

RetractorDB™

Baza danych serii czasowych dla potrzeb przetwarzania sygnałów

Michał Widera

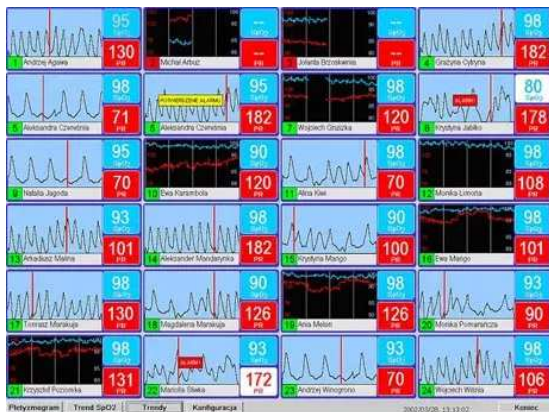
12 marca 2025

RetractorDB

Overview

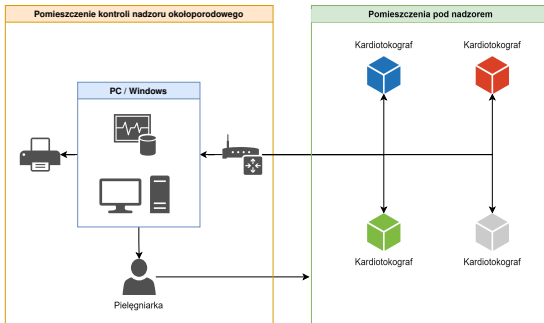
1. Motywacja
2. Algebra
3. Język zapytań
4. Podstawy konstrukcji systemu
5. Królicza norka
6. Dodatkowe materiały

Po co to? Dlaczego? - jaka była motywacja?



Rysunek: Monako Saturno - ekran systemu, itam-system.com

Po co to? Dlaczego? - jaka była motywacja?



Rysunek: Monaco Saturno, schemat systemu

Teoria Liczb - Samuel Beatty (1926)

Twierdzenie (Beatty)

Jesli 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) (1).

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

lub sekwencją Beatty (sufit) (2):

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

Twierdzenie (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 $\lceil q\alpha' \rceil + \lceil q\beta' \rceil = 1$.*

Definicja

Struktura algebraiczna składająca się z jednego lub kilku zbiorów oraz działań określonych na tych zbiorach

Operatory algebry

- Rzutowanie
- Suma i różnica
- Przeplot i rozplątanie
- Agregacja i Serializacja
- Przesunięcie

Model danych

$S ::= (s_n, \Delta)$ gdzie $\Delta \in \mathbb{Z} > 0$ stanowi wymiar czasu, s_n jest zbiorem obserwacji danego zjawiska indeksowany n .

M.Widera: Deterministic method of data sequence processing,

Annales UMCS Sectio AI Informatica, Vol. IV, 2006, str. 314-331

M.Widera: Deterministyczna metoda przetwarzania ciagow danych,

XXI Autumn Meeting of Polish Information Processing Society, 2005, Conf.Pr

Input: A=[1,2,3,4,...],3; B=[a,b,c,d,...],1

```
deltaC = min(deltaA,deltaB)
for i in range(0,10):
    if deltaC == deltaA:
        print str(A[i])+B[int(i*deltaA/deltaB)],
    else:
        print str(A[int(i*deltaB/deltaA))]+B[i],
```

Output: 1a 1b 1c 2d 2e 2f 3g 3h 3i 4j, Delta = 1

retractordb.com/assets/sum.html

Input: C=[1a 1b 1c 2d 2e 2f 3g 3h 3i 4j ...],1
Arg: DeltaA = 3,deltaB = 1

```
for i in range(0,10):  
    if deltaA > deltaB :  
        print C[int(ceil(i*deltaA/deltaB))][0],  
    else:  
        print C[i][0],
```

Output: 1 2 3 4 5 6 7 8 9 10, Delta = 3

Suma i różnica - zapis formalny

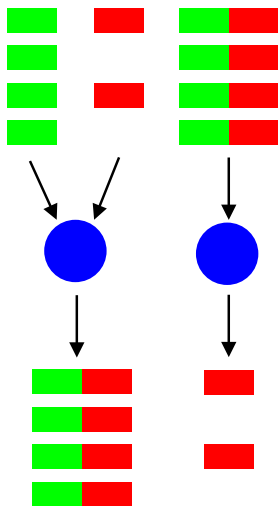
Suma:

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

Różnica:

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

Suma i różnica reprezentacja graficzna



Rysunek: Suma i różnica

Przeplot

Input: $A=[1,2,3,4,\dots], 2$; $B=[a,b,c,d,\dots], 1$

```
for i in range(0,10):  
    if floor(i*delta)==floor((i+1)*delta):  
        print B[i-int(floor((i+1)*delta))],  
    else:  
        print A[int(floor(i*delta))],
```

$\text{deltaC} = (\text{deltaA} * \text{deltaB}) / (\text{deltaA} + \text{deltaB})$

Output: a b 1 c d 2 e f 3 g , $\text{delta} = 2/3$

Rozplątanie

```
Input: C = [ a b 1 c d 2 e f 3 g ...], deltaC = 2/3
Arg: deltaB = 1
A_=[]
deltaA_ = deltaB*deltaC/abs(deltaB-deltaC)
for i in range(0,10) :
    A_.append( C[i+int(ceil((i+1)*deltaA/deltaB))] )

Output: 1 2 3 4 5 , delta = 2
```

Rozplątanie - Residue

Input: $C = [a\ b\ 1\ c\ d\ 2\ e\ f\ 3\ g\ \dots]$, $\text{delta}C = 2/3$

Arg: $\text{delta}A = 2$

$B_=[]$

$\text{delta}B_ = \text{delta}A * \text{delta}C / \text{abs}(\text{delta}A - \text{delta}C)$

for i in range(0,10) :

$B_.append(C[i + \text{int}(i * \text{delta}B_ / \text{delta}A)])$

Output: $a\ b\ c\ d\ e\ f\ g\ h\ i\ j$, $\text{delta} = 1$

Przeplot i rozplątanie - zapis formalny

Przeplot:

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

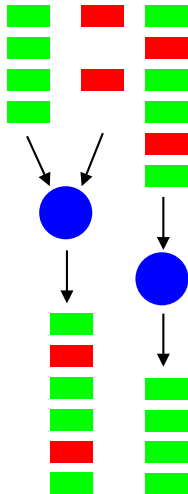
Rozplątanie:

$$a_n = c_{n+\left\lceil \frac{(n+1)\Delta_a}{\Delta_b} \right\rceil}, \Delta_a = \frac{\Delta_c \Delta_b}{|\Delta_c - \Delta_b|} \quad (6)$$

Rozplątanie - residue:

$$b_n = c_{n+\left\lfloor \frac{n\Delta_b}{\Delta_a} \right\rfloor}, \Delta_b = \frac{\Delta_c \Delta_a}{|\Delta_c - \Delta_a|} \quad (7)$$

Przeplot i rozplątanie reprezentacja graficzna



Rysunek: Przeplot i rozplątanie

Formalnie...

```
Zapytanie ::=  
SELECT ListaSelekcji  
STREAM NazwaStrumienia  
FROM WyrażenieStrumieniowe
```

Przykład poprawnego zapytania zgodnego z gramatyką:

```
SELECT a,b  
STREAM nazwaTworzonegoStrumienia  
FROM strumienWejscowy1 + strumienWejscowy2
```

Operacje wewnątrz wyrażenia strumieniowego

$A \# B$	Przeplot strumieni A i B
$A \& 1/2$ lub $A \% 0.5$	Rozplątanie i dopełnienie rozplątania strumienia A względem $\Delta = \frac{1}{2}$
$A + B$	Suma strumieni danych A i B
$A - 1/2$	Różnica strumienia danych względem $\Delta = \frac{1}{2}$
$A > 30$	Przesunięcie strumienia A w domenie czasu o 30 próbek
$A @ (2, 3)$	Agregacja i serializacja strumienia A.

Przykłady wyrażeń zgodnych z gramatyką

- | | |
|---------------|--|
| $(A+B)>20$ | Połączenie sumaryczne strumieni A oraz B oraz przesunięcie wyniku w czasie o 20 elementów. |
| $A\#(B>20)$ | Przesunięcie w czasie elementów strumienia B o dwadzieścia elementów. Wynik przepleciony ze strumieniem A. |
| $(B+A)@(2,2)$ | Agregacja połączonych strumieni A oraz B. |
| $A.MAX$ | Wyznaczenie maksymalnego elementu w schemacie strumienia A |

Gramatyka wyrażenia strumieniowego

WyrażenieStrumieniowe ::=

StrumieniowySymbolTerminalny

{ '+' StrumieniowySymbolTerminalny

| '-' Delta

| '>' LiczbaCalkowita };

StrumieniowySymbolTerminalny ::=

StrumieniowySymbolProdukcji

{ '#' StrumieniowySymbolProdukcji

| '&' Delta

| '%' Delta

| '@' (LiczbaCalkowita,LiczbaCalkowita)

| '.' { MIN | MAX | AVG | SUM } };

StrumieniowySymbolProdukcji ::=

IdentyfikatorStrumienia

| (WyrażenieStrumieniowe)

| '{' Zapytanie '}';

Programy

Programy tworzące procesy systemu zarządzania danymi:

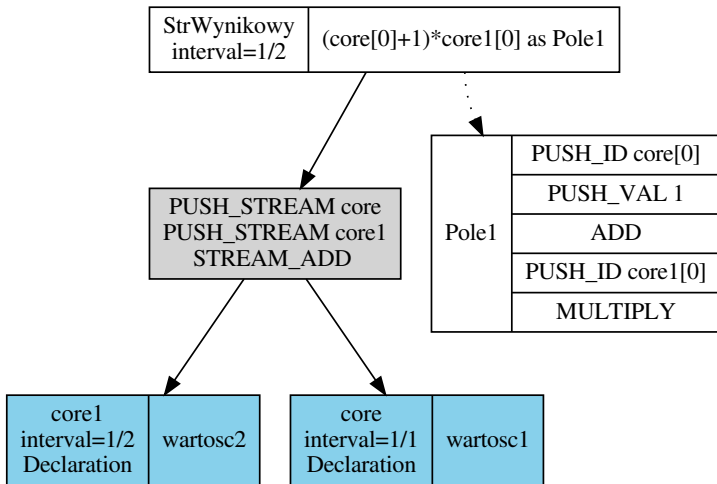
- **xretractor** - Kompilator i Serwer danych
- **xqry** - Klient
- **xtrdb** - narzędzie do testowania

Przykład zapytania:

```
DECLARE a INTEGER,b INTEGER STREAM core0,1
DECLARE c INTEGER,d INTEGER STREAM core1,0.5

SELECT (core0[0]+1)*core1[0] as Pole1
STREAM StrWynikowy FROM core0 + core1
```

Plan realizacji zapytania



Rysunek: Plan realizacji zapytania

Szeregowanie zadań - sloty czasowe

Dobór kolejnych slotów czasowych w oparciu o zbiór liczników i częstotliwości strumieni danych.

Funkcja WybórKolejnegoSlotu()

wynik = dowolnieDużaLiczba

DLA KAŻDEGO zbiórDelta.element

 JESLI wynik > licznik[element] * element T0:

 wynik = licznik[element] * element

DLA KAŻDEGO zbiórDelta.element

 JEŚLI wynik == licznik[element] * element T0:

 licznik[element] := licznik[element] + 1

ZWRÓĆ wynik

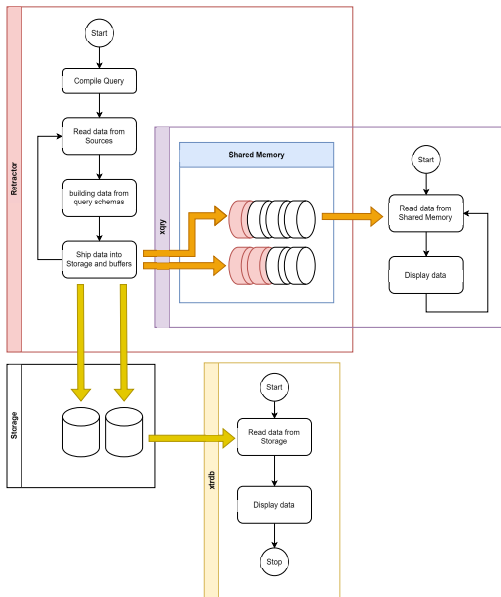
Jeśli w zbiorze znajdowały się np wartości $\{\frac{1}{2}, \frac{3}{4}, 1, 3, 9\}$ to po wykonaniu tej operacji w zbiorze pozostaną wartości $\{\frac{1}{2}, \frac{3}{4}\}$.

Szeregowanie zadań - pętla procesora zapytań

```
Procedura ProcesorZapytań()  
poprzedniInterwał := 0  
WYKONUJ  
    interwał = WybórKolejnegoSlotu()  
    period = interwał - poprzedniInterwał  
    poprzedniInterwał = interwał  
    DLA KAZDEGO Zadania  
        JEŚLI N*Zadanie.Delta = interwał TO:  
            WYKONAJ Zadanie  
        ODCZEKAJ period
```

Maksymalne opóźnienie podlega kontroli na etapie kompilacji i wykonania. Średnie opóźnienie jest stałe i równe maksymalnemu. Wymogi dla systemu twardego czasu rzeczywistego.

Konstrukcja systemu



Moja królicza norka...

Liczby zespolone Gaussa:

$$z = a + bi \quad (8)$$

pl.wikipedia.org/wiki/Liczby_calkowite_Eisensteina

$$z = a + b\omega$$
$$\omega = \frac{-1 + i\sqrt{3}}{2} = e^{\frac{2}{3}\pi i} \quad (9)$$

Liczby wymierne Eisensteina:

$$z = \frac{a}{b} + \frac{c}{d}\omega \quad (10)$$

Implementacja: github.com/michalwidera/equations

Artykuł (trochę przeterminowany...)



Rysunek: Programista Magazyn nr 92, 5/2020, str. 14-20