uniwersytecie Stanford. Projekt Aurora, a obecnie Borealis [11, 12] jest prowadzony na uniwersytecie MIT. Autorzy projektu STREAM poszukują ogólnych reguł przetwarzania strumieni danych w oparciu o rozwinięcie algebry relacyjnej. Dla potrzeb opracowanego modelu zmodyfikowano definicję relacji. Relację R reprezentuje zmienny w czasie zbiór krotek. Zapis relacji przedstawiony został w postaci $R(t)$. Istniejące operatory nazwano relacyjno-relacyjnymi natomiast do operowania na strumieniach danych wprowadzono zespół operatorów relacyjno-strumieniowych i strumieniowo-relacyjnych. Na uwagę zasługuje fakt że nie wprowadzono operatorów strumieniowo-strumieniowych. W przypadku projektu Borealis nie rozpatruje się jawnie zagadnienia połączenia zbioru relacji ze zbiorem strumieni danych, wprowadzone operatory umożliwiają jedynie operacje na strumieniach danych. Projekt prowadzony na uniwersytecie MIT doczekał się komercyjnej implementacji.

2 Algebry strumieni danych

Algebra jest jednym z działów matematyki, natomiast pojęcie algebry ogólnej (lub po prostu algebry) dotyczy struktury matematycznej złożonej ze zdefiniowanego zbioru danych oraz pewnej liczby operacji zdefiniowanych na tym zbiorze danych. Przykładem definicji prostej algebry może być następująca struktura matematyczna: $A_1 := (N, (+, -))$. Algebra A_1 została zdefiniowana jako zbiór liczb naturalnych na których możemy jedynie wykonywać operacje dodawania i odejmowania.

W przypadku algebr opartych na strumieniach danych [11, 4] przyjęto za model danych parę uporządkowaną, której pierwszym elementem jest krotka (krotka - uogólnienie pojęcia pary dla dowolnej liczby elementów) a drugim czas jej wystąpienia - (s, t) .

2.1 Operacje na strumieniach danych

Prowadząc obliczenia na strumieniach danych zakłada się istnienie danych o następującym charakterze:

- dane nadchodzą w postaci sekwencji elementów,
- każdy element jest przeglądany i przetwarzany tylko raz.

Głównym problemem jest zapewnienie przestrzeni zasobów koniecznych do przetworzenia pojedynczego elementu. Zapotrzebowanie na pamięć operacyjną i czas realizacji zadania w przypadku idealnym powinno być niezależne od wielkości strumienia danych. System wyposażony w ograniczone zasoby uniemożliwia wyznaczenie dokładnej odpowiedzi na zadane zapytanie w oparciu o nieskończony w czasie i wymiarze strumień danych. W takim przypadku za zadowalające przyjmowane są odpowiedzi aproksymowane. Rozważa się kilka podstawowych technik tworzenia algorytmów aproksymujących: szkice, przypadkowe próbkowanie, histogramy oraz algorytmy falkowe.

W przypadku zadań przetwarzania sygnałów konieczne jest przedstawienie algorytmów deterministycznych, z tego powodu przedstawione założenia systemów strumieniowych nie mogą zostać bezpośrednio zastosowane w realizacji tych zadań.

3 Przetwarzanie ciągów danych

Potrzebę deterministycznego przetwarzania strumieni danych dostrzeżono w trakcie prac nad systemami czasu rzeczywistego. Twórcy rozwijający projekt QStream [13] określili założenia umożliwiające stworzenie systemu zarządzania danymi dla potrzeb tego rodzaju systemów. Zaprezentowane wyniki prac badawczych [14, 7] idą o krok dalej. Zaproponowano deterministyczną algebrę strumieni danych opartą na matematycznych podstawach teorii liczb a w szczególności układów pokrywających (ang. *Covering System*). Rozważanym problemem jest sposób wyznaczania podziału zbioru dodatnich liczb naturalnych. Dwie sekwencje dokonują podziału zbioru liczb naturalnych jeśli ich przekrój jest zbiorem pustym, natomiast ich suma tworzy zbiór liczb naturalnych. Formalne sformułowanie twierdzenia będącego podstawą badań nad tym problemem podał S. Beatty w 1926r [15].

Twierdzenie 1 (Beatty). *Jeśli p, q są dodatnimi liczbami niewymiernymi i zachodzi pomiędzy nimi zależność $\frac{1}{p} + \frac{1}{q} = 1$ to sekwencje $\{\lfloor np \rfloor\}_{n=1}^{\infty} = \lfloor p \rfloor, \lfloor 2p \rfloor, \lfloor 3p \rfloor, \dots$ oraz $\{\lfloor nq \rfloor\}_{n=1}^{\infty} = \lfloor q \rfloor, \lfloor 2q \rfloor, \lfloor 3q \rfloor, \dots$ dokonują podziału zbioru dodatnich liczb całkowitych.*

Podstawą prowadzonych rozważań w literaturze jest następująca sekwencja nazywana sekwencją Beatty (podłoga).

$$\mathcal{B}(\alpha, \alpha') := \left(\left\lfloor \frac{n - \alpha'}{\alpha} \right\rfloor \right)_{n=1}^{\infty} \tag{1}$$

lub sekwencją Beatty (sufit):

$$\mathcal{B}^{(c)}(\alpha, \alpha') := \left(\left\lceil \frac{n - \alpha'}{\alpha} \right\rceil \right)_{n=1}^{\infty} \tag{2}$$

Gdzie α oznacza gęstość, $\frac{1}{\alpha}$ oznacza nachylenie, α' oznacza przesunięcie oraz $\frac{-\alpha'}{\alpha}$ y-przechywcenie.

Twierdzenie Beatty umożliwia podział zbioru jedynie w oparciu o dobór dwóch liczb niewymiernych. W 1969 Aviezi S. Fraenkel [16] uogólnił poprzednie twierdzenie przedstawiając sposób tworzenia sekwencji dokonujących podziału zbioru również w oparciu o liczby wymierne.

Twierdzenie 2 (Fraenkel). *Sekwencje $\mathcal{B}(\alpha, \alpha')$ oraz $\mathcal{B}(\beta, \beta')$ dokonują podziału zbioru \mathbb{N} wtedy i tylko wtedy gdy następujące pięć warunków zostanie spełnionych:*

1. $0 < \alpha < 1$.
2. $\alpha + \beta = 1$.
3. $0 \leq \alpha + \alpha' \leq 1$.
4. *Jeśli α jest liczbą niewymierną, wtedy $\alpha' + \beta' = 0$ i $k\alpha + \alpha' \notin \mathbb{Z}$ dla $2 \leq k \in \mathbb{N}$.*
5. *Jeśli α jest liczbą wymierną, (niech $q \in \mathbb{N}$ będzie najmniejszą liczbą taką że $q\alpha \in \mathbb{N}$) wtedy $\frac{1}{q} \leq \alpha + \alpha'$ i $\lceil q\alpha' \rceil + \lceil q\beta' \rceil = 1$.*

Dowód twierdzenia można odnaleźć w literaturze [17]. Twierdzenia przedstawione w dalszej części pracy opierają się na tych badaniach. Przedstawione metody znajdują obecnie zastosowanie w zadaniu przetwarzania sygnałów [18].

3.1 Model danych

Przyjęty w literaturze [19, 4, 20] model danych (s, τ) , umożliwia opis zjawisk, które w tej samej chwili czasu przyjmują kilka różnych stanów. Jest to rozwiązanie nadmiarowe w stosunku do potrzeb opisu strumienia danych, w którym nie występują dwie różne krotki w tej samej chwili czasu. Takie założenie znacząco komplikuje definicje wprowadzanych operacji [21]. Strumień danych dla potrzeb strumieniowego systemu zarządzania danymi opisany jest [7, 22] przez parę uporządkowaną (s_n, Δ_n) , której pierwszym elementem jest ciąg krotek a drugim ciąg wartości wyznaczających ciąg różnic czasu pomiędzy kolejnymi krotkami.

Model danych w postaci (s_n, τ_n) opisuje tą samą klasę zjawisk co model (s, τ) . Natomiast model różnicowy (s_n, Δ_n) uniemożliwia przedstawienie strumienia danych, w którym występują dwie różne krotki w tej samej chwili czasu τ .

Zawężając przedział zastosowania modelu przyjęto [7], że ciąg odstępów czasu pomiędzy kolejnymi krotkami jest wartością stałą. Dzięki temu można określić model danych w następujący sposób: (s_n, Δ) . Taki model danych odpowiadający serii czasowej jest podstawą dalszych rozważań.

Przyjmuje się że relacje oznaczane są literami alfabetu łańciskiego [23]. Dla strumienia przyjęto podobną konwencję. Dodatkowo z każdym strumieniem danych związana jest wartość oznaczająca stały odstęp czasu pomiędzy kolejnymi krotkami. Wartość tą zapisujemy po nazwie strumienia danych po przecinku, np. zapis A,3 oznacza strumień danych A którego krotki przychodzą raz na trzy sekundy. Z każdym strumieniem związane jest pojęcie schematu.

Schemat strumienia to dowolny, uporządkowany zbiór atrybutów opisujących kolejne pola krotek należących do strumienia danych. Analogicznie jak w przypadku modelu relacyjnego, każdy atrybut odpowiada kolumnie danych. Przykładowy identyfikator strumienia w postaci A(imię ,nazwisko),3 oznacza strumień A, którego każda krotka składa się z dwóch atrybutów: imię i nazwisko oraz każda kolejna krotka pojawia się raz na trzy sekundy. Przez schemat jednorodny strumienia danych rozumiemy schemat, którego wszystkie atrybuty są tego samego typu.

3.2 Wprowadzone operatory

Po analizie większości typowych operacji prowadzonych na danych otrzymywanych z urzędzeń medycznych zidentyfikowano następujący zbiór operacji: Przeplot, Rozplątanie, Suma, Różnica, Projekcja, Selekcja, Agregacja i Serializacja (AGSE) oraz Przesunięcie. Przyjmując, że dane są dwa strumienie A, B operacje zostały zdefiniowane w następujący sposób:

Suma i różnica W przypadku łączenia strumieni danych pochodzących z urzędzeń pracujących synchronicznie występuje potrzeba złączenia strumieni danych niejako w naturalny sposób, poprzez złączenie ich schematów. O ile w przypadku strumieni danych, których elementy nadchodzą z identyczną częstością problem sumarycznego złączenia nie sprawia trudności, o tyle w przypadku strumieni danych, których elementy nadchodzą z różną częstością, elementy strumienia nadchodzącego z mniejszą częstością muszą zostać powielone.

Uwzględniając różnice pomiędzy zbiorem relacji oraz zbiorem strumieni danych operację złączenia sumarycznego (sumy) i rozłączenia różnicowego (różnicy) zdefiniowano w następujący sposób: wartością wyrażenia $\Sigma(A, B)$ jest strumień danych. Sposób tworzenia kolejnych krotek strumienia wynikowego oraz wyznaczania wartości Δ tego strumienia opisuje wzór:

$$c_n = \begin{cases} a_n | b_{\lfloor \frac{n\Delta_a}{\Delta_b} \rfloor} & \Delta_c = \Delta_a \\ a_{\lfloor \frac{n\Delta_b}{\Delta_a} \rfloor} | b_n & \Delta_c = \Delta_b \end{cases}, \Delta_c = \min(\Delta_a, \Delta_b) \quad (3)$$

Gdzie:

$\Delta_{a,b,c}$ to wartości określające stały odstęp czasu pomiędzy krotkami w strumieniach A, B oraz C.

n oznacza pozycję kolejnej krotki

a_n, b_n, c_n oznaczają krotki strumieni A, B oraz C.

Część całkowita liczby rzeczywistej x oznaczana jest jako $\lfloor x \rfloor$. Sufit $\lceil x \rceil$ liczby rzeczywistej x to najmniejsza liczba całkowita nie mniejsza od x .

Przykład 1. Zakładając istnienie dwóch strumieni danych:

Alfa(znak), 2: (1, 2, 3, 4, ...) oraz Beta(znak), 1: (a, b, c, d, e, f, g, ...)

to wyrażenie: $\Sigma(\text{Alfa}, \text{Beta})$ opisuje strumień danych o postaci:

Omega(znak, znak), 1: ((1, a), (1, b), (2, c), (2, d), ...)

Operacja różnicy umożliwia odtworzenie strumieni biorących udział w operacji sumowania. Należy jednak podkreślić, że nie jest to operacja projekcji [1, 23]. Krotki, które w wyniku sumowania strumieni danych o różnych częstościach zostały powielone, są w wyniku operacji rozłączenia różnicowego usuwane. Dopiero zastosowanie operatora projekcji umożliwia odtworzenie pierwotnej postaci strumienia danych.

Argumentem tej operacji jest strumień danych, (który w domyśle został poddany operacji sumowania) oraz para dwóch liczb, przedstawiających stosunek odstępów czasu dwóch pierwotnych strumieni danych. Sposób tworzenia kolejnych krotek strumienia wynikowego opisuje:

$$a_n = \begin{cases} c_n & \Delta_b \geq \Delta_a \\ c_{\lfloor \frac{n\Delta_a}{\Delta_b} \rfloor} & \Delta_b < \Delta_a \end{cases} \quad (4)$$

Symboliczny zapis tej operacji przedstawia się następująco $\delta_{a,b}(S)$. Gdzie a, b oznaczają parę liczb określających stosunek odstępów czasu pierwotnych strumieni danych. S jest strumieniem, argumentem operacji różnicy.

Przykład 2. Zakładając istnienie strumienia danych Omega o postaci przedstawionej poprzednio to wyrażenie $\delta_{1,2}(\text{Omega})$ opisuje strumień danych o postaci:

Alfa2(znak, znak), 2: ((1, a), (2, c), (3, e), ...)

Przeplot i rozplątanie Operacje sumarycznego splątania (przeplotu) i rozplątania różnicowego (rozplątania) tworzą ortogonalny zbiór operatorów w stosunku do operacji sumy i różnicy. O ile w przypadku sumy i różnicy schematy łączonych strumieni danych mogą być różne, o tyle w przypadku operacji przeplotu i rozplątania schematy łączonych strumieni danych muszą być ze sobą zgodne. Dwa schematy strumieni danych są ze sobą zgodne jeśli odpowiadające sobie kolejne typy pól są takie same.

Jeśli częstość obu strumieni jest identyczna, wynikowy strumień danych po przeplocie zawiera na przemian krotki ze strumieni będących argumentami operacji. Jeśli częstość argumentów jest różna to kolejność, w jakiej umieszczane są po sobie krotki wyznacza reguła (5) określająca operację przepłotu.

Operacja ta zapisywana jest w następujący sposób $\phi(A, B)$. Następujący wzór przedstawia sposób wyznaczania kolejnych krotek strumienia wynikowego:

$$c_n = \begin{cases} b_{n-\lfloor nz \rfloor} & \lfloor nz \rfloor = \lfloor (n+1)z \rfloor \\ a_{\lfloor nz \rfloor} & \lfloor nz \rfloor \neq \lfloor (n+1)z \rfloor \end{cases}, z = \frac{\Delta_b}{\Delta_a + \Delta_b}, \Delta_c = \frac{\Delta_a \Delta_b}{\Delta_a + \Delta_b} \quad (5)$$

Twierdzenie 3. *Operacja splątania zapewnia sekwencyjne pokrycie obu zbiorów zawierających elementy strumieni danych.*

Dowód. W pierwszym kroku analizowany jest pierwszy warunek (równości) w równaniu (5). Dla każdego n spełniającego warunek^(*) $\lfloor nz \rfloor = \lfloor (n+1)z \rfloor$, kolejne wartości wyrażenia $n - \lfloor nz \rfloor$ tworzą drugi (skojarzony) ciąg liczb naturalnych wybierających kolejne elementy z ciągu b_n .

Czyli dla każdego n spełniającego warunek^(*) powinna zachodzić dla wyrażenia $n - \lfloor nz \rfloor$ zależność $x_n = x_{n+1} - 1$. Formalnie zapis tego zdania przedstawia się następująco:

$$n - \lfloor nz \rfloor = (n+1) - \lfloor (n+1)z \rfloor - 1 \quad (6)$$

Podstawiając warunek^(*) do powyższej zależności otrzymujemy:

$$n - \lfloor (n+1)z \rfloor = (n+1) - \lfloor (n+1)z \rfloor - 1 \quad (7)$$

Poprzez proste algebraiczne uproszczenie dochodzimy do następującej tożsamości: $n = (n+1) - 1$

Drugą część dowodu (w oparciu o warunek nierówności) prowadzi się analogicznie. □

Przykład 3. Przy założeniu istnienia strumieni danych Alfa oraz Beta, wyrażenie $\phi(Alfa, Beta)$ opisuje przepłot dwóch strumieni danych tworzących strumień wynikowy o postaci:

Gamma (znak), 2/3: (a, b, 1, c, d, 2, e, f, 3, ...)

Natomiast wyrażenie $\phi(Beta, Alfa)$ opisuje strumień wynikowy o postaci:

Delta (znak), 2/3: (1, a, b, 2, c, d, 3, e, f, ...)

Jak widać operacja przepłotu nie jest przemiana .

Rozplątanie jest operacją komplementarną w stosunku do operacji przepłotu. Rozplątując strumień można wybrać te krotki ze strumienia, które należały do jednego z argumentów operacji splątania. Argumentami operacji są: strumień danych oraz liczba określająca odstęp czasu. Dla formalności przyjęto dwa operatory określające tę operację: $\Theta_n(A)$ oraz $\sim \Theta_n(A)$. Pierwszy z nich przedstawia operację uzyskania pierwotnego strumienia danych, a drugi „reszty” z operacji rozplątania. Operację rozplątania definiują dwa wzory:

$$a_n = c_{n+\lfloor \frac{(n+1)\Delta_a}{\Delta_b} \rfloor}, \Delta_a = \frac{\Delta_c \Delta_b}{|\Delta_c - \Delta_b|} \quad (8)$$

oraz

$$b_n = c_{n+\lfloor \frac{n\Delta_b}{\Delta_a} \rfloor}, \Delta_b = \frac{\Delta_c \Delta_a}{|\Delta_c - \Delta_a|} \quad (9)$$

Twierdzenie 4. *Operacja rozplątania spełnia postulaty twierdzenia Fraenkela.*

Dowód. W pierwszej części dowodu poszukiwana jest sekwencja Beatty przedstawiona w sposób (9) zaprezentowany w definicji operacji rozplątania. Zapis przy pomocy tej notacji przedstawia się następująco:

$$\left(n + \left\lfloor \frac{nb}{a} \right\rfloor \right)_{n=1}^{\infty} \quad (10)$$

Jeśli $n \in \mathbb{N}$ to wtedy na mocy własności (21) można powyższe równanie zapisać:

$$\left(\left\lfloor \frac{n - \alpha'}{\alpha} \right\rfloor \right)_{n=1}^{\infty} = \left(\left\lfloor n + \frac{nb}{a} \right\rfloor \right)_{n=1}^{\infty} \quad (11)$$

Upraszczając,

$$\left(\left\lfloor n\alpha^{-1} - \frac{\alpha'}{\alpha} \right\rfloor \right)_{n=1}^{\infty} = \left(\left\lfloor n \frac{a+b}{a} \right\rfloor \right)_{n=1}^{\infty} \quad (12)$$

Symbol $-\frac{\alpha'}{\alpha}$ oznacza y-przechwycenie, jeśli wartość przesunięcia sekwencji $\alpha' = 0$ wtedy $\alpha = \frac{a}{a+b}$, czyli sekwencja (10) może zostać przedstawiona w następujący sposób:

$$\mathcal{B}\left(\frac{a}{a+b}, 0\right) := \left(\left\lfloor n \frac{a+b}{a} \right\rfloor\right)_{n=1}^{\infty} \quad (13)$$

Poprzez kilka prostych przekształceń algebraicznych z sekwencji opisującej wybór kolejnych krotek w operacji rozplątania (9) otrzymano postać sekwencji Beatty. W kolejnej części dowodu na mocy postulatów twierdzenia Fraenkela określamy residuum sekwencji (10).

1. Wartość wyrażenia $\alpha = \frac{a}{a+b}$ dla $a, b > 0$ jest mniejsza od jedności i większa od zera.
2. Warunek $\alpha + \beta = 1$ jest spełniony dla $\beta = \frac{b}{a+b}$.
3. Dla $\alpha' = 0$ postulat jest równoważny 1.
4. Poszukujemy rozwiązań w zbiorze liczb wymiernych.
5. Jeśli $q\alpha \in \mathbb{N}$ i $q \in \mathbb{N}$ oraz $\frac{1}{q} \leq \alpha + \alpha'$ to skoro $\alpha' = 0$ to $\frac{1}{q} \geq \frac{a}{a+b}$ a to jest prawdą dla $q \leq \frac{a+b}{nwd(a,b)}$ z tego wynika że $\left\lceil \frac{a+b}{nwd(a,b)} \beta' \right\rceil = 1$. Czyli $\beta' = \frac{nwd(a,b)}{a+b}$.

W wyniku przekształceń otrzymamy postać ciągu przedstawioną w równaniu (9) opisującym operację rozplątania. Postać ciągu residuum dla $\mathcal{B}\left(\frac{a}{a+b}, 0\right)$ spełniająca postulaty twierdzenia Fraenkela przedstawia się następująco:

$$\mathcal{B}\left(\frac{b}{a+b}, \frac{nwd(a,b)}{a+b}\right) = \left(\left\lfloor \frac{(n+1) - \frac{nwd(a,b)}{a+b}}{\frac{b}{a+b}} \right\rfloor\right)_{n=1}^{\infty} \quad (14)$$

Opuszczając nawiasy opisujące sekwencję po kilku prostych przekształceniach można wykazać, że:

$$\left\lfloor \frac{(n+1) - \frac{nwd(a,b)}{a+b}}{\frac{b}{a+b}} \right\rfloor := \left\lfloor n \frac{a}{b} + n + \frac{a}{b} + 1 - \frac{nwd(a,b)}{b} \right\rfloor \quad (15)$$

Przyrównując wyznaczoną postać residuum do przyjętej sekwencji (8) otrzymujemy równanie którego prawdziwość należy udowodnić:

$$\left\lfloor n \frac{a}{b} + n + \frac{a}{b} + 1 - \frac{nwd(a,b)}{b} \right\rfloor = n + \left\lfloor \frac{(n+1)a}{b} \right\rfloor \quad (16)$$

Stąd można wykazać, że:

$$\left\lfloor \frac{(n+1)a}{b} - \frac{nwd(a,b)}{b} \right\rfloor + 1 = \left\lfloor \frac{(n+1)a}{b} \right\rfloor \quad (17)$$

Podstawiając za $n+1$ liczbę naturalną n otrzymujemy:

$$\left\lfloor n \frac{a}{b} - \frac{nwd(a,b)}{b} \right\rfloor + 1 = \left\lfloor n \frac{a}{b} \right\rfloor \quad (18)$$

W dalszej części dowodu zastosowanie znajdujące następujące własności operacji podłoga oraz sufit:

$$\lfloor x \rfloor = \lceil x \rceil \Leftrightarrow x \in \mathbb{N} \quad (19)$$

$$\lfloor x \rfloor + 1 = \lceil x \rceil \Leftrightarrow x \in \mathbb{R} \setminus \mathbb{N} \quad (20)$$

$$\lfloor x + C \rfloor = \lfloor x \rfloor + C \Leftrightarrow c \in \mathbb{N} \quad (21)$$

Uwzględniając własności $nwd(a, b)$ dowód można rozważać w oparciu o następujące dwa przypadki:

$$nwd(a, b) = b \Leftrightarrow \frac{a}{b} = c \in \mathbb{N} \quad (22)$$

oraz

$$1 \leq nwd(a, b) \leq a \Leftrightarrow 0 < \frac{a}{b} < 1 \quad (23)$$

Uwzględniając przypadek (22), równanie (18) można zapisać w postaci:

$$\left\lfloor \frac{(n+1)a}{b} - \frac{b}{b} \right\rfloor + 1 = \left\lceil \frac{(n+1)a}{b} \right\rceil \tag{24}$$

Uwzględniając własności (19,21) oraz dziedzinę dla tego przypadku (22) można stwierdzić, że obie sekwencje tworzą te same elementy.

Rozważając kolejny przypadek (23) można założyć że istnieją takie dwie liczby a i b , dla których równanie (18) nie jest fałszywe, czyli dla:

$$n \frac{a}{b} - \frac{nwd(a,b)}{b}, n \frac{a}{b} \in \mathbb{N} \tag{25}$$

Korzystając z własności (19,21) nie zachodzi:

$$\left\lfloor n \frac{a}{b} - \frac{nwd(a,b)}{b} + 1 \right\rfloor \neq \left\lceil n \frac{a}{b} \right\rceil \tag{26}$$

Korzystając z własności (19) poszukiwane są takie a i b dla których zachodzi własność:

$$n \frac{a}{b} - \frac{nwd(a,b)}{b} + 1 \neq n \frac{a}{b} \tag{27}$$

Równanie (18) jest spełnione dla $nwd(a,b) = b$, a w rozważanej dziedzinie (23) $1 \leq nwd(a,b) \leq a$ równanie to nie ma rozwiązań. Nie istnieją takie a i b należące do dziedziny (25) które by przeczyły rozważanemu równaniu (18).

Analizując drugą własność (20) można założyć że istnieją dwie takie liczby a i b że równanie (18) nie jest spełnione. Czyli dla a i b takich, że:

$$n \frac{a}{b} - \frac{nwd(a,b)}{b}, n \frac{a}{b} \in \mathbb{R} \setminus \mathbb{N} \tag{28}$$

Powinna zawsze zachodzić:

$$n \frac{a}{b} - \frac{nwd(a,b)}{b} \neq n \frac{a}{b} \tag{29}$$

Nie istnieją jednak dwie takie liczby, dla których $nwd(a,b) = 0$

Dla $\frac{a}{b} \in \mathbb{R} \setminus \mathbb{N}$ równanie to jest zawsze prawdziwe.

Tak więc oba równania (8,9) są przypadkiem sekwencjami Beatty spełniającymi postulaty twierdzenia Fraenkela dla liczb wymiernych. □

Przykład 4. Zakładając istnienie przedstawionego w poprzednim przykładzie strumienia $\text{Gamma}, 2/3$ to wyrażenie $\Theta_1(\text{Gamma})$ opisuje strumień danych:

Alfa(znak), 2: (1, 2, 3, 4, ...)

Natomiast wyrażenie $\Theta_2(\text{Gamma})$ opisuje strumień danych

Beta(znak), 1: (a, b, c, d, e, f, g, ...)

Rzutowanie Operator rzutowania (projekcji) wykorzystywany jest przy tworzeniu nowego strumienia, który powstaje ze strumienia A poprzez usunięcie lub zmianę kolejności kolumn. Wartością wyrażenia $\pi_{T_1, T_2, \dots, T_n}(A)$ jest strumień, który powstaje ze strumienia A poprzez przepisanie wartości wszystkich atrybutów T_1, T_2, \dots, T_n . Operacja rzutowania nie wpływa na wartość stałego odstępu czasu pomiędzy kolejnymi krotkami w strumieniu. Wartość Δ po operacji rzutowania pozostaje niezmienną.

Przykład 5. Zakładając istnienie przedstawionego strumienia danych Alfa2 wyrażenie $\pi_{Znak}(\text{Alfa2})$ opisuje strumień danych o postaci

Alfa(znak), 2: (1, 2, 3, ...)

Selekcja W wyniku zastosowania operatora selekcji na strumieniu A powstaje nowy strumień, do którego należy pewien podzbiór krotek strumienia A . Krotki strumienia wynikowego są wybierane według kryterium określonego przez pewien warunek W . Tę operację, bardzo podobnie jak w przypadku algebry relacji, oznaczono symbolem $\sigma_{W,\delta}(A)$. Schemat strumienia wynikowego po operacji selekcji pozostaje bez zmian. Porządek atrybutów również zostaje zachowany. W stanowi wyrażenie warunkowe oznaczające prawdę lub fałsz. Operandami mogą być stałe lub atrybuty strumienia A . Jeśli wyrażenie przyjmuje wartość prawdę, to wówczas krotka jest dołączana do strumienia wynikowego, w przeciwnym wypadku

krotka nie zostanie dołączona. Operacja selekcji wpływa bezpośrednio na wartość stałego odstępu czasu pomiędzy kolejnymi krotkami w strumieniu. Wartość Δ strumienia wynikowego jest określana z góry i przyjmuje wartość δ .

Przykład 6. Zakładając istnienie strumienia $\Omega(\text{znak}, 1)$ wyrażenie: $\sigma_{\text{is_number}(\text{znak}), 2}(\Omega)$ opisuje strumień $\text{Alfa}(\text{znak}, 2)$.

Agregacja i serializacja Akronim AGSE powstał ze złożenia pierwszych sylab słów Agregacja oraz Serializacja. Jest to operacja, której argumentem jest jeden strumień danych o jednorodnym schemacie. W wyniku tej operacji powstaje kolejny strumień danych o zadanym schemacie oraz innej, wyznaczonej wartości Δ . Operację oznacza się symbolem $\Psi_{n,\lambda}(A)$. Symbol λ oznacza krok, z jakim przesuwane jest ruchome okno danych nad strumieniem. Jest on wyrażony w liczbie krotek. Symbol n oznacza szerokość tego okna danych wyrażoną w ilości atrybutów schematu danych.

Przykład 7. Jeśli istnieje strumień $\text{Alfa}(\text{znak}, 1)$ wyrażenie $\Psi_{2,2}(\text{Alfa})$ opisuje nowy strumień o postaci: $\text{Agse1}(\text{znak}, \text{znak}), 2$ którego elementy tworzą ciąg: $((a, b), (c, d), (e, f), \dots)$. Natomiast wyrażenie $\Psi_{1,1}(\text{Beta})$ opisuje strumień $\text{Beta}(\text{znak}, 1)$. Pierwsza operacja jest przykładem agregacji, natomiast druga jest przykładem serializacji.

Przesunięcie Operator przesunięcia to podstawowy operator stosowany w zapisie filtrów sygnałowych. Umożliwia on odwołanie się do danych archiwalnych. Jest to operacja w wyniku, której tworzony jest strumień danych, którego elementy są przesunięte względem argumentu o zadaną liczbę krotek. Operację oznaczono symbolem $\tau_n(A)$. Symbol n oznacza liczbę krotek, o jaką należy opóźnić strumień wynikowy w stosunku do argumentu A .

Przykład 8. Jeśli istnieje strumień $\text{Alfa}(\text{znak}, 1)$ wyrażenie: $\tau_2(\text{Alfa})$ definiuje strumień $\text{Shift2}(\text{znak}), 1: (3, 4, 5, 6, \dots)$

4 Własności operacji

W przypadku optymalizacji czasowej planów realizacji zapytań stosuje się m.in. regułę przenoszenia operacji selekcji ponad operacje złączeń [24]. Podobne zagadnienia są rozważane w przypadku systemów strumieniowych [25]. W trakcie prac nad deterministycznym sposobem optymalizacji czasowej odkryto kilka własności oraz metod algebraicznego upraszczania wyrażeń strumieniowych. Pierwsza z własności dotyczy zaburzenia kolejności elementów. Następna określa stosunkowo prostą zasadę przemienności sumowania. Ostatnia metoda, określona jako metoda dopasowania przepływu umożliwia zamianę kolejności atrybutów operacji przepływu przy pewnych warunkach.

4.1 Zaburzenie kolejności zdarzeń

Twierdzenie 5. *Kolejność elementów w strumieniu nie zawsze odzwierciedla faktyczną kolejności występowania elementów w świecie rzeczywistym*

Dowód. Tą własność można w prosty sposób odkryć, analizując operację przepływu zastosowaną na następujących strumieniach danych:

$\text{Alfa}(\text{znak}), 2: (1, 2, 3, 4, 5, 6, \dots)$

$\text{Epsilon}(\text{znak}), 3: (a, b, c, d, e, f, \dots)$

Wyrażenie $\phi(\text{Epsilon}, \text{Alfa})$ opisuje przepływ strumieni danych o postaci:

$\text{Tau}(\text{znak}), 6/5: (1, 2, a, 3, b, 4, 5, c, 6, d, \dots)$

W strumieniu Tau krotka oznaczona literą c występuje po krotce oznaczonej cyfrą 5. Tymczasem w strumieniu Epsilon krotka oznaczona literą c występuje w 9 sekundzie, a w strumieniu Alfa krotka oznaczona cyfrą 5 występuje w 10 sekundzie. Naturalny porządek zdarzeń został w strumieniu wynikowym naruszony. Należy wyciągnąć następujący wniosek: Kolejność krotek w strumieniu danych nie zawsze odpowiada kolejności ich występowania w świecie rzeczywistym. Prowadząc analizę względem czasu zawartego w strumieniach danych konieczne jest uwzględnienie tego aspektu i zastosowanie w tym przypadku operacji rozplątania w celu uzyskania pierwotnej postaci strumieni danych. \square

4.2 Przemienność sumowania

Twierdzenie 6. *Operacja sumowania strumieni danych, nie uwzględniając kolejności atrybutów sumowanych strumieni danych jest przemienna.*

Dowód. Załóżmy, że: $C = \Sigma(A, B)$ oraz $C = \Sigma(S, A)$. Korzystając ze wzoru na sumę strumieni danych można przedstawić dwie zależności:

$$C = \Sigma(A, B) \quad c_n = \begin{cases} a_n | b_{\lfloor \frac{n\Delta_a}{\Delta_b} \rfloor} & \Delta_a < \Delta_b \\ a_{\lfloor \frac{n\Delta_b}{\Delta_a} \rfloor} | b_n & \Delta_a > \Delta_b \end{cases} \quad (30)$$

oraz

$$D = \Sigma(S, A) \quad d_n = \begin{cases} s_n | a_{\lfloor \frac{n\Delta_s}{\Delta_a} \rfloor} & \Delta_s < \Delta_a \\ s_{\lfloor \frac{n\Delta_a}{\Delta_s} \rfloor} | a_n & \Delta_s > \Delta_a \end{cases} \quad (31)$$

Pomijając kolejność atrybutów wynikającą z operacji połączenia elementów krotek oraz zmieniając kolejność warunków można wyprowadzić następujący wzór:

$$D = \Sigma(S, A) \quad d_n = \begin{cases} a_n | s_{\lfloor \frac{n\Delta_a}{\Delta_s} \rfloor} & \Delta_s > \Delta_a \equiv (\Delta_a < \Delta_s) \\ a_{\lfloor \frac{n\Delta_s}{\Delta_a} \rfloor} | s_n & \Delta_s < \Delta_a \equiv (\Delta_a > \Delta_s) \end{cases} \quad (32)$$

Podstawiając we wzorze (32) za symbol S symbol B dowodzimy poprawności twierdzenia o przemienności operacji sumowania strumieni danych. Istnieje jeszcze jeden, nieuwzględniony przypadek - kiedy Δ obu strumieni danych jest równa. Dowód nie został zaprezentowany z uwagi na trywialność. \square

4.3 Metoda dopasowania przeplotu

Jak pokazano na przykładzie operacji przeplotu, nie jest to operacja przemienna. Jednak istnieje algebraiczna metoda umożliwiająca zmianę kolejności atrybutów operacji przeplotu przy pewnych założeniach.

Twierdzenie 7. *Jeśli wybierzemy dwie liczby naturalne i, k , których stosunek jest równy stosunkowi wartości elementów Δ_n strumieni łączonych operacją przeplotu to operacja przeplotu strumieni danych przesuniętych względem tych wartości tworzy strumień danych równy strumieniowi powstałemu poprzez przeplot, przesunięcie względem sumy wartości tych liczb i zamianę kolejności argumentów tej operacji. Formalnie: Jeśli spełnione są następujące warunki: $\frac{i}{k} = \frac{\Delta_a}{\Delta_b}$ oraz $i, k \in \mathbb{N}$ to:*

$$\phi(\tau_i(A), \tau_k(B)) = \tau_{i+k}(\phi(B, A)) \quad (33)$$

Dowód. Analizując lewą stronę równania (33) oraz biorąc pod uwagę równanie opisujące operację przeplotu (5) można otrzymać następujący wzór:

$$C = \phi(\tau_i(A), \tau_k(B)) \quad c_n = \begin{cases} b_{(n-\lfloor nz \rfloor)+i} & \lfloor nz \rfloor = \lfloor (n+1)z \rfloor \\ a_{\lfloor nz \rfloor+k} & \lfloor nz \rfloor \neq \lfloor (n+1)z \rfloor \end{cases} \quad (34)$$

Analizując prawą stronę równania (33) oraz biorąc pod uwagę równanie opisujące operację przeplotu (5), można otrzymać następujący wzór:

$$C = \tau_{i+k}(\phi(B, A)) \quad c_n = \begin{cases} a_{\lfloor (n+i+k)z \rfloor} & \lfloor nz \rfloor = \lfloor (n+1)z \rfloor \\ b_{n+i+k-\lfloor (n+i+k)z \rfloor} & \lfloor nz \rfloor \neq \lfloor (n+1)z \rfloor \end{cases} \quad (35)$$

Rozważając warunki, dla których równania dobiegają odpowiednie próbki ze strumieni A i B oraz korzystając z założeń twierdzenia (33) możemy stwierdzić, że:

$$b_{n-\lfloor nz \rfloor+i} = b_{n+i+k-\lfloor (n+i+k)z \rfloor} \implies -\lfloor nz \rfloor = k - \lfloor (n+i+k)z \rfloor \quad (36)$$

oraz:

$$i + k = \frac{\Delta_a}{\Delta_b} k + k = k \left(\frac{\Delta_a}{\Delta_b} + 1 \right) = \frac{k}{z} \quad (37)$$

Mając na uwadze założenia tezy (33) oraz poprzednie równania można przedstawić następujący wniosek:

$$-\lfloor nz \rfloor = k - \lfloor k + nz \rfloor \quad (38)$$

Biorąc pod uwagę założenie tezy (33) tzn. $k \in \mathbb{N}$ można stwierdzić że równanie (38) jest spełnione. Druga część dowodu, prowadzona w oparciu o warunek nierówności, jest analogiczna. \square

5 Budowa planów zapytań

W poprzednich punktach przedstawiono zbiór operatorów oraz metody umożliwiające poprawę efektywności dostępu do danych. Efektywnością realizacji zarządza procesor zapytań. Procesor zapytań jest elementem systemu zarządzania danymi, który służy do przetłumaczenia zapytań wyrażonych w języku formalnym na ciąg operacji na bazie danych. Błędna strategia realizacji zapytań może prowadzić do powstania algorytmów, które przetwarzają zapytania w znacznie dłuższym czasie niż jest to niezbędne. W większości relacyjnych systemów zarządzania danymi w celu wyrażenia wewnętrznej reprezentacji zapytań zapisanych w języku SQL stosowana jest notacja algebry relacji. Zaletą algebry relacji jest możliwość reprezentacji wyrażeń algebraicznych w postaci drzewa wyrażeń opisujących logiczny plan zapytania [24].

Podobne zasady przyjęto podczas tworzenia procesora zapytań dla potrzeb tworzonego systemu. W celu wyrażenia wewnętrznej reprezentacji zapytań stosowana jest przedstawiona w pracy algebra strumieni danych. Zdefiniowany dla potrzeb systemu zbiór operacji różni się zbioru operacji zdefiniowanych w algebrze relacji jednak podobnie jak w przypadku algebry relacji przewidziano możliwość reprezentacji wyrażeń algebraicznych w postaci drzewa wyrażeń. Aby stworzyć drzewo wyrażeń procesor zapytań wykonuje szereg etapów [24]. Najpierw należy przeanalizować składniowo zapytanie zapisane w deklaratywnym języku zapytań. Do opisu języków programowania stosuje się gramatyki formalne, a w szczególności gramatyki bezkontekstowe [26]. Pozwalają one na przejrzystą reprezentację struktury zapytania w postaci drzewa rozbioru składniowego oraz umożliwiają stworzenie algorytmu, który będzie akceptował poprawne zdania. Następnie należy przekształcić drzewo składniowe w drzewo wyrażeń algebraicznych wg przyjętej algebry danych. Na tym etapie system zarządzania danymi decyduje o sposobie wykonania zapytania wybierając najbardziej efektywny sposób realizacji zadania. Stworzone drzewo wyrażeń nazywane jest też logicznym planem zapytania. W ostatniej fazie logiczny plan zapytania należy przekształcić w fizyczny plan zapytania, który wskazuje nie tylko wykonywane operacje, ale również kolejność ich wykonywania oraz algorytm stosowany do realizacji każdej z nich, a także sposoby przekazywania danych pomiędzy kolejnymi krokami algorytmu.

5.1 Drzewa wyrażeń

Jedno wyrażenie algebraiczne może zawierać wiele operatorów. Działają one kaskadowo na wynikach poprzednich działań. Zatem można przedstawić kolejność wykonywania operatorów w postaci drzewa wyrażeń. Liśćmi są nazwy strumieni danych, każdy węzeł wewnętrzny jest etykietowany operatorem.

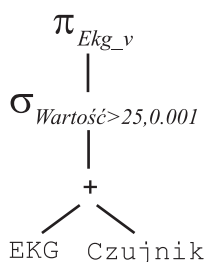
Założmy że są dane dwa strumienie danych: `EKG(pacjent_id, ekg_v), 0.001` oraz `Czujnik(pacjent_id, wartość), 0.1`. Pierwszy strumień danych zawiera informacje o zmierzonych potencjałach elektrycznych serca, natomiast drugi strumień zawiera pomiary z czujnika biometrycznego. Zapytanie, którego zadaniem jest stworzenie strumienia danych zawierającego jedynie próbki EKG dla których pole wartość strumienia czujnik jest większe od 25.

```
SELECT ekg_v
STREAM bol_ekg
FROM EKG + Czujnik
FILTER BY wartość > 25, 0.001
```

Tak sformułowane zapytanie jest tłumaczone przez analizator składniowy na logiczny plan zapytania, w którym pierwszy krok polega na połączeniu strumieni `EKG` oraz `Czujnik`. W następnym kroku wykonywana jest selekcja, odpowiadająca klauzuli `FILTER BY`, a ostatni krok to rzutowanie na listę atrybutów umieszczoną w klauzuli `SELECT`. Wyrażenie algebraiczne powyższego zapytania przedstawiono w postaci drzewa na rysunku 1.

Istnieje kilka wyrażeń równoważnych wyrażeniu przedstawionemu na rysunku 1. Równoważnych w tym sensie, że bez względu na wybór planu zapytania ich wyniki są takie same. Ogólna zasada znana z algebry relacyjnej stanowi, że (zazwyczaj) warto wykonywać selekcje jak najszybciej. Podobna zasada obowiązuje w przypadku przetwarzania strumieni danych gdyż inne operacje trwają zazwyczaj dłużej niż selekcje, które redukują rozmiary strumieni danych.

Zastosowanie przez procesor zapytań wprowadzonej metody dopasowania przeplotu (33) oraz przemienności sumowania (30,31) umożliwi procesorowi zapytań dobór bardziej odpowiedniego planu realizacji zapytania.



Rysunek 1. Logiczny plan zapytania

6 Podsumowanie

W ramach prac badawczych realizowane są zadania mające na celu stworzenie systemów monitorujących dla medycyny [6]. W większości przypadków zadanie zarządzania danymi powierza się bezpośrednio systemowi monitorowania. Takie rozwiązanie początkowo nie komplikuje znacząco konstrukcji systemu. Jednak z czasem sytuacja ulega zmianie. Okazuje się, że zaimplementowane algorytmy są silnie związane z wybraną implementacją i dlatego zastosowanie nowszych rozwiązań sprzętowych jest utrudnione. Prowadzone prace mają na celu przedstawienie systemu zarządzania danymi wyposażonego w formalny język zapytań, który rozwiąże ten problem. Rozwiązanie oparto na rozszerzonej koncepcji strumieniowego przetwarzania danych. Zastosowanie proponowanego systemu upraszcza konstrukcję algorytmów przetwarzania sygnałów, zapewnia bezpieczeństwo danym oraz rozwiązuje problemy z współużytkowaniem informacji w systemie monitorującym.

Systemy zarządzania strumieniami danych to obecnie jedną z bardziej intensywnie rozwijanych dziedzin nauki związanej z problematyką systemów baz danych. Prowadzone prace mają na celu wykazanie efektywności zastosowania stworzonego modelu przetwarzania informacji w stosunku do istniejących rozwiązań [27]. Zadanie to jest o tyle utrudnione, że przyjęty model relacyjny jest modelem uniwersalnym. Od modelu strumieniowego oczekuje się, że zastosowanie go w zadaniu monitorowania pozwoli na prostsze i bardziej eleganckie wyrażenie zapytań oraz dzięki uproszczeniu konstrukcji uzyskany zostanie znaczący wzrost efektywności przetwarzania informacji. Osiągnięcie stopnia uniwersalności modelu relacyjnego będzie jednak bardzo utrudnione.

W artykule przedstawiono opracowane w ramach prac badawczych twierdzenia algebry ciągów danych (strumieni) wraz z wyprowadzonymi dowodami. Udowodniono bezpośredni związek zastosowanych operatorów z odkrytymi w połowie zeszłego wieku twierdzeniami Teorii Liczb. Twierdzenia te doczekały się stosunkowo niedawno kolejnych dowodów [17].

Literatura

1. C.J. Date. *Wprowadzenie do systemów baz danych*. Wydawnictwa Naukowo-Techniczne, 2000.
2. Michał Widera, Adam Domański, and Piotr Kasprzyk. Analiza zastosowania baz danych w zadaniu przetwarzania sygnałów biomedycznych. In *Bazy danych Modele, technologie, narzędzia - Analiza danych i wybrane zastosowania*, pages 371–378. WKiŁ, 2005.
3. Alberto Lerner and Dennis Shasha. Aquery: Query language for ordered data, optimization techniques, and experiments. In *Proceedings of 29th International Conference on Very Large Data Bases*, pages 345–356, 2003.
4. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM Press, 2002.
5. Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, 2001.
6. Michał Widera, Adam Matonia, Janusz Wróbel, Janusz Jeżewski, Krzysztof Horoba, and Tomasz Kupka. Neonatal surveillance system based on data stream technology. In *Journal of Medical Informatics & Technologies JMIT*, volume 9, pages 93–106. Dept. of Computer Systems University of Silesia, October 2005.
7. Michał Widera, Janusz Jeżewski, Ryszard Winiarczyk, Janusz Wróbel, Krzysztof Horoba, and Adam Gacek. Data stream processing in fetal monitoring system: I. algebra and query language. In *Journal of Medical Informatics and Technologies*, volume 5, pages 83–90. Dept. of Computer Systems University of Silesia, 2003.
8. Michał Widera, Janusz Wróbel, Adam Widera, and Adam Gacek. System zarządzania danymi dla potrzeb medycznych systemów monitorujących. In *Współczesne problemy Systemów Czasu Rzeczywistego*, pages 448–458. WNT, 2004.

9. Michał Widera and Stanisław Kozielski. Strumieniowe systemy zarządzania danymi przegląd rozwiązań. In *Bazy Danych Modele, metody formalne, bezpieczeństwo*, pages 257–266. WKiŁ, 2005.
10. Michał Widera and Stanisław Kozielski. Metody przetwarzania w strumieniowych systemach zarządzania danymi. In *Bazy Danych Modele, metody formalne, bezpieczeństwo*, pages 267–276. WKiŁ, 2005.
11. Hari Balakrishnan, Magdalena Balazinska, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Eddie Galvez, Jon Salz, Michael Stonebraker, Nesime Tatbul, Richard Tibbetts, and Stan Zdonik. Retrospective on aurora. *The VLDB Journal*, 13(4):370–383, December 2004.
12. Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S Maskey, Alexander Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stan Zdonik. The design of the borealis stream processing engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Asilomar, CA, January 2005.
13. Sven Schmidt, Henrike Berthold, and Wolfgang Lehner. Qstream: Deterministic querying of data streams. In *Proceedings of the 2004 VLDB Conf.*, pages 1365–1368, 2004.
14. Michał Widera, Janusz Wróbel, Aleksander Owczarek, Adam Matonia, and Michał Jezewski. Data management system for computer aided biophysical monitoring. In *Proc. 27th Annual Int. Conf. on the IEE-EMBS, 2005 Innovation from Biomolecules to Biosystems*, pages 1.6.2–5. IEEE EMBC '05, IEEE Press, September 2005.
15. S. Beatty. Problem 3173. *Amer. Math. Monthly*, 33:159, 1926.
16. A. S. Fraenkel. The bracket function and complementary sets of integers. *Canad. J. Math*, 21:6–27, 1969.
17. Kevin O'Bryant. Fraenkel's partition and brown's decomposition. *The Electronic Journal of Combinatorial Number Theory INTEGERS*, 3:A11–17, 2003.
18. S. Samadi, M.O. Ahmad, and M.N.S. Swamy. Characterization of nonuniform perfect-reconstruction filterbanks using unit-step signal. *IEEE Transactions on Signal Processing*, 52(9):2490–2499, 2004.
19. Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, 2003.
20. Daniel J. Abadi, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.
21. Arvind Arasu and Jennifer Widom. A denotational semantics for continuous queries over streams and relations. *SIGMOD Rec.*, 33(3):6–11, 2004.
22. Janusz Wróbel, Michał Widera, Krzysztof Horoba, Janusz Jezewski, and Ryszard Winiarczyk. Declarative algebra and continuous query language for biomedical stream processing in fetal monitoring system. In *proc. 26th International Conference of IEEE Engineering in Medicine and Biology Society*, pages 3175–3178. IEEE IFMBE, 2004.
23. Jeffrey D. Ullman and Jennifer Widom. *Podstawowy wykład z systemów baz danych*. Wydawnictwa Naukowo-Techniczne, 2001.
24. Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Implementacja systemów baz danych*. Wydawnictwa Naukowo-Techniczne, 2003.
25. M. Tamer and Ozsu. Processing sliding window multi-joins in continuous queries over data streams. In *VLDB*, pages 500–511, 2003.
26. Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Kompilatory. Reguły, metody i narzędzia*. Wydawnictwa Naukowo-Techniczne, 2002.
27. Arvind Arasu, Mitch Cherniack, Eduardo F. Galvez, David Maier, Anurag Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. Linear road: A stream data management benchmark. In *Proceedings of the 2004 VLDB Conf.*, pages 480–491, 2004.