

Przetwarzanie strumieni danych



Plan wykładu

- Wstęp
- Zastosowania dla systemów zarządzania strumieniami danych - DSMS
- Charakterystyka DSMS
- Model strumieniowych danych
- Przetwarzanie zapytań w DSMS
- Algorytmy przetwarzania zapytań
- Otwarte problemy badawcze



Klasyczne systemy bazy danych

- Pasywne repozytoria dużych zbiorów trwałych danych składowane na dyskach, które są wypełniane i przetwarzane przez personalnych użytkowników (human active model).
- Przechowują bieżący stan danych. Dostęp do historii zmian danych może być utrudniony lub niemożliwy.



Klasyczne systemy bazy danych

- Elementy aktywne: triggerzy i alerty są drugorzędne i pomocnicze. Brak skalowalności dla dużej liczby triggerów.
- Dostęp do danych jest synchronizowany i wykonywany na całym zbiorze danych dzięki czemu odpowiedzi na zapytania są dokładne.
- Brak wymagań, co do przetwarzania danych w czasie rzeczywistym.



Nowe dziedziny zastosowań

- Zarządzanie ruchem i wydajnością sieci telekomunikacyjnych i komputerowych, serwerów WWW
- Zarządzanie procesami przemysłowymi – przetwarzanie danych z sieci czujników
- Aplikacje analizy finansowej monitorujące strumień danych giełdowych, sieci bankomatów
- Aplikacje monitorujące duże zbiory obiektów (śledzenie pojazdów)
- Aplikacje militarne – monitorowanie organizmów żołnierzy (ciśnienie krwi, tętno, pozycja)



Przetwarzanie strumieni danych

- Aktywne repozytoria dużych zbiorów danych, które są wypełniane automatycznie przez czujniki. Aplikacje monitorujące przetwarzają ciągłe strumienie danych wejściowych i alarmują użytkowników w przypadku występowania nietypowych sekwencji danych (human passive model).
- Ważna jest historia pojawiania się danych w źródłach danych, tj. kolejność i czas pojawiania się elementów strumieni.



Przetwarzanie strumieni danych

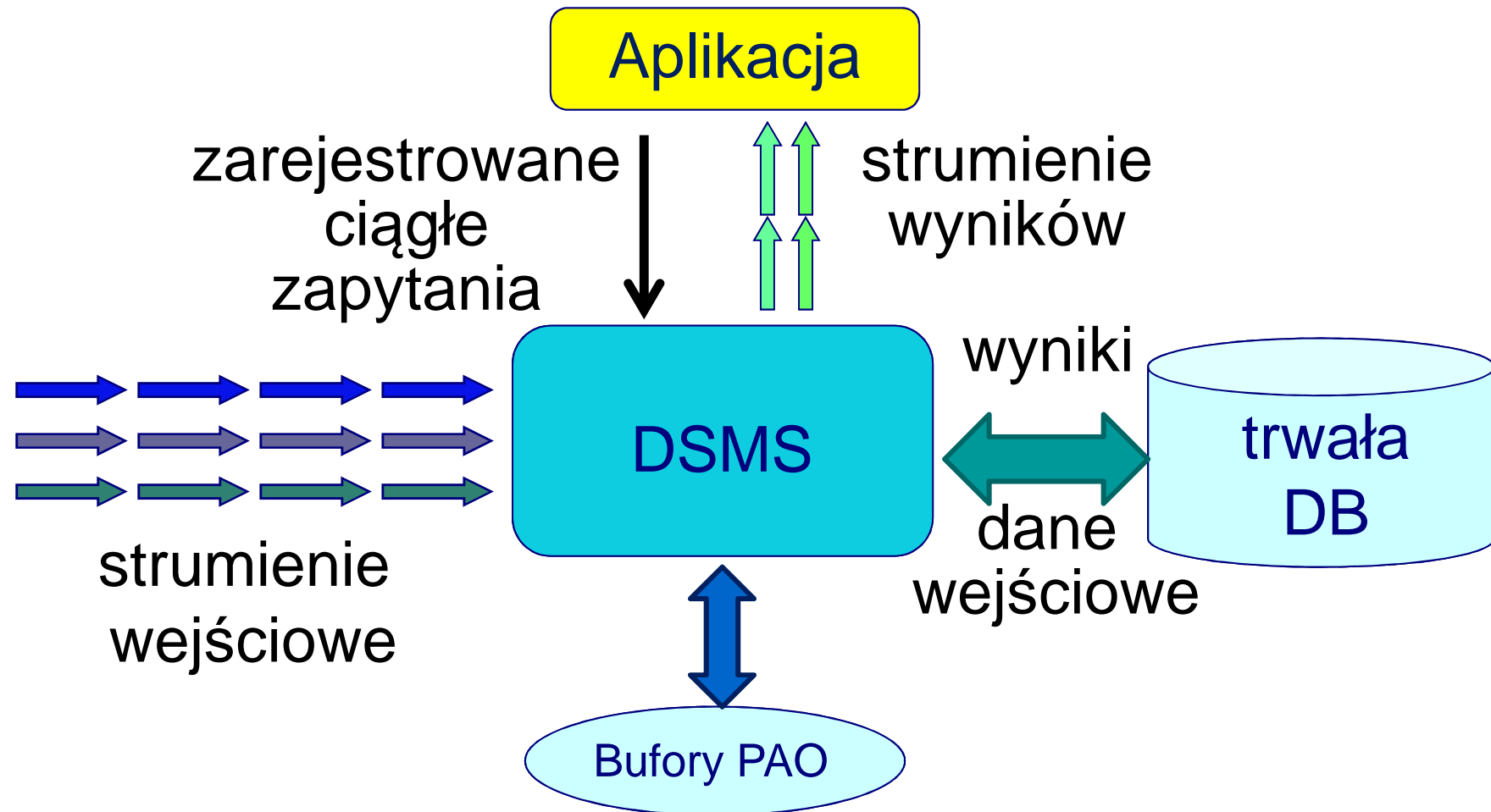
- Systemowa aktywność, architektura gwarantująca skalowalność triggerów.
- W intensywnych strumieniach, dane wejściowe mogą być gubione. W takim wypadku, zapytania będą wykonywane dla niekompletnej informacji wejściowej i będą generować jedynie przybliżone wyniki.
- Istotne wymagania, co do przetwarzania danych w czasie rzeczywistym.



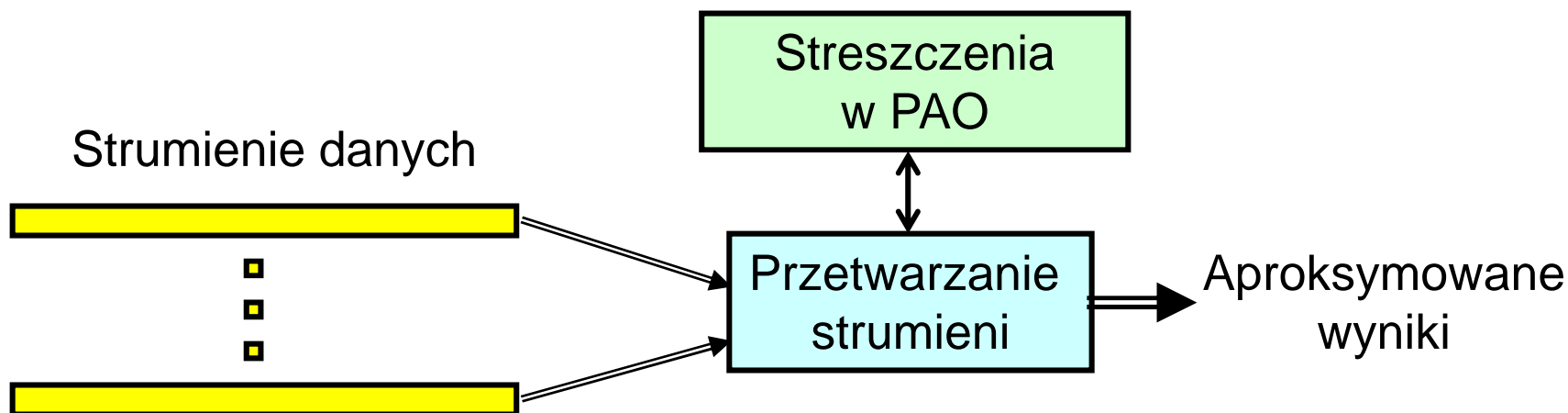
DBMS, a DSMS

- trwałe, ograniczone relacje
- zapytania punktowe
- miejscem przechowywania danych jest nieograniczona pamięć dyskowa
- dostęp swobodny
- plan zapytań definiowany na znanym rozkładzie danych
- stosowalność algorytmów wielo-przebiegowych
- ulotne, nieograniczone strumienie danych
- zapytania rejestrowane i ciągłe
- dane lub ich reprezentacje są składowane w ograniczonej pamięci operacyjnej
- dostęp sekwencyjny
- nieprzewidywalny rozkład danych
- konieczność stosowania algorytmów jednoprzebiegowych

Data Streams Management System



Model obliczeniowy strumieni danych



- Strumień danych jest potencjalnie nieograniczoną, intensywną, zmienną i nieregularną sekwencją danych
- Przetwarzanie strumieni danych wiąże się z:
 - Co najwyżej jednokrotnym dostępem do każdego z elementów
 - Ograniczonym obszarem do składowania streszczeń elementów strumienia w pamięci operacyjnej
 - Przetwarzaniem w czasie rzeczywistym – czas przetwarzania pojedynczego rekordu musi być krótszy od częstotliwości strumienia
 - Nieznajomością rozkładów danych



Algorytmy przetwarzania strumieni danych

- Złożoność obliczeniowa algorytmów musi być mniejsza od liniowej – dla umożliwienia przetwarzania bardzo intensywnych strumieni
- W ogólności, algorytmy wyznaczają jedynie przybliżone odpowiedzi
 - Trudność wyznaczenia dokładnych wyników jest konsekwencją ograniczonego rozmiaru PAO (nie można przetworzyć wszystkich danych) i intensywności strumienia wejściowego (nie można wczytać wszystkich danych)
- Wyniki aproksymowane - granice deterministyczne
 - Algorytmy wyznaczają przybliżone wyniki, ale wielkość błędów powinna być znana i ograniczona
- Wyniki aproksymowane - granice probabilistyczne
 - Algorytmy wyznaczają przybliżone wyniki z określonym prawdopodobieństwem
- Jednoprzebiegowe algorytmy przetwarzania strumieni stosowalne dla intensywnych terabajtowych strumieni danych



Przykładowe aplikacje

- Bezpieczeństwo i wydajność sieci komputerowych (np. Cisco NetFlow, iPolicy, NetForensics/Cisco, Niksun)
 - Strumień pakietów sieciowych, informacje o sesjach użytkowników
 - Zapytania: filtrowanie URL, wykrywanie włamań, ataków typu DOS, wirusów
- Aplikacje finansowe (High Frequency Trading - np. Tradebot, Tradeworx)
 - Strumień danych giełdowych, cen surowców, walut
 - Zapytania: wyszukiwanie arbitażu, szukanie określonych wzorców zmian cen akcji w celu wykonania transakcji, szukanie podejrzanych transakcji, itp.



Przykładowe aplikacje – zarządzanie sieciami

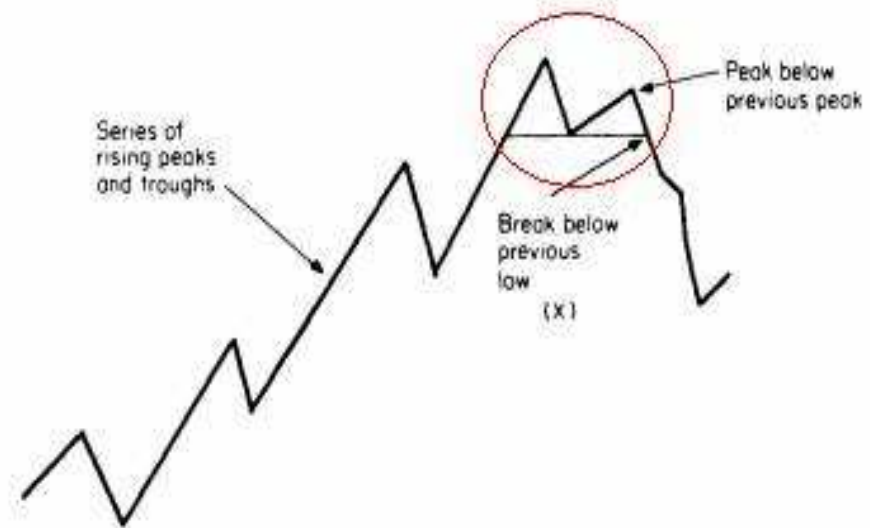
- Dane o sesjach (zebrane za pomocą Cisco NetFlow)

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

- AT&T zbiera 100 GB danych każdego dnia

Przykładowe aplikacje – monitorowanie giełdy

- ❑ Strumień cen i sprzedaży na giełdach w czasie
 - ❑ Analiza techniczna dla inwestorów giełdowych
 - ❑ Wsparcie dla decyzji o kupnie/sprzedaży
-
- *Sygnalizacja wzrostu notowań giełdowych o co najmniej 5%*
 - *Sygnalizacja monotonicznego wzrostu cen giełdowych dłuższego niż 1 godzina*
 - *Pik mniejszy od poprzedniego*
 - *Przełamanie poprzedniego dołka*



Model danych strumieniowych

■ Strumień

- Seria krotek uporządkowanych za pomocą znaczników czasowych
- Para (s, τ) , s jest krotką, a τ znacznikiem czasowym

■ Relacja

- zmienny w czasie wielozbiór krotek

Strumień **S**:

<nazwa, wartość, τ >

<XXXX, 12, 7:05>
<YYYY, 13, 7:06>
<XXXX, 17, 7:10>
<YYYY, 11, 7:12>

Relacja **R**

<nazwa, wartość>

R(7:08)	<XXXX, 12> <YYYY, 13>
R(7:15)	<XXXX, 17> <YYYY, 11>



Znaczniki czasowe

- **Explicite** – są częścią struktury elementów strumienia danych
- **Implicite** – są dodawane przez DSMS w momencie odbioru poszczególnych elementów strumienia
- **Czas ważności** – znaczniki reprezentują czas wystąpienia/ważności danej w świecie rzeczywistym
- **Czas transakcyjny** – znacznik rejestruje czas pojawienia się informacji w systemie



Realizowane projekty DSMS

■ Prototypy uniwersalnych DSMS

- **STREAM** (Stanford)
- **Aurora** (Brown, Brandeis, MIT)
- **TelegraphCQ** (Berkeley)

■ Systemy wyspecjalizowane

- **Tradebot** (www.tradebot.com) – finansowy serwis internetowy
- **Tribeca** (Bellcore) – monitorowanie sieci
- **Cougar** (Cornell) – sieci czujników
- **Hancock**(AT&T) – sieci telekomunikacyjne

■ Uniwersalne systemy komercyjne

- **StreamBase** (StreamBase Systems)
- **Esper** (EsperTech)



Język zapytań w DSMS

- deklaratywny (CQL) - oparty na rachunku relacji (STREAM),
- obiektowy - systemy monitorowania sieci czujników (Cougar),
- proceduralny - plan realizacji zapytania tworzony jest bezpośrednio przez użytkownika (Aurora/Borealis).



Języki bazujące na SQL

Rozszerzanie funkcjonalności języka SQL o przetwarzanie strumieni danych.

- Potrzeba nowej implementacji operatorów relacyjnych
- Operatory semantycznie blokujące zapytania ciągłe
- Operatory implementacyjnie blokujące
- Operatory transformujące strumienie w relacje i historię przetwarzania relacji w strumienie
- Ruchome okna zapytań



Operacje blokujące

Dana operacja jest blokująca, jeżeli uzyskanie pierwszej krotki wyniku wymaga odczytania całego zbioru wejściowego, np. : *avg, sort, min, max, median*

```
select AVG(wartość) from S
```

Ograniczenia w stosowaniu operacji blokujących na strumieniach danych – dla nieograniczonego strumienia wejściowego – nigdy nie zwrócą wyniku.



Operacje blokujące

Dane: operacja \circ i zbiory krotek A , B . Operacja \circ jest nieblokująca jeżeli:

$$A \subseteq B \Rightarrow \circ(A) \subseteq \circ(B)$$

Operacje nieblokujące: *selekcja, połączenie, eliminacja duplikatów.*

`select * from S where S.wartość>10`

Blokująca implementacja operacji

Operacja połączenia jest operacją nieblokującą.


Algorytm nested-loop jest blokującą implementacją operacji join.



Operacje blokujące

■ Potencjalne rozwiązania:

- Ruchome okna danych - wyniki w strumieniu wyjściowym są generowane dla ograniczonych podzbiorów danych (np. 5 minut, 100 danych)
- rozszerzenie strumienia danych o dodatkowe informacje o tym, co pojawi się w pozostałej części strumienia – znaki interpunkcyjne (ang. punctuation), są wyrażeniami logicznymi, które będą spełnione dla wszystkich następujących po nich elementach strumienia
- własności strumienia danych, na przykład wiedza o monotoniczności danych



Ruchome okna (sliding windows)

- Idea ruchomych okien polega na przetwarzaniu tylko ograniczonego ciągłego podzbioru danych ze strumienia, które mieszczą się w zdefiniowanym oknie
- Zalety Ruchomego okna:
 - metoda optymalizacji – przetwarzanie ograniczonego zbioru danych,
 - dobrze zdefiniowane,
 - łatwe do zrozumienia,
 - determinizm działania,
 - analiza tylko najnowszych danych.



Ruchome okna w języku CQL

Język bazujący na SQL rozszerzony, między innymi, o dwa dodatkowe operatory:

[Rows n] – okno o stałym rozmiarze obejmujące ostatnich **n** elementów strumienia

[Range m jednostek czasu] – okno o stałym rozmiarze obejmujące elementy strumienia, których znaczniki czasowe mieszczą się w przedziale czasowym $[\text{now} - m \text{ jednostek czasu}, \text{now}]$

Szczególne przypadki okien:

- $[-\infty, \infty]$ – okno obejmuje cały strumień;
- $[-\infty, \text{now}]$ – ciągle powiększające się okno od początku strumienia do *teraz*;
- $[\text{now}, \text{now}]$ – pojedynczy element strumienia, okno punktowe



Ruchome okna w języku CQL

```
select avg(S.wartość)
from S [Rows 5]
```

Transformacja strumienia w relację zawierającą 5 krotek o największych znacznikach czasu

```
select *
from R [Rows 1000], S [Range 5 min]
where R.A = S.A and R.B > 10
```

Transformacja strumienia w relację zawierającą krotki, których znaczniki czasowe mieszczą się w przedziale [now – 5 min, now]

Ruchome okna

```
select avg(S.wartość) from S [Rows 5]
```

<AAAA, 12, 7:05>
<BBBB, 21, 7:06>
<CCCC, 17, 7:10>
<AAAA, 11, 7:12>
<DDDD, 7, 7:15>
<EEEE, 13, 7:16>
<FFFF, 19, 7:17>





Język CQL – prototyp Stanford

Rozszerzenie języka SQL o:

- Nowy typ kolekcji danych - **Streams**
- Semantyka **ciągłych** zapytań
- Definiowanie **okien** na strumieniach danych – operator stream-to-relation
- **Próbkowanie** strumieni danych
- Trzy operatory relation-to-stream

Istream, Dstream Rstream

Dane są strumienie wejściowe:

- **Zamówienia**(id, klient, wartość),
- **Wystawianie**(id_zam, pracownik)

```
select sum(z.wartość)
from zamówienia z, wystawianie w [Range 1 Day]
where z.id = w.id_zam and w.pracownik = 'Tarzan'
      and z.klient = 'Kowalski i Synowie'
```

Język CQL – prototyp Stanford

■ Próbkowanie strumieni danych

```
select w.pracownik, max(z.wartość)
from zamówienia z, wystawianie w
  [partition by pracownik rows 5] 10% sample
where z.id = w.id_zam
group by w.pracownik
```

tworzenie
równoległych okien

Wynik powyższego zapytania jest relacją modyfikowaną odpowiedni do zmian w strumieniu wejściowym.

■ Transformacja relacji w strumień

```
select Istream(w.pracownik, max(z.wartość))
from zamówienia z, wystawianie w
  [partition by pracownik rows 5] 10% sample
where z.id = w.id_zam
group by w.pracownik
```

Wynik powyższego zapytania jest strumieniem elementów generowanych w momencie zmiany wartości max lub nowego pracownika.



Język CQL – prototyp Stanford

■ Tworzenie strumienia zmian stanu relacji

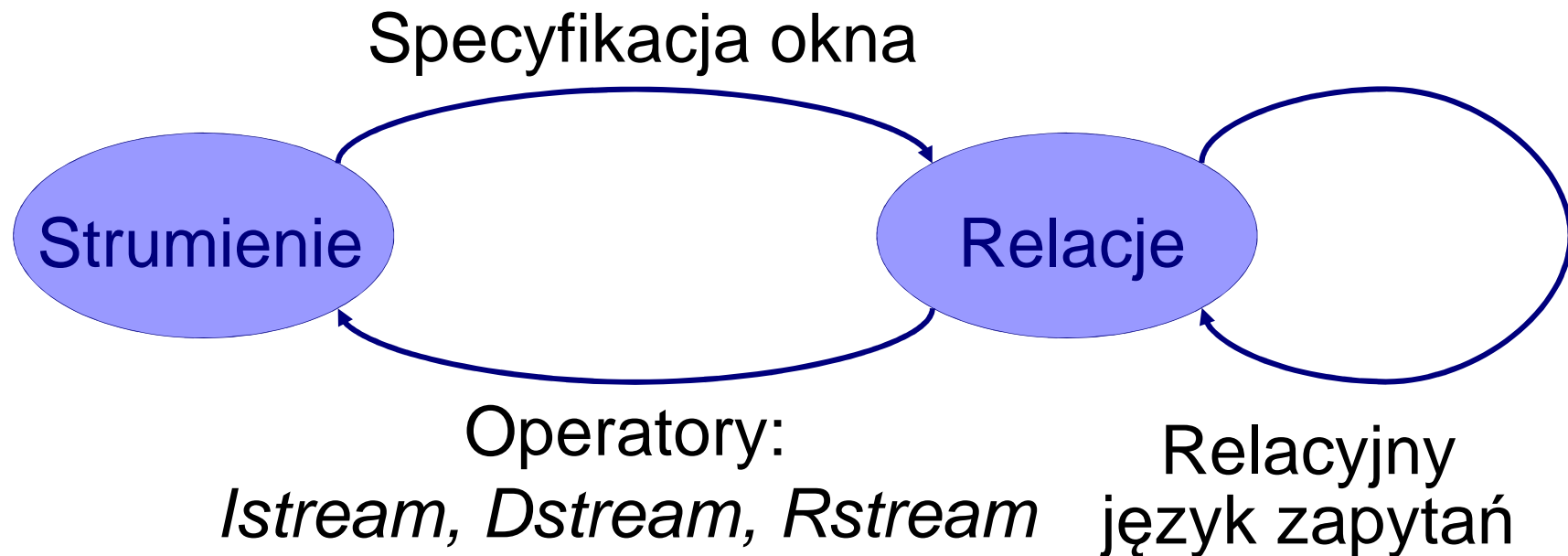
Dana jest relacja:

`AktualneCenyAkcji (id_akcji, cena),`
modyfikowana na bieżąco przez zmiany cen akcji.

```
select id_akcji, avg(cena)
from Istream(AktualneCenyAkcji) [Range 1 Day]
group by id_akcji
```

Operator *Istream* generuje strumień zmian relacji `AktualneCenyAkcji`.
Dla tego strumienia wyliczana jest średnia cena każdej z akcji dla ostatniego dnia .

Język CQL



ISTREAM: strumień wstawianych i modyfikowanych krotek

DSTREAM: strumień usuwanych krotek

RSTREAM: strumień wszystkich krotek



Operatory konwersji

- Strumień -> Relacja

W punkcie czasowym T relacja R jest zbiorem wszystkich krotek okna W zdefiniowanego na strumieniu S , które pojawiły się w strumieniu do czasu T .

Dla nieskończonego okna relacja zawiera wszystkie elementy strumienia.

- Relacja -> Strumień

- ☐ Istream(R) – jest strumieniem, który zawiera wszystkie pary (r, T) , takie że $r \in R$ w chwili T , i $r \notin R$ w chwili $T-1$
- ☐ Dstream(R) – jest strumieniem, który zawiera wszystkie pary (r, T) , takie że $r \in R$ w chwili $T-1$, i $r \notin R$ w chwili T
- ☐ Rstream(R) - jest strumieniem, który zawiera wszystkie pary (r, T) , takie że $r \in R$ w chwili T



Wydajność przetwarzania strumieni danych

- Zastosowanie ruchomych okien nie jest mechanizmem gwarantującym skalowalność przetwarzania strumieni danych.
- Duża część zastosowań przetwarzania strumieni danych, wiąże się z zastosowaniem sprzętu komputerowego o mocno ograniczonych zasobach: komponenty systemu zarządzania sieciami komputerowymi (np. routery), sterowanie ruchem ulicznym (komputery jednoukładowe), itd.
- Przykładowy problem:

Router ma zliczać, pakiety pochodzące z określonej domeny w oknie o rozmiarze 10^{10} . Zasoby routera nie pozwalają na składowanie wszystkich elementów okna o rozmiarze 10^{10} .



Wydajność przetwarzania strumieni

■ wydajność czasowa

- ☐ operacja uaktualnienia struktur pomocniczych (*update(krotka)*)
- ☐ operacja wyznaczająca wynik zapytania (*computeAnswer()*)

■ wydajność zasobowa – rozmiar PAO

- ☐ możliwość dynamicznego dopasowania do określonego rozmiaru pamięci



Metody przetwarzania danych

- Przetwarzanie grupowe
- Próbkowanie
- Streszczenia (synopses)
 - Wavelet
 - Szkicowanie
 - Histogramy
 - Mikro-grupy



Przetwarzanie grupowe

■ Przetwarzanie grupowe - własności

- Szybka operacja *update* – wstaw daną do bufora,
 - wolna *computeAnswer()* – przelicz wszystkie dane w buforze
-
- ☐ Dane przed przetworzeniem są buforowane
 - ☐ Dane nie są przetwarzane na bieżąco
 - ☐ Przetwarzanie danych w grupach (paczkach)
 - ☐ Wynik – dane z przed „chwili”
 - ☐ Nie rozwiązuje problemu dostępności zasobów PAO



Próbkowanie

■ Próbkowanie - własności

- stosunkowo szybka operacja *computeAnswer()*,
- wolna *update()* – wyznaczanie próbki
- Dane przychodzą szybciej niż może zostać wykonane uaktualnienie struktury
- Stały obszar na przechowywanie reprezentatywnego podzbioru danych wejściowych
- Pojawienie się nowej danej na wejściu wiąże się z określonym prawdopodobieństwem jej wstawienia w zamian za inną losowo wybraną składowaną daną
- Zapytanie wykonywane na próbce strumienia danych
- Odpowiedź przybliżona



Próbkowanie

Dany strumień danych o długości N elementów i obszar w PAO do składowania próbki o rozmiarze n .

- Wyznaczanie kolejnego elementu do wstawienia do próbki na podstawie prawdopodobieństwa n/N

Wyjściowy algorytm jest niestosowalny dla strumieni danych ze względu na brak ograniczenia ich rozmiaru.

Potrzebne modyfikacje:

- Niezależność od rozmiaru strumienia
- Możliwość analizy danych przed zakończeniem strumienia
- Możliwość pomijania w analizie elementów strumienia wejściowego
- Możliwość wykonywania zapytań w krótkim (okna) i długim horyzoncie czasowym



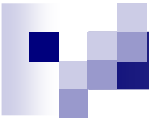
Próbkowanie zbiornikowe

ang. Reservoir Sampling

Wybór losowej próbki o rozmiarze n ze strumienia o nieznanej długości.

Rozwiązania bazujące na algorytmie zbiornikowym:

- Wstaw pierwsze n elementów strumienia do zbiornika.
- Przetwarzaj sekwencyjnie kolejne elementy strumienia.
 - Każdy z przetwarzanych elementów może być wstawiony do zbiornika z prawdopodobieństwem $n/(t+1)$, gdzie t jest liczbą porządkową przypisywaną kolejnym elementom strumienia. Wstawiane elementy zastępują losowo wybrane elementy przechowywane w zbiorniku. W dowolnym momencie prawdopodobieństwo znalezienia się wszystkich elementów w zbiorniku jest stałe i wynosi n/t .
- W dowolnym momencie ze zbiornika może być pobrana losowa próbka danych o rozmiarze n .

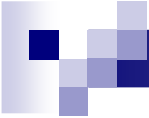


Próbkowanie – algorytm R

Algorytm R (Alan Waterman, Knuth) działa następująco:

- Kiedy przetwarzany jest $(t+1)$ element strumienia wejściowego, gdzie $t > n$, utrzymywanych w zbiorniku n elementów jest losową próbą pierwszych t elementów strumienia.
- Element $(t+1)$ może być wstawiony do zbiornika z prawdopodobieństwem $n/(t+1)$. Wstawiany element zastępuje losowo wybrany element ze zbiornika.

```
for j := 0 to n - 1 do ReadNextElement(C[j]);
t := n;                                // numer kolejnego elementu strumieni
while not eof do                        // przetwarzaj kolejne elementy strumienia
begin
    t := t + 1;
    m := Trunc (t x Random( )); // m zmienna losowa z zakresu  $0 \leq m \leq t-1$ 
    if m < n then                    // wstaw nowy element do zbiornika
        ReadNextRecord(C[m]);      // zastąp m-ty element ze zbiornika
    else
        SkipElement();
end;
```



Algorytm R - modyfikacja

Złożoność obliczeniowa algorytmu R jest równa $O(N)$.

Zwiększenie wydajności algorytmu przez analizowanie jedynie podzbioru elementów ze strumienia N (Jeffrey Vitter). Złożoność czasowa algorytmu: $O(n(1+\log(N/n)))$

```
for j := 0 to n - 1 do ReadNextElement(C[j]);
t := n;                                // numer kolejnego elementu strumieni
while not eof do                        // przetwarzaj kolejne elementy strumienia
begin
  Generate an independent random variate  $\vartheta(n, t)$ ;
  SkipElements( $\vartheta$ );                // pomiń następnych  $\vartheta$  elementów
  if not eof then
  begin
    m := Trunc (n x Random( )); // m zmienna losowa z zakresu  $0 \leq m \leq n-1$ 
    ReadNextRecord(C[m]);
    t := t +  $\vartheta$  + 1;
  end;
end;
```




Przykład zastosowania próbki danych

SELECT agg(el) FROM stream WHERE el BETWEEN 4 AND 7

Strumień: 9 3 5 2 7 1 8 5 8 4 9 1 (N=12)

Próbka: 9 5 1 8 (n=4)

agg = AVG – średnia elementów z bufora spełniających
warunek selekcji: 5 (5.25)

agg = SUM – suma elementów z bufora spełniających
warunek selekcji skorygowana o $\lfloor N/n \rfloor$: $5 * (12/4) = 20$ (21)

agg = COUNT – liczba elementów w buforze spełniających
warunek selekcji skorygowana o $\lfloor N/n \rfloor$:
 $1 * (12/4) = 3$ (4)



Streszczenia

Streszczenia (synopses) - Skrócowa reprezentacja danych

Wymagane własności:

- Szeroki zakres zastosowania
- Jednoprzebiegowe algorytmy utrzymania i wyznaczania wyników
- Duża wydajność czasowa (obciążenie procesora) i pamięciowa (zajętość pamięci)
- Poprawność dla różnych rozkładów danych
- Czułość na zmieniający się w czasie rozkład danych
- Ograniczona wielkość błędu przetwarzania zależna od rozmiaru dostępnej pamięci



Streszczenia

Złożoność pamięciowa streszczeń:

Udowodniono, że minimalny rozmiar pamięci niezbędny do składowania streszczenia, które pozwala uzyskiwać wyniki zapytań z błędem względnym równym ε , wynosi:

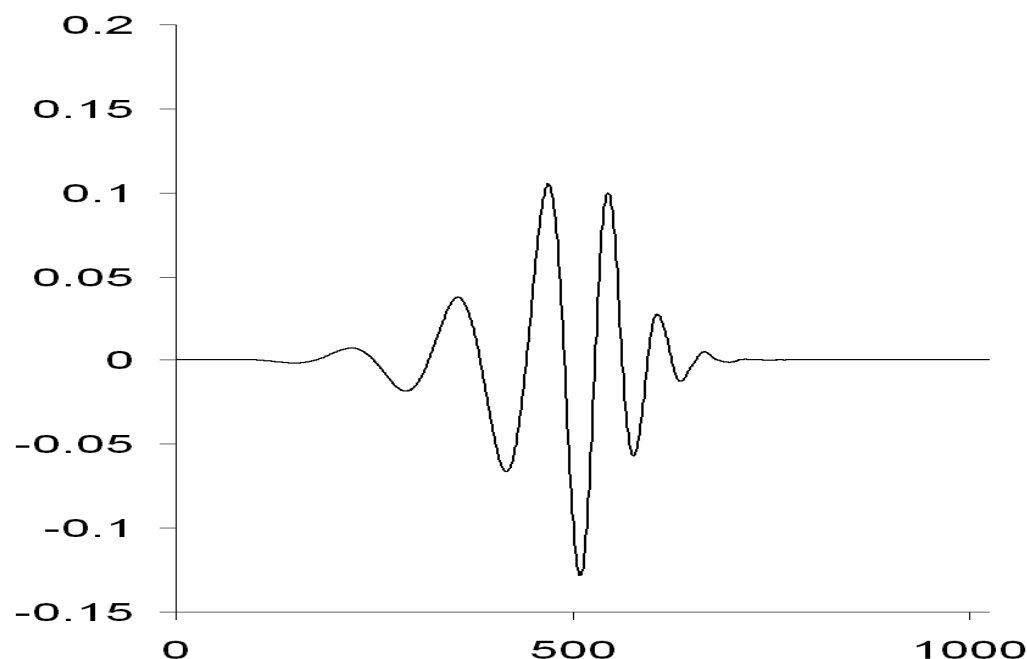
$$O(1/\varepsilon \log^2 N) \text{ bitów,}$$

gdzie, N jest rozmiarem okna lub liczbą elementów w strumieniu.

Dla wartości oczekiwanej względnego błędu równej $\varepsilon = 0.1$, okna o rozmiarze $N=10^{10}$, i strumieniu będącego ciągiem 0 i 1, minimalny rozmiar bufora pamięci wynosi: 300 bajtów.

Falki (ang. Wavelets)

- Pojęcie *falki* pochodzi z sejsmologii
- Falki opisują rozkład fal w impulsie
- W informatyce falki znalazły zastosowanie w metodach kompresji, grafice komputerowej i bazach danych





Falki Haara

- Funkcja macierzysta falki Haara $\psi(t)$ jest zdefiniowana następująco:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

- Funkcja skalująca $\phi(t)$ jest zdefiniowana następująco:

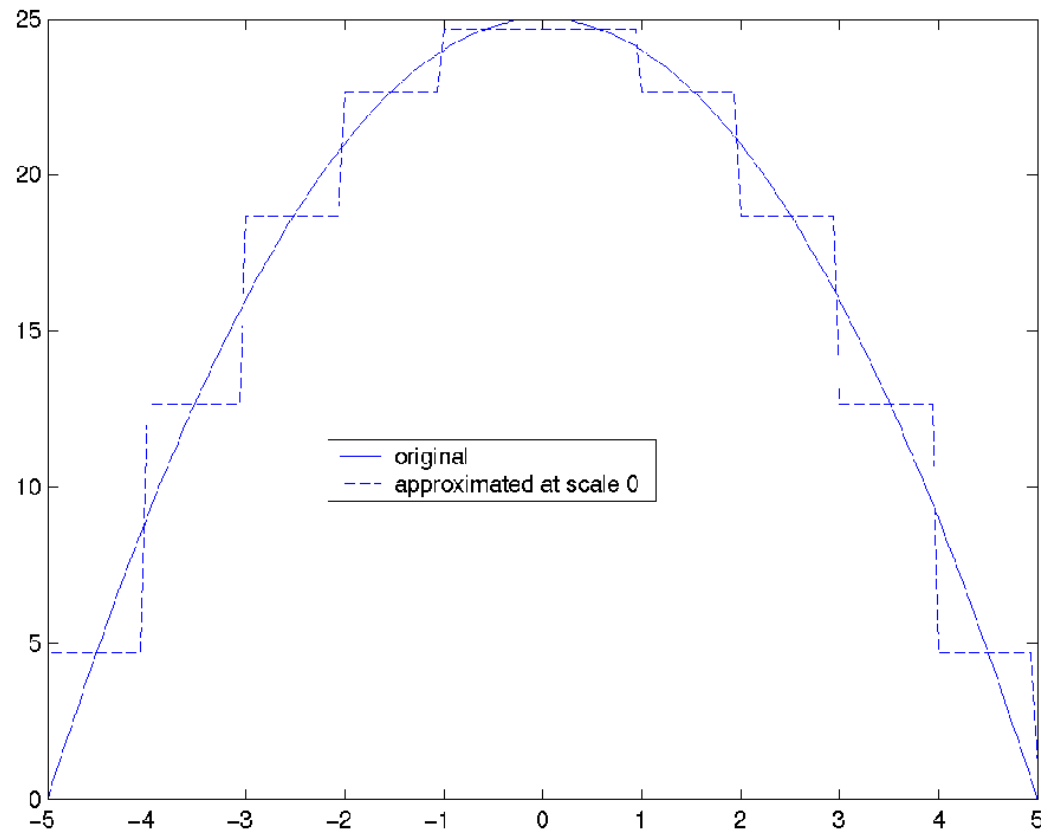
$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

- Dowolna funkcja może być aproksymowana przez liniową kombinację stałych funkcji:

$$\psi(t), \psi(2t), \psi(4t), \dots, \psi(2^k t), \dots$$

Jednowymiarowe falki Haara

- Falki Haara umożliwiają wielopoziomową aproksymację kompletnego sygnału wejściowego (strumienia danych)



Jednowymiarowe falki Haara

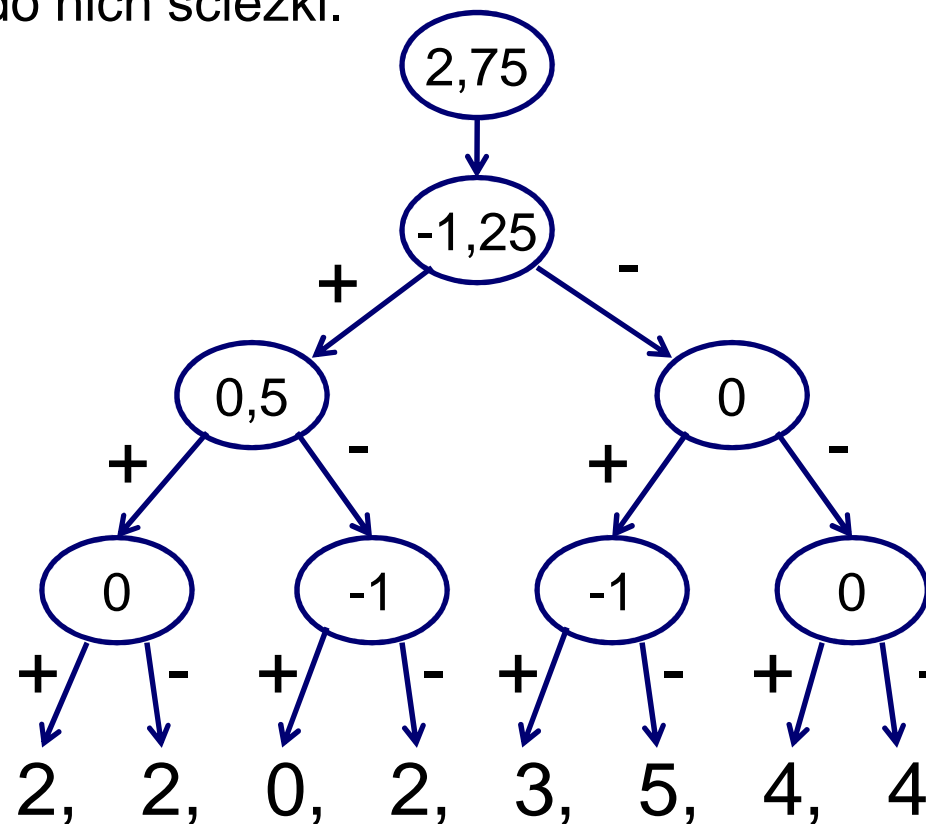
- **Falki**: matematyczne narzędzie do hierarchicznej dekompozycji funkcji i sygnałów (rozkładu wartości strumieni danych)
- **Falka Haara**: Najprostsza falka łatwa do zrozumienia i implementacji
 - *Rekurencyjne wyznaczanie średnich i różnic między sąsiednimi parami wartości na kolejnych poziomach ogólności*

Rozdzielczość	Średnie	Utracona informacja
3	[2, 2, 0, 2, 3, 5, 4, 4]	-
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[-1.25]

Dekompozycja falką Haara: [2.75, -1.25, 0.5, 0, 1, -1, -1, 0]

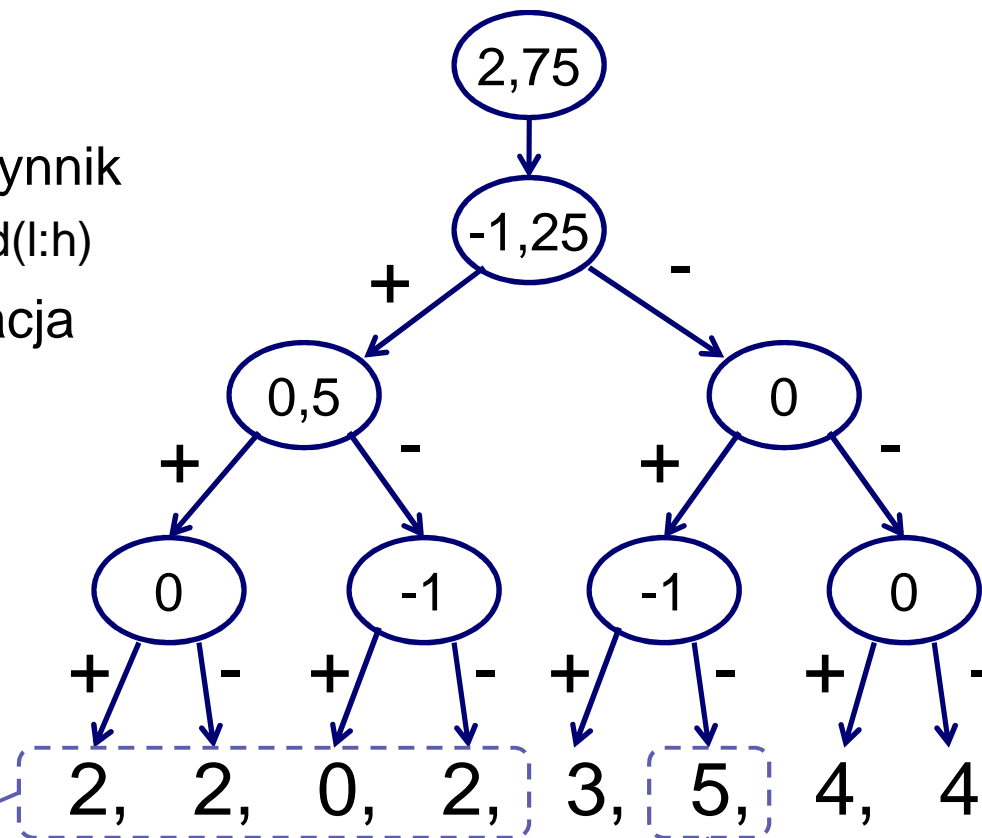
Dekompozycja – drzewo błędów

- Wynik dekompozycji umożliwia odtworzenie sygnału wyjściowego. Źródłowe wartości danych mogą być obliczone jako suma wartości wzdłuż prowadzącej do nich ścieżki.



Przetwarzanie drzewa błędów

- Rekonstrukcja źródłowych wartości danych $d(i)$
 - $d(i) = \sum (+/-1) * \text{współczynnik}$
- Obliczanie sum częściowych $d(l:h)$
 - $d(l:h)$ = liniowa kombinacja współczynników na ścieżkach od l do h
- Złożoność $O(\log N)$



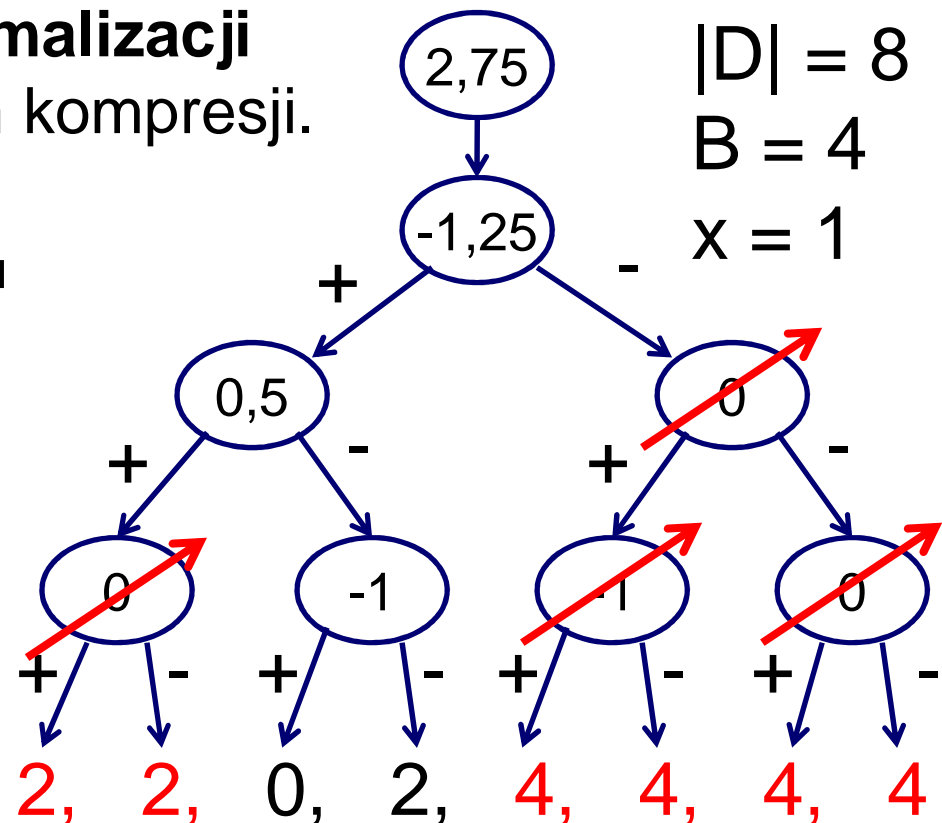
$$d(1:4) = 4 * 2.75 + 4 * (-1.25) = 6$$

$$d(6) = 2.75 - (-1.25) + 0 - (-1) = 5$$

Streszczenia za pomocą falek

Możliwości kompresji drzew dekompozycji do zadanego rozmiaru drzewa **przy minimalizacji błędów** będących wynikiem kompresji.

- Dla danego drzewa dekompozycji i zadanego obszaru pamięci B przeznaczonego dla streszczenia ($B \ll |D|$);
wyznacz wartość progową dla współczynników x , taką by:
istniało co najmniej $|D| - B$
współczynników o wartościach bezwzględnych mniejszych od $|x|$.
Usuwanie węzłów musi zachować poprawną strukturę drzewa.

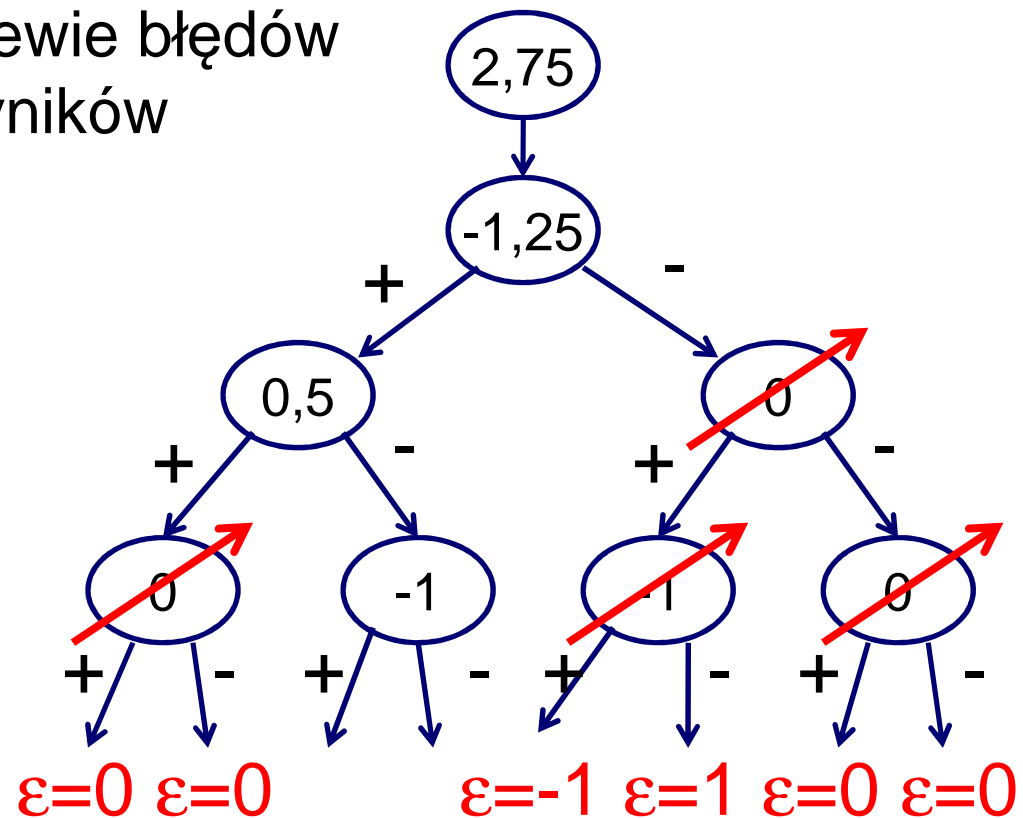


Błędy oszacowań

Celem przyjętej metody jest usuwanie węzłów w drzewie błędów minimalizujące błędy wyników zapytań.

- Błąd maksymalny jest równy maksymalnej sumie współczynników dla wszystkich gałęzi drzewa. Dla podanego przykładu maksymalny błąd wynosi:

$$|\varepsilon| = 1$$





Zastosowanie falki Haara

- Dekompozycja strumieni danych za pomocą falki Haara jest stosowana dla specyficznego modelu danych. Dla **skończonego** zbioru danych reprezentujących: stan urządzeń sieciowych, procesów przemysłowych, danych giełdowych i pozyskiwanych ze skończonego zbioru czujników lub sensorów, generowany jest nieskończony strumień modyfikacji wartości tych danych.
- Streszczenie w postaci skompresowanego drzewa błędów reprezentuje aktualny stan danych.
- Dla reprezentacji złożonych wartości danych stosuje się wielowymiarowe drzewa błędów.
- Drzewo błędów musi być utrzymywane dynamicznie dla kolejnych elementów strumienia



Szkicowanie

- **Techniki szkicowania (sketching techniques)**
 - Szkicowanie polega na budowaniu podsumowania strumienia danych przy użyciu małej ilości pamięci. Jest stosowane do szacowania odpowiedzi na pewne zapytania na zbiorze danych.
 - Metody szkicowania: CM (Count-Min) Sketch, Hash Sketch, projekcji liniowej AMS Sketch

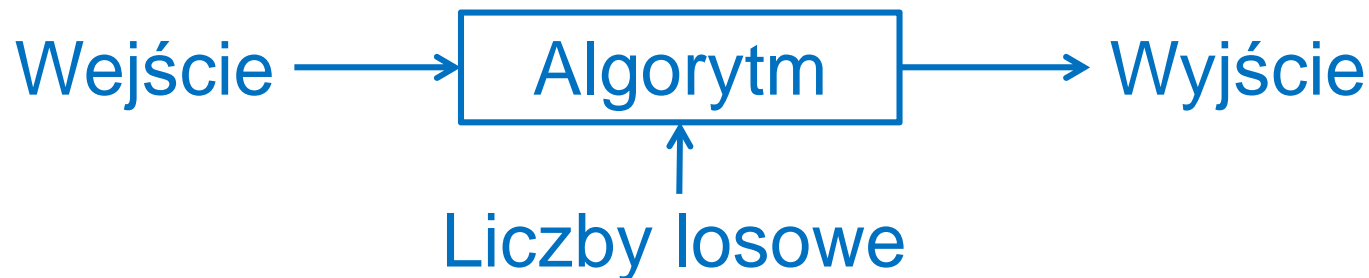
Algorytmy probabilistyczne

■ Algorytmy deterministyczne



Wyznaczają poprawne rozwiązanie problemu. Dla pewnych klas problemów ich wydajność może nie być wystarczająca.

■ Algorytmy probabilistyczne



Wyznaczają przybliżone, ale wystarczająco dobre rozwiązanie problemu. Dla tych samych danych wejściowych mogą generować różne wyniki, za to dla pewnych klas problemów mogą być prostsze i bardziej wydajne niż algorytmy deterministyczne.



Szkicowanie Count-Min

Model danych:

- Analizowane dane tworzą zmienny w czasie wektor o postaci:

$$\mathbf{a}(t) = [a_1(t), a_2(t), \dots, a_i(t), \dots, a_n(t)]$$

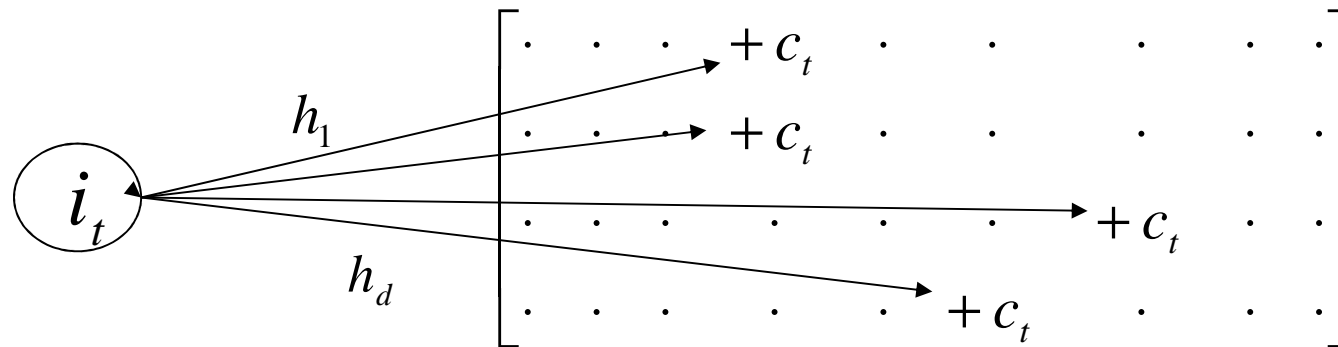
- W stanie początkowym wartości wszystkich składowych wektora są równe zeru.
- Zmiany stanu wektora tworzą strumień par o postaci:
 (i_t, c_t) , gdzie:
 - t – reprezentuje czas pojawienia zmiany,
 - i – jest indeksem elementu wektora,
 - c – jest modyfikacją tego elementu.
- Pojawienie się w strumieniu elementu (i_t, c_t) powoduje zmianę i -tego elementu wektora: $a_i(t) = a_i(t) + c_t$.
Pozostałe elementy wektora pozostają niezmiennione.

Szkicowanie Count-Min

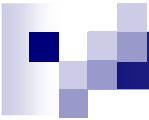
Szkic Count-Min o parametrach: ε - wielkość błędu oszacowania i δ , gdzie $(1 - \delta)$ to prawdopodobieństwo, że błąd jest $\leq \varepsilon$ obejmuje:

- dwu-wymiarową tabelę o szerokości: $w = \lceil e / \varepsilon \rceil$ (dla $\varepsilon = 0.05$, $w=55$) i wysokości: $d = \lceil \ln(1 / \delta) \rceil$ (dla $\delta = 0.01$, $d=5$).
- d-elementowego zbioru parami niezależnych funkcji haszowych:

$$h_1, \dots, h_d : \{ 1, \dots, n \} \rightarrow \{ 1, \dots, w \}$$



Kiedy w strumieniu wejściowym pojawia się element (i_t, c_t) w każdym wierszu tabeli jest modyfikowana kolumna wyznaczona przez odpowiednią funkcję mieszającą.



Rodzina niezależnych parami funkcji haszowych klasy 2-universal

- Niech $U = \{1, 2, \dots, m-1\}$ i $V = \{1, 2, \dots, n-1\}$
- Niech $p \geq m$ będzie liczbą pierwszą
- Funkcja haszowa: $h_{a,b}(u) = ((ax+b) \bmod p) \bmod n$
- Rodzina funkcji haszowych zdefiniowana jako:

$$H = \{ h_{a,b} \mid 1 \leq a \leq p-1, 0 \leq b \leq p \}$$

posiada własność 2-universal, i wszystkie funkcje z tej rodziny są parami niezależne.



Rozmiary szkicu

Rozmiary tabeli szkicu dla wybranego poziomu błędu ε i jego prawdopodobieństwa ($1 - \delta$):

	ε	0,2	0,15	0,1	0,09	0,08	0,07	0,06	0,05	0,04	0,03	0,02	0,01	0,005
$1 - \delta$	d/w	14	19	28	31	34	39	46	55	68	91	136	272	544
0,9	3	42	57	84	93	102	117	138	165	204	273	408	816	1632
0,96	4	56	76	112	124	136	156	184	220	272	364	544	1088	2176
0,99	5	70	95	140	155	170	195	230	275	340	455	680	1360	2720
0,995	6	84	114	168	186	204	234	276	330	408	546	816	1632	3264
0,999	7	98	133	196	217	238	273	322	385	476	637	952	1904	3808



Operacje dostępne na szkicu Count-Min

Szkice Count-Min umożliwiają obsługę następujących zapytań:

- zapytanie punktowe $Q(i)$ – zwracające wartość i -tego elementu wektora;
- zapytanie zakresowe $Q(i, j)$ – zwracające sumę wartości elementów wektora o indeksach z przedziału $i \div j$;
- iloczyn skalarny $Q(a, b)$ – zwracający wartości iloczynu skalarnego dwóch wektorów a i b ;
- wyznaczanie kwantyli rzędu p w zbiorze wartości wektora
- znajdowanie elementów częstych w zbiorze wartości wektora

Szacowanie zapytań

- Wynikiem oszacowania zapytania punktowego $Q(i)$ dla dodatnich modyfikacji elementów wektora jest wartość:

$$\bar{a}_i = \min_j \text{count}[j, h_j(i)]$$

Gwarantowana jakość oszacowania wynosi:

$$\bar{a}_i \leq a_i + \epsilon \|a\|_1$$

z prawdopodobieństwem $1 - \delta$.

- Wynikiem oszacowania zapytania punktowego $Q(i)$ dla dowolnych modyfikacji jest wartość:

$$\bar{a}_i = \text{median}_j \text{count}[j, h_j(i)]$$

Złożoność czasowa algorytmów zapytania punktowego i modyfikowania wektora danych jest stała, to jest niezależna od rozmiaru wektora. Złożoność zależy jednak od liczby wierszy w szkicu, czyli od dokładności szacowania, i jest równa:

$$O(\ln(1 / \delta))$$

Przykład

i	tab[i]	i	szk[1,i]	szk[2,i]
1	0	1	0	0
2	5	2	0	5
3	2	3	10	2
4	0	4	2	0
5	0	5	0	0
6	0	6	0	0
7	0	7	0	0
8	0	8	0	0
9	0	9	0	0
10	0	10	0	0
11	0	11	0	0
12	0	12	0	5
13	0	13	0	0
14	0	14	0	0
15	0			
16	5			
17	0			
18	0			
19	0			
20	0			

kolizja

$$\varepsilon = 0,2 \quad w = 14$$

$$\delta = 0,15 \quad d = 2$$

Funkcje haszowe:

- $h_1 : (n \bmod w) + 1$
- $h_2 : \lceil n * w / N \rceil$

Strumień wejściowy: (3,2), (2,5), (16,5)

$$Q(2) = \min \{ \text{szk}[1,3], \text{szk}[2,2] \} = \min \{ 1, 5 \} = 1$$

$$Q(3) = \min \{ \text{szk}[1,4], \text{szk}[2,3] \} = \min \{ 2, 2 \} = 2$$

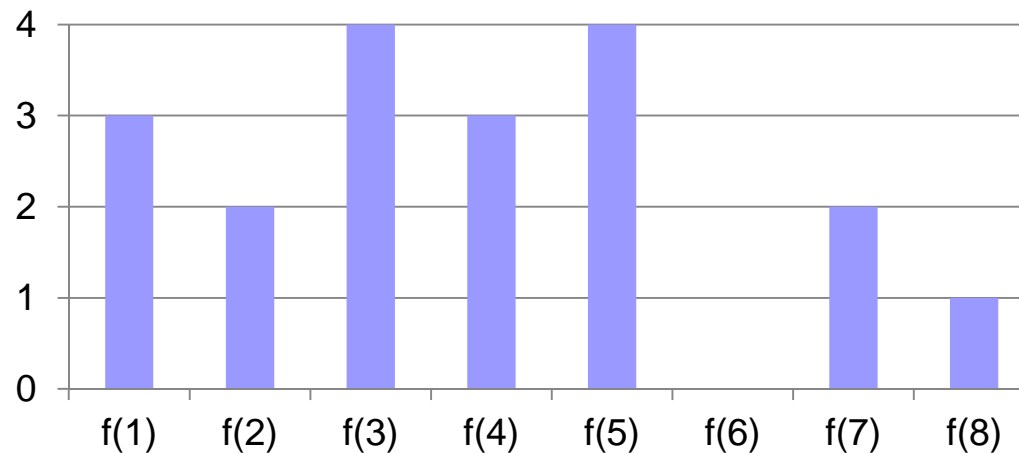


Szkice AMS

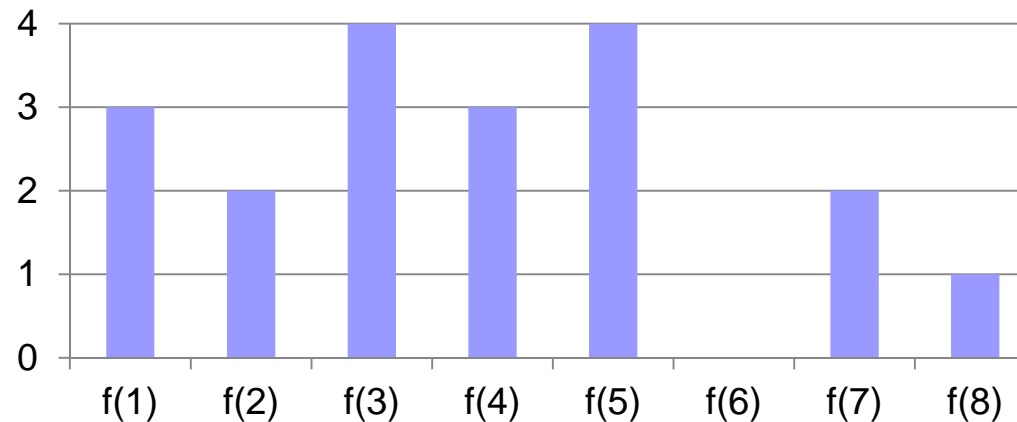
Dany strumień wejściowy:

■ $A(1, 2, 3, 4, 5, 5, 7, 1, 3, 4, 7, 2, 5, 8, 1, 3, 5, 3, 4)$

Rozkład wartości danych jest następujący:



Zastosowanie rozkładu wartości elementów strumienia



Rozkłady wartości elementów strumienia mogą być wykorzystywane do złożonej analizy statystycznej wartości elementów strumienia za pomocą momentów rozkładu.



Wyznaczanie momentów rozkładu

Bazuje na momentach częstości opisujących rozkład wartości elementów strumienia wejściowego

Dane są:

- strumień wejściowy $A(a_1, a_2, \dots, a_n)$, gdzie: $a_i \in N$ i $|N| < n$
- m_i są liczbami wystąpień poszczególnych wartości danych $m_i = |\{j : a_j = i\}|$

Definicja:

- Momentem częstości rzędu k nazywamy:

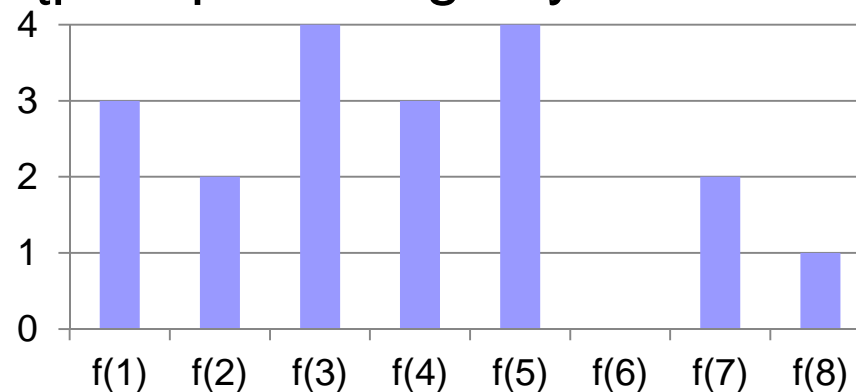
$$F_k = \sum_{i=1}^n m_i^k$$

Przykład

Dany strumień wejściowy:

■ $A(1,2,3,4,5,5,7,1,3,4,7,2,5,8,1,3,5,3,4)$

Liczby wystąpień poszczególnych wartości danych:



Momenty częstości danych:

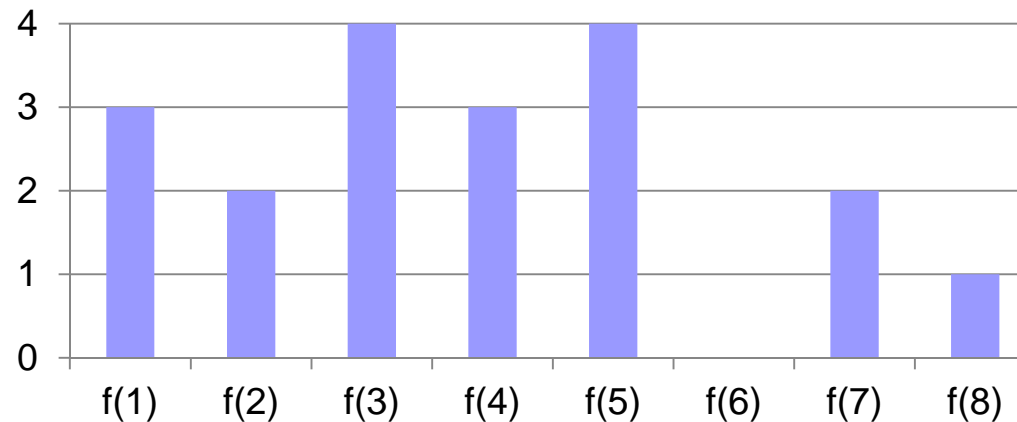
$F_0 = 7$ - liczba różnych wartości danych

$F_1 = 19$ (m) - liczba elementów strumienia wejściowego

$F_2 = 59$ – Gini indeks różnorodności danych

$F_\infty = 4$ – największa liczba wystąpień

Wykonywanie zapytań na rozkładzie danych



SELECT SUM(a) FROM A

$$\text{SUM}(a) = 1 * 3 + 2 * 2 + 3 * 4 + 4 * 3 + 5 * 4 + 6 * 0 + 7 * 2 + 8 * 1$$

$$\text{SUM}(a) = 73$$

Dokładne wyznaczenie wartości jest jednak zbyt kosztowne. Złożoność zajętości pamięci liniowa: $O(N)$;

Gdzie: $N = |\text{Domain}(x)|$



Szkicowanie AMS

- losowe rzutowanie liniowe

- Wyznaczanie momentów rozkładu danych strumienia wejściowego w ograniczonym obszarze pamięci z określonym błędem szacowania, poprzez zastąpienie wektora rozkładu danych $\mathbf{F}[\mathbf{n}]$, gdzie n jest liczbą różnych wartości danych, zredukowanym wektorem $S[m]$, gdzie $m \ll n$
- Losowe rzutowanie liniowe rozkładu wartości strumienia wejściowego jest rzutowaniem za pomocą iloczynu skalarnego wektora $\mathbf{F}[\mathbf{n}]$ i wektorów $\xi_i[m]$ niezależnych zmiennych losowych o *odpowiednim* rozkładzie.

$$S[i] = \sum_{j=1}^n F[j] * \xi_i[j]$$



Szkicowanie AMS - przykład

Dany wektor rozkładu wartości elementów strumienia:

$$\mathbf{F} = [2, 1, 3, 1]$$

Rozmiar szkicu $m=2$

Dwa wektory niezależnych zmiennych losowych o rozkładzie $N(0,1)$:

$$\xi_1 = [-0.45, -0.09, 0.10, 0.87]$$

$$\xi_2 = [-0.19, 0.73, -0.61, 0.21]$$

Wektor streszczenia reprezentującego rozkład danych:

$$\mathbf{s} = [0.18, -1.28]$$



Wydajne utrzymywanie szkiców

- Szkic rozkładu wartości strumienia wejściowego jest wyznaczany przez dodawanie do szkicu strumienia wartości losowej ξ_i , kiedykolwiek w strumieniu pojawi się wartość i .

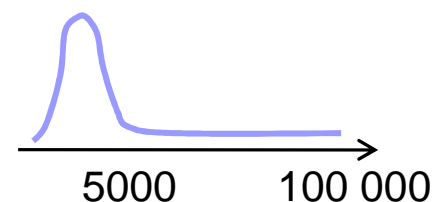
$$S_i(1,2,3,1,4,5,1,5, \dots) \rightarrow 3\xi_1 + \xi_2 + \xi_3 + \xi_4 + 2\xi_5$$

- Wartości ξ_i są generowane w pamięci o ograniczonym rozmiarze $\log(M)$ za pomocą generatorów pseudo-losowych przy gwarantowanym błędzie aproksymacji momentów rozkładu.

Histogramy

- **Histogramy** są popularnym w systemach baz danych narzędziem służącym do szacowania rozkładów danych

```
select * from pracownicy  
where płaça < 100 000
```



Oszacuj selektywność zapytania, przy założeniu, że:

- $\min(\text{płaça}) = 1200 \text{ zł}$
- $\max(\text{płaça}) = 250\,000 \text{ zł}$
- brak informacji o rozkładzie danych

Dla rozkładu równomiernego

$$\sigma = (100\,000 - 1200) / (250\,000 - 1200) = \mathbf{0,4}$$

40% pracowników zarabia poniżej 100 000 zł



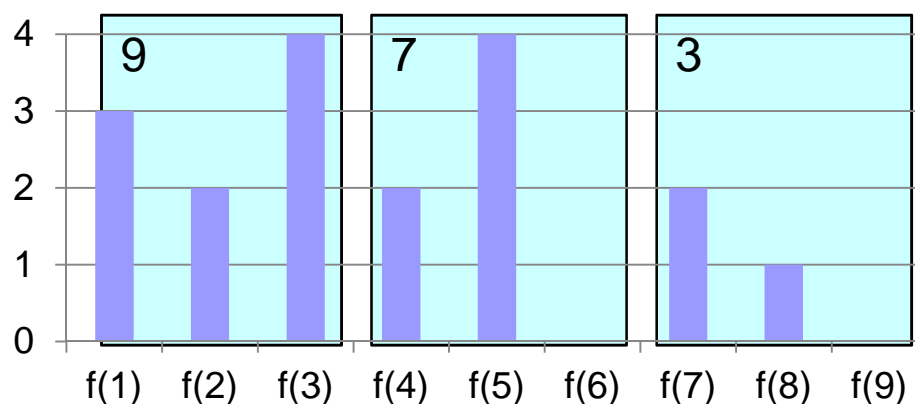
Histogramy

Celem jest utrzymywanie informacji o rozkładzie w efektywny sposób. Typy histogramów:

- **EquiWidth** - podział dziedziny na rozłączne przedziały o stałej długości:
 $<1 \div 100) - 5, <101 \div 200) - 42, <201 \div 300) - 79, \dots$
- **EquiDepth** - podział zbioru danych na n podzbiorów o stałej wielkości:
 $(58 \div 221)_1 - 100, (221 \div 335)_2 - 100, \dots, (510 \div 1281)_n - 100$
- **End-Biased** - najczęściej występujące wartości:
wartość 355 – 123 razy, 228 – 57, 118 – 25, pozostałe wartości – 459 razy

Wykorzystanie histogramów do szacowania wyników zapytań

■ Histogram **EquiWidth**



SELECT COUNT(*) FROM R WHERE a BETWEEN 3 AND 8 // =13

Bucket 1: $\langle 1 \div 3 \rangle \cap \text{BETWEEN } 3 \text{ AND } 8 \neq \emptyset \rightarrow (1/3) * 9 = 3$

Bucket 2: $\langle 4 \div 6 \rangle \subseteq \text{BETWEEN } 3 \text{ AND } 8 \rightarrow 7$

Bucket 3: $\langle 7 \div 9 \rangle \cap \text{BETWEEN } 3 \text{ AND } 8 \neq \emptyset \rightarrow (2/3) * 3 = 2$

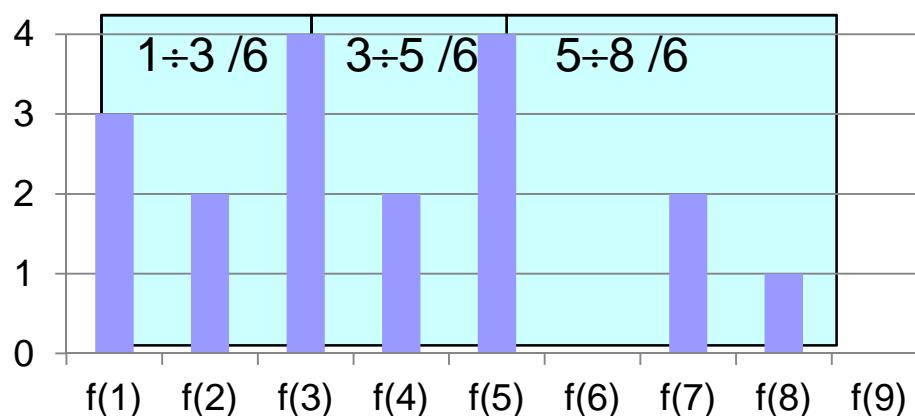
Oszacowanie $\text{COUNT} (*) = 12 \rightarrow \text{błąd} = 1$

Maksymalny błąd = $|\text{bucket}_{\text{gr i}}| + |\text{bucket}_{\text{gr j}}|$

- przedziały zawierające dolną i górną granicę zakresu zapytania

Wykorzystanie histogramów do szacowania wyników zapytań

■ Histogram **EquiDepth**



SELECT COUNT(*) FROM R WHERE a BETWEEN 3 AND 8 // =13

Bucket 1: $\langle 1 \div 3 \rangle \cap \text{BETWEEN } 3 \text{ AND } 8 \neq \emptyset \rightarrow (1/3) * 6 = 2$

Bucket 2: $\langle 3 \div 5 \rangle \subseteq \text{BETWEEN } 3 \text{ AND } 8 \rightarrow 6$

Bucket 3: $\langle 5 \div 8 \rangle \subseteq \text{BETWEEN } 3 \text{ AND } 8 \rightarrow 6$

Oszacowanie $\text{COUNT} (*) = 13 \rightarrow \text{błąd} = 1$

Maksymalny błąd = $2 * |\text{bucket}|$

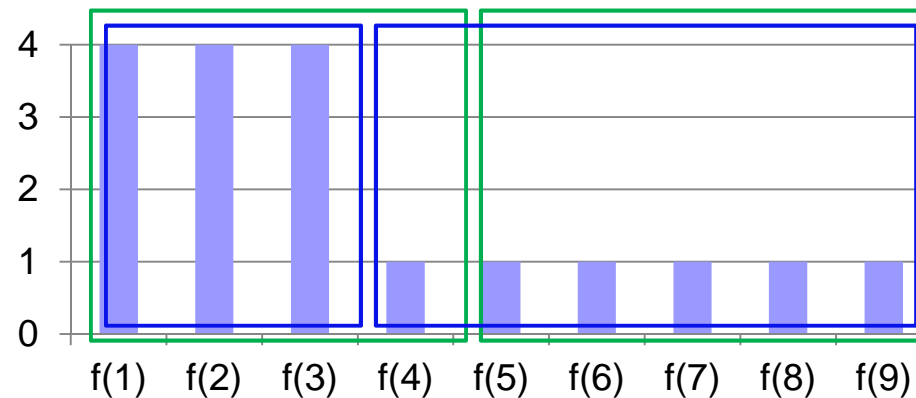
Dokładność szacowania zapytań

Histogram v-optymalny

Optymalny podział danych uwzględnia minimalizację ważonej wariancji w poszczególnych przedziałach:

$$W = \sum_i^m n_i * Var_i$$

gdzie: n_i jest liczbą danych, a Var_i wariancją podzbioru i ,



$$W = 4 * 2,25 + 5 * 0 = 9$$

`SELECT COUNT(*) FROM R WHERE a BETWEEN 3 AND 8 // = 9`

$$\text{COUNT} (*) = 1/2 * 13 + 4/5 * 5 = 10,5$$

$$W = 3 * 0 + 6 * 0 = 0$$

$$\text{COUNT} (*) = 1/3 * 12 + 5/6 * 6 = 9$$



Histogramy ekspotencjalne

- Problem: wyznaczać na bieżąco liczbę elementów strumienia spełniających odpowiedni warunek logiczny, np., że dana ramka w sieci pochodzi z określonego źródła w przesuwным oknie o rozmiarze N .

W takim wypadku, strumień może być zredukowany do strumienia bitów, w którym 1 oznaczają elementy spełniające dany warunek, a 0 elementy nie spełniające danego warunku.

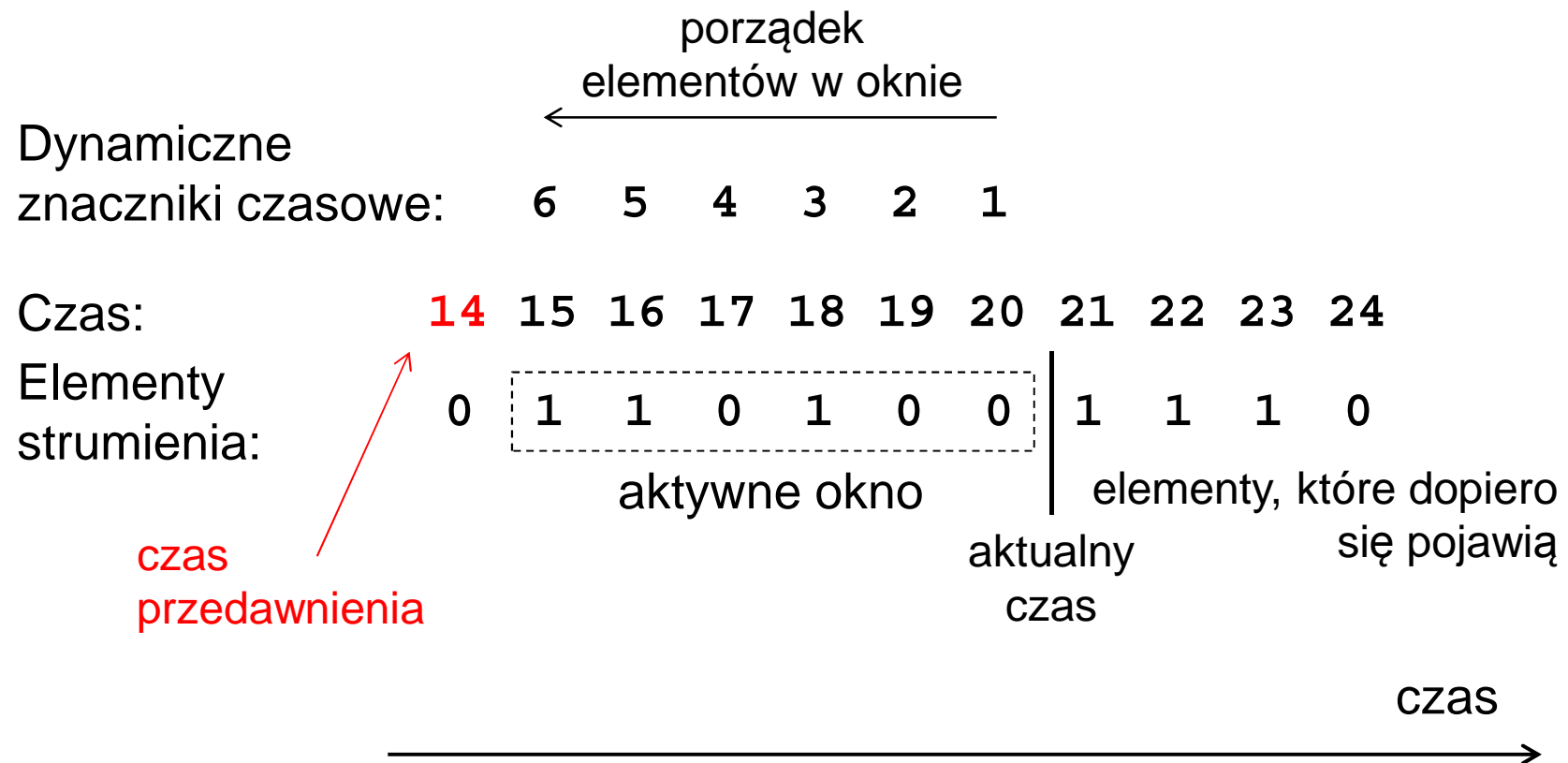
- Rozwiązanie naiwne

Utrzymywanie okna jako kolejki FIFO. Zmienna **Counter** jest zwiększana dla '1' pojawiających się na wejściu okna i zmniejszana dla '1' opuszczających okno.

Dokładne wyznaczanie wyniku wymaga utrzymywania w buforze kompletnego strumienia. Dla dużych wartości N może to być niemożliwe.

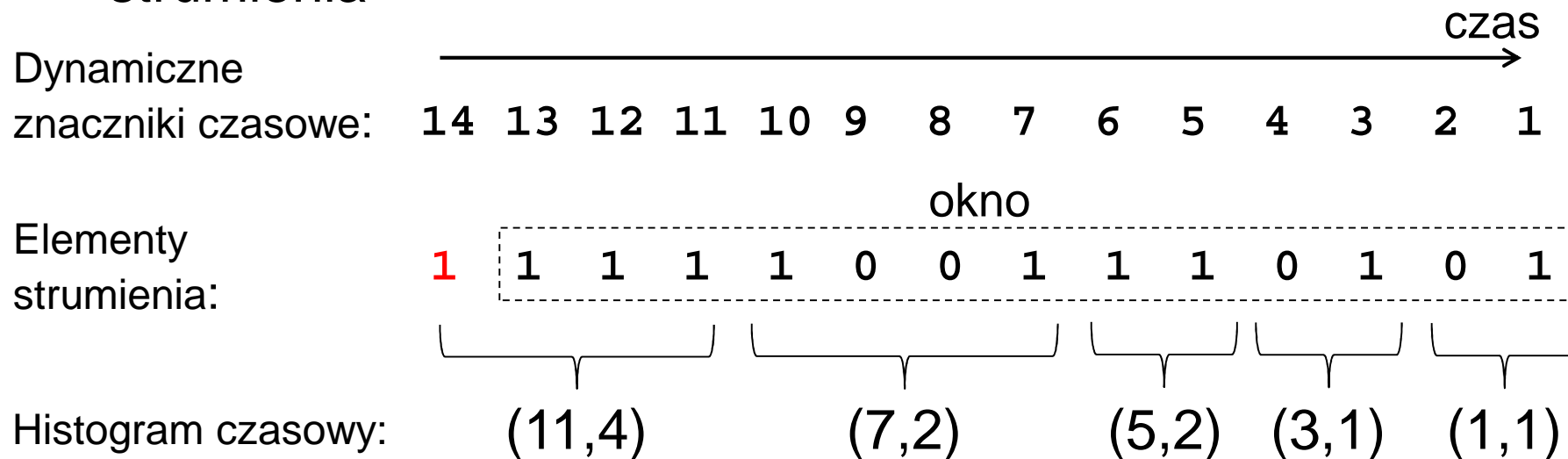
Histogramy eksponentencjalne

- Model ruchomego okna danych



Histogramy czasowe

- Histogram reprezentujący czasowy rozkład '1' w oknie strumienia



Przedziały histogramu mają następującą strukturę: $(timestamp_i, rozmiar_i)$

- $timestamp_i$ - znacznik czasowy najświeższej jedynek w przedziale,
- $rozmiar$ przedziału C_i - równy liczbie jedynek w tym przedziale.

Oszacowanie liczby '1': $\sum_{i=1}^{m-1} C_i + \frac{1}{2} C_m$

Szacunkowa liczba '1' = $1 + 1 + 2 + 2 + \frac{1}{2} \cdot 4 = 8$



Histogramy ekspotencjalne

Konstrukcja histogramów ekspotencjalnych opiera się na znalezieniu kompromisu między:

- maksymalnym błędem oszacowania ε ,
- a wielkością histogramu m .

Maksymalny bezwzględny błąd oszacowania jest równy $1/2 C_m$, gdzie C_m jest rozmiarem ostatniego przedziału histogramu. Błąd względny oszacowania jest co najwyżej równy: $(C_m/2)/(1 + \sum_{i=1}^{m-1} C_i)$.

Wąskie przedziały gwarantują mniejsze błędy oszacowań, ale dla danego rozmiaru okna, wymagają większej liczby przedziałów i tym samym zwiększają rozmiar histogramu.

Dla danego histogramu możliwe jest bezstratne sklejanie sąsiednich przedziałów, np. :

$$(7,2) + (5,2) = (5,4)$$



Histogramy eksponentialne

Niezmienniki histogramów eksponentialnych

- **Niezmiennik gwarantujący maksymalny błąd oszacowania**

Rozmiary poszczególnych przedziałów C_1, \dots, C_m spełniają zawsze zależność: dla każdego $j \leq m$, $(C_j/2)/(1 + \sum_{i=1}^{j-1} C_i) \leq \frac{1}{k}$, gdzie $k = \left\lceil \frac{1}{\varepsilon} \right\rceil$.

Każdy z przedziałów histogramu, będzie kiedyś ostatnim przedziałem.

- **Niezmienniki minimalizujące wielkość histogramu**

Rozmiary przedziałów spełniają zależność: $C_1 \leq C_2 \leq \dots \leq C_{m-1} \leq C_m$.

Rozmiary przedziałów są ograniczone do potęgi 2: $1, 2, 4, 8, \dots, 2^{m'}$ dla $m' \leq m$ i $m' \leq \log \frac{2N}{k} + 1$.

Dla każdego przedziału, za wyjątkiem pierwszego i ostatniego, istnieje co najwyżej $k/2+1$ i co najmniej $k/2$ przedziałów o tym samym rozmiarze.



Algorytm utrzymania histogramu

- Kiedy pojawia się nowy element w strumieniu wyznacz nowy czas przedawnienia. Jeżeli znacznik czasowy ostatniego przedziału jest większy lub równy od czasu przedawnienia usuń ten przedział i wskaź przedostatni przedział jako ostatni. Zmniejsz liczbę przedziałów histogramu.
- Jeżeli nowy element jest równy 0, zignoruj go, w przeciwnym wypadku utwórz nowy przedział o rozmiarze równym 1. Zwiększ liczbę przedziałów.
- Przeglądaj listę przedziałów, począwszy od najnowszych. Jeżeli znajdziesz $k/2+2$ (lub $k+2$ o rozmiarze 1) przedziałów o tym samym rozmiarze, połącz najstarsze dwa z nich. Następnie zmodyfikuj liczbę przedziałów.

32, 32, 16, 8, 8, 4, 2, 1, 1

32, 32, 16, 8, 8, 4, 2, 1, 1, 1 (nowa 1)

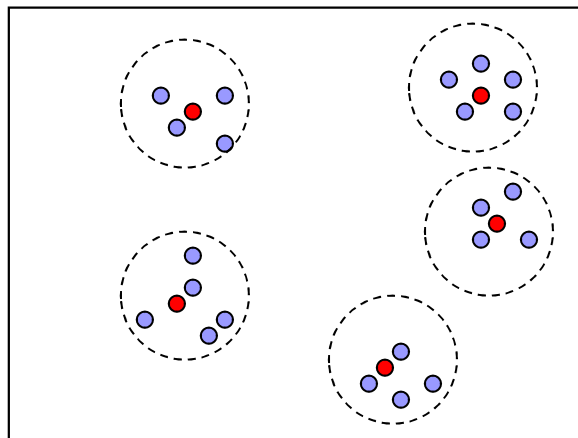
32, 32, 16, 8, 8, 4, 2, 1, 1, 1, 1 (nowa 1)

32, 32, 16, 8, 8, 4, 2, 2, 1, 1 (łącz przedziały)

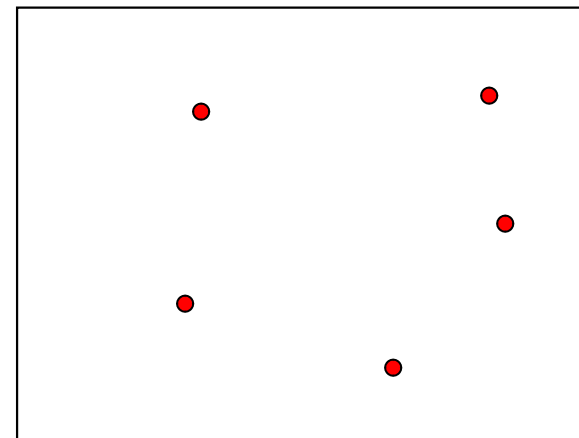
Micro-Clusters

■ Co to są mikro-grupy?

Mikro-grupa jest zbiorem elementarnych punktów danych, które znajdują się na tyle blisko siebie, że są traktowane jako pojedyncze punkty w dalszej analizie makro-grup. Pozwala to na redukcję rozmiaru pamięci potrzebnej na pamiętanie wszystkich punktów.



Perspektywa mikro-grup



Perspektywa makro-grup



Micro-Clusters

■ Jak pamiętane są mikro-grupy

Mikro-grupa jest reprezentowana przez uporządkowaną piątkę:

$$\overline{CFT}(C) = (\overline{CF2^x}, \overline{CF1^x}, \overline{CF2^t}, \overline{CF1^t}, n)$$

- Dla każdego wymiaru, suma kwadratów wartości danych jest utrzymywana w $CF2^x$. Stąd $CF2^x$ zawiera d wartości.
- Dla każdego wymiaru, suma wartości danych jest utrzymywana w $CF1^x$. Stąd $CF1^x$ zawiera d wartości.
- Suma kwadratów znaczników czasowych jest utrzymywana w $CF2^t$.
- Suma znaczników czasowych jest utrzymywana w $CF2^t$.

StreamBase

Uniwersalny komercyjny DSMS

The screenshot displays the StreamBase Studio environment. The main workspace shows a diagram for processing NYSE feed data. The diagram includes an input stream 'NYSE_Feed' connected to a 'Query' operator, which then feeds into a 'Filter' operator labeled 'IsNewBestAsk'. This is followed by a 'Map' operator labeled 'PreserveAskInfo', which outputs to 'BestAsks'. A similar path exists for 'IsNewBestBid' and 'PreserveBidInfo' leading to 'BestBids'. A central 'Query Table' labeled 'Bids_and_Aasks' is connected to both filter operators. The 'Properties (Update_Bids_and_Aasks)' panel is open, showing the 'General' tab with the 'Type of write' set to 'Update'. The 'Values to Update' table is as follows:

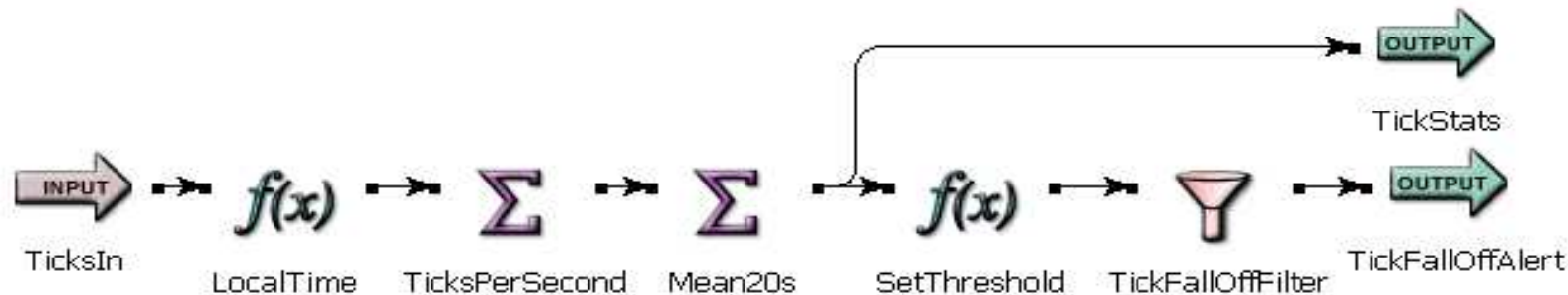
Field ...	Type	Size	Expression
best_bid	double	8	if bid_price > best_bid then bid_price else best_bid
best_ask	double	8	if ask_price < best_ask then ask_price else best_ask

The 'Operators' palette on the left includes Map, Filter, Aggregate, BSort, Union, Join, Gather, Merge, Query, Lock, Unlock, Heartbeat, Metronome, Java, and Split. The 'Data Constructs' section includes Streams, Adapters, and References. The 'Getting Started' sidebar on the right provides an introduction and a list of topics to explore.

Bids_and_Aasks (Query Table)
Type: in memory
Primary Index (hash): symbol
Schema:
- symbol string(9)
- best_bid double(8)
- best_ask double(8)

Hold CTRL to toggle parameters/description view
No application is running

StreamBase



```
CREATE STREAM TicksWithTime AS
```

```
SELECT *, now() AS LocalTime FROM TicksIn;
```

```
CREATE STREAM TicksPerSecond AS
```

```
SELECT openval() AS StartOfTimeSlice, count() AS  
NumberTicks, FeedName
```

```
FROM TicksWithTime [SIZE 1 ON LocalTime  
PARTITION BY FeedName]
```

```
GROUP BY FeedName;
```



StreamBase

- Następujący przykład pokazuje buforowanie strumienia danych w .NET:

```
using StreamBase.SB;
using StreamBase.SB.Client;
...
// connect to the StreamBase Server.
string uri = "sb://localhost:10000";
int buf_size = 100;
int flush_interval = 1000;
StreamBaseClient client = new StreamBaseClient(uri);
client.EnableBuffering(buf_size,flush_interval);
```