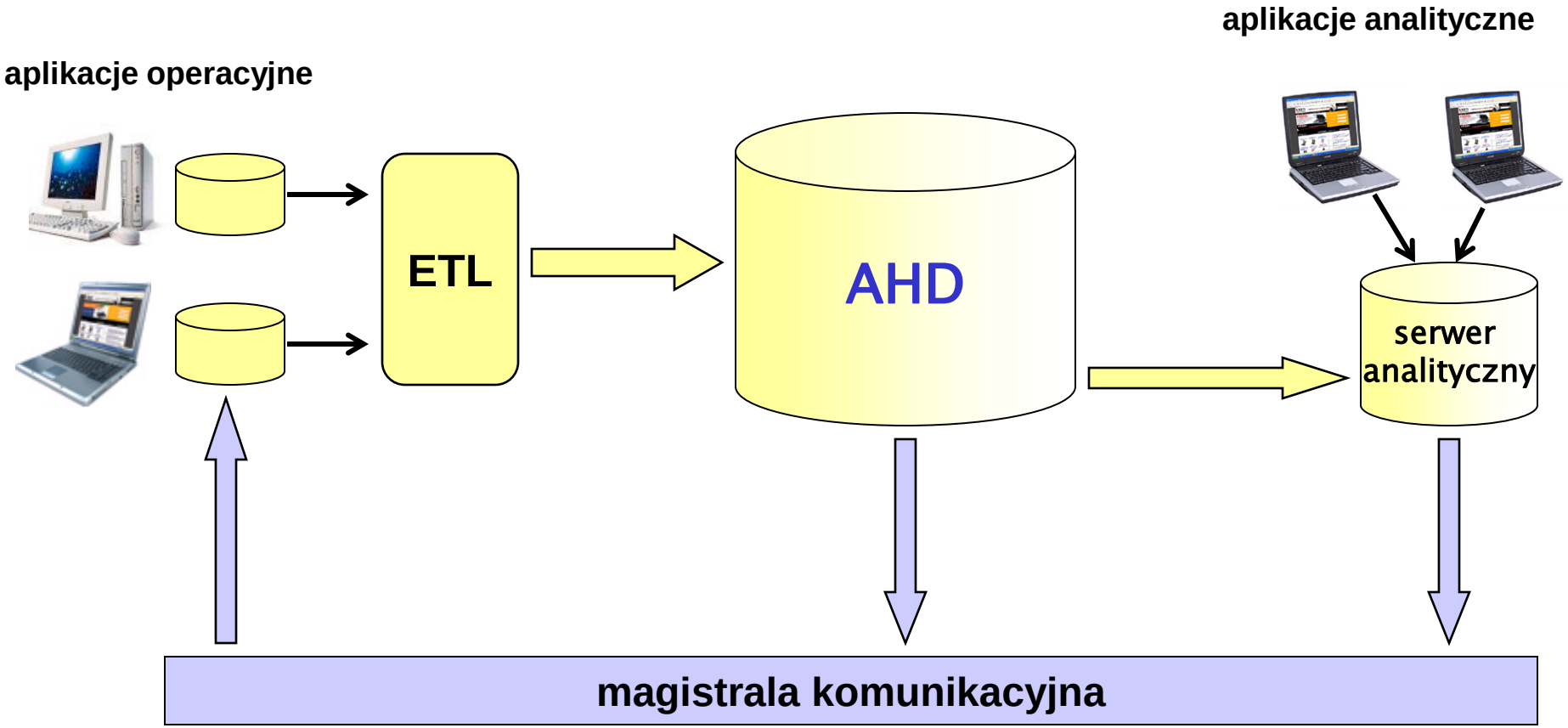
Hurtownie danych - przegląd technologii

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Aktywna hurtownia danych

- **AHD** [T. Thalhammer, M. Schrefl, and M. K. Mohania. Active data warehouses: complementing OLAP with analysis rules. Data Knowledge Engineering, 39(3), 2001]
  - **hurtownia automatycznie reaguje na "zdarzenia"**
    - **wykrycie anomalii**
    - **obniżenie sumarycznej wartości sprzedaży poniżej zadanego progu**
  - **zapytania monitorujące (subscription queries)**
  - **dane analityczne dostępne również dla pracowników operacyjnych**
- **Zastosowania**
  - **bankowość - analiza wykorzystania kart kredytowych**
  - **handel**

Architektura systemu AHD



AHD - technologia

- Dużo większa częstotliwość odświeżania danych
- Jednoczesna praca wielu procesów odświeżających i wielu zapytań analitycznych
- Implikacje
  - nowe modele transakcji
  - zapewnienie spójności zapytań analitycznych
  - optymalizacja tradycyjnych zapytań OLAP i zapytań monitorujących działających w tym samym czasie
  - optymalizacja struktury hurtowni danych

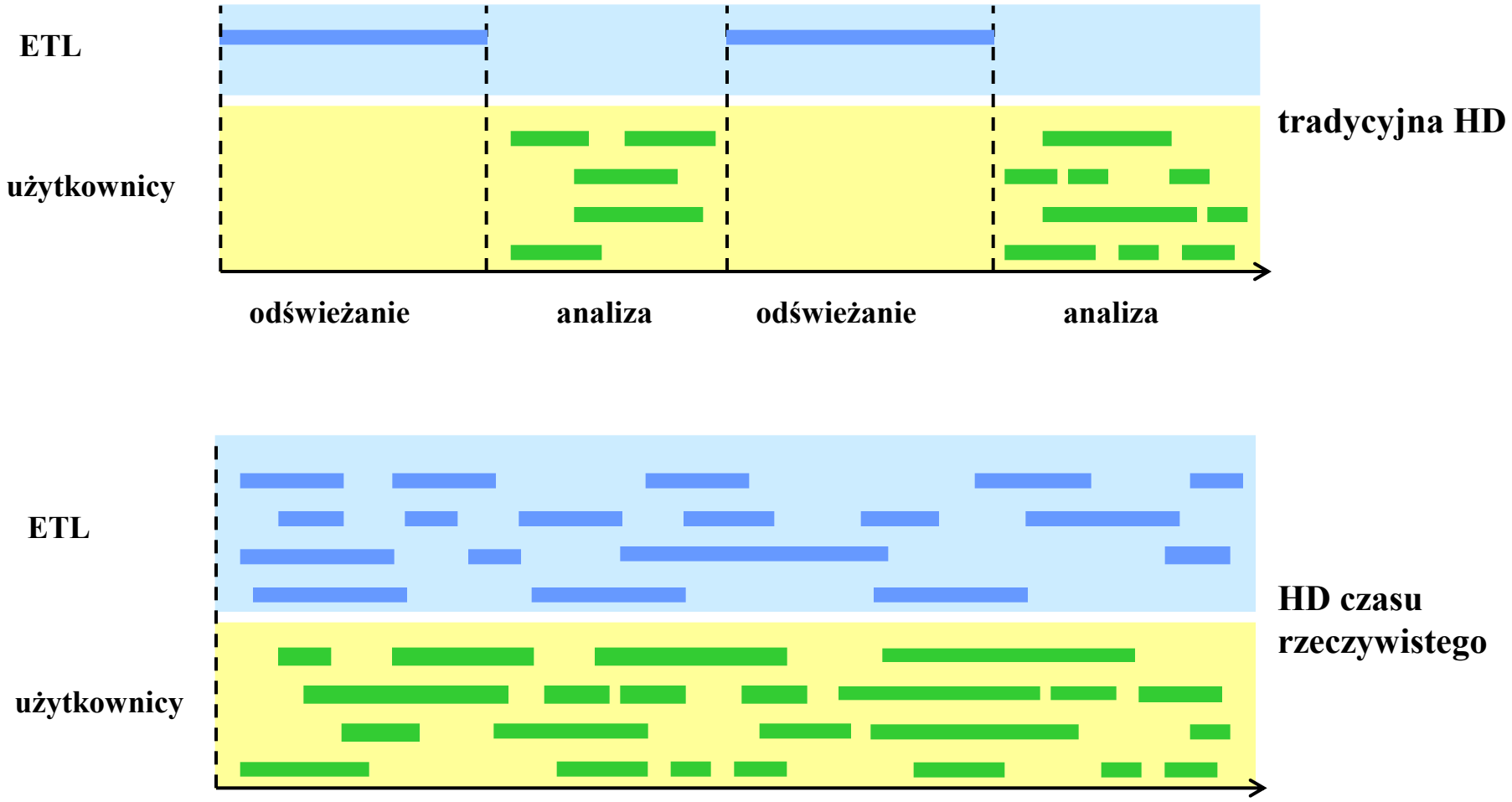
HD czasu rzeczywistego

- Okres propagacji danych ze źródła do HD CzRz
  - minuty
- Zastosowania
  - bankowość
    - wykrywanie transakcji podejrzanych (skradziona karta) wymaga natychmiastowej reakcji
  - zarządzanie ruchem
    - monitorowanie natężenia ruchu i wykrywanie potencjalnych korków
- ETL czasu rzeczywistego

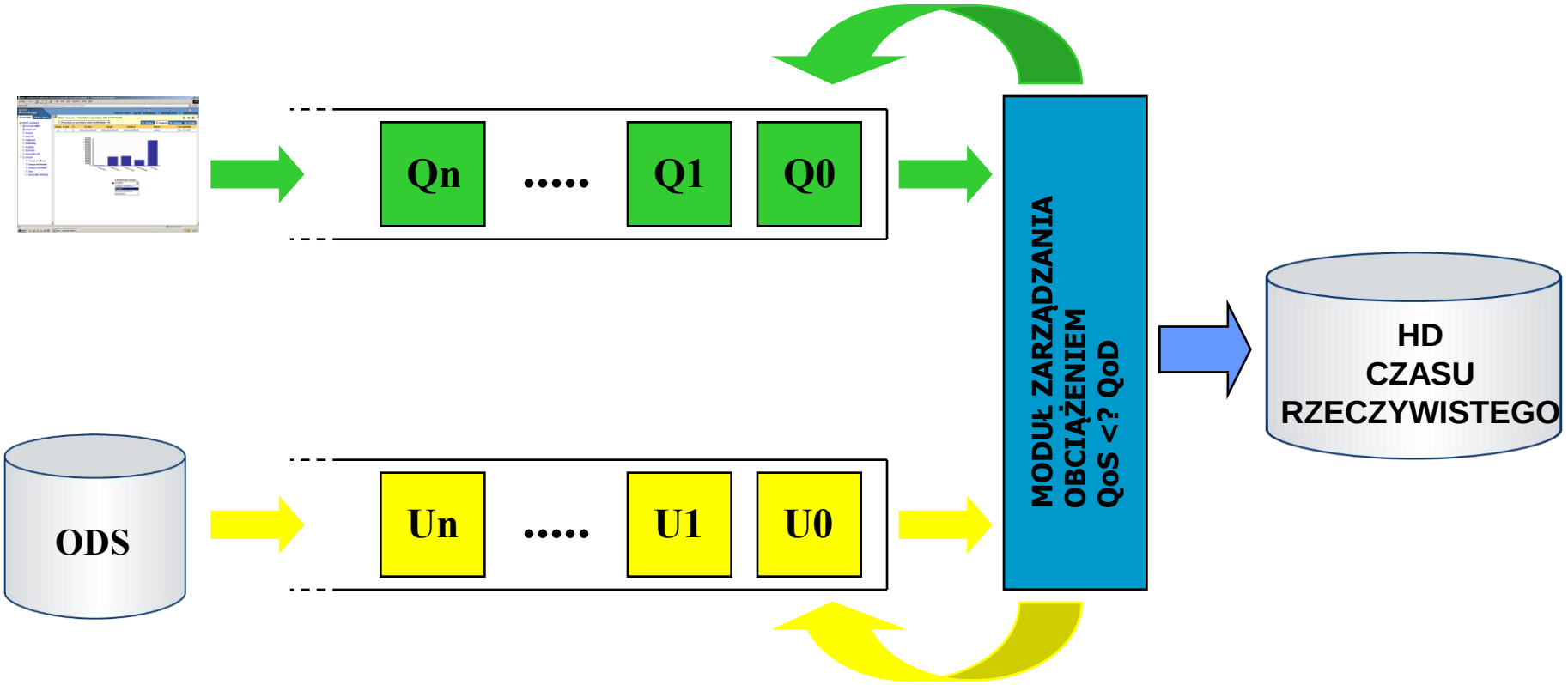
HD czasu rzeczywistego

- **Wymagania**
  - **krótki czas odpowiedzi na zapytania analityczne (Quality of Service - QoS)**
  - **aktualność danych (Quality of Data - QoD)**
    - **zmiana danych w źródle propaguje się natychmiast (z niewielkim opóźnieniem) do HD ⇨liczba niezaaplikowanych modyfikacji jak najniższa**
    - **źródło wysyła dane (push-based)**
  - **odpowiednie uszeregowanie operacji odświeżania i zapytań analitycznych**
- **Zastosowanie**
  - **bankowość - analiza operacji kartami płatniczymi i wykrywanie podejrzanych operacji**
  - **międzynarodowe firmy (brak okresu bezczynności użytkowników)**

HD czasu rzeczywistego



Architektura



Architektura

- Zarządzanie kolejkami zapytań i modyfikacji
  - FIFO
  - FIFO z preferencją zapytań
  - FIFO z preferencją modyfikacji
  - WINE (Work Balancing by Election)

WINE (1)

- **Miary wydajności**
  - **QoS - średni czas pozostawania zapytania w systemie = czas wykonania–czas przybycia**
    - **rt_{qi} - czas przebywania zapytania qi**
    - **$|W_q|$ - długość obciążenia**

$$QoS(W) = \sum_{q_i \in W_q} \frac{rt_{q_i}}{|W_q|}$$

  - **QoD - liczba niezrealizowanych odświeżeń dla zapytania (odświeżenia oczekujące w momencie pojawienia się zapytania)**
    - **uu(pi) - niezrealizowane odświeżenia partycji pi**

$$QoD(q) = \min_{p_i \in P_q} \left(\frac{1}{1 + uu(p_i)} \right)$$

    - **dla całego obciążenia**

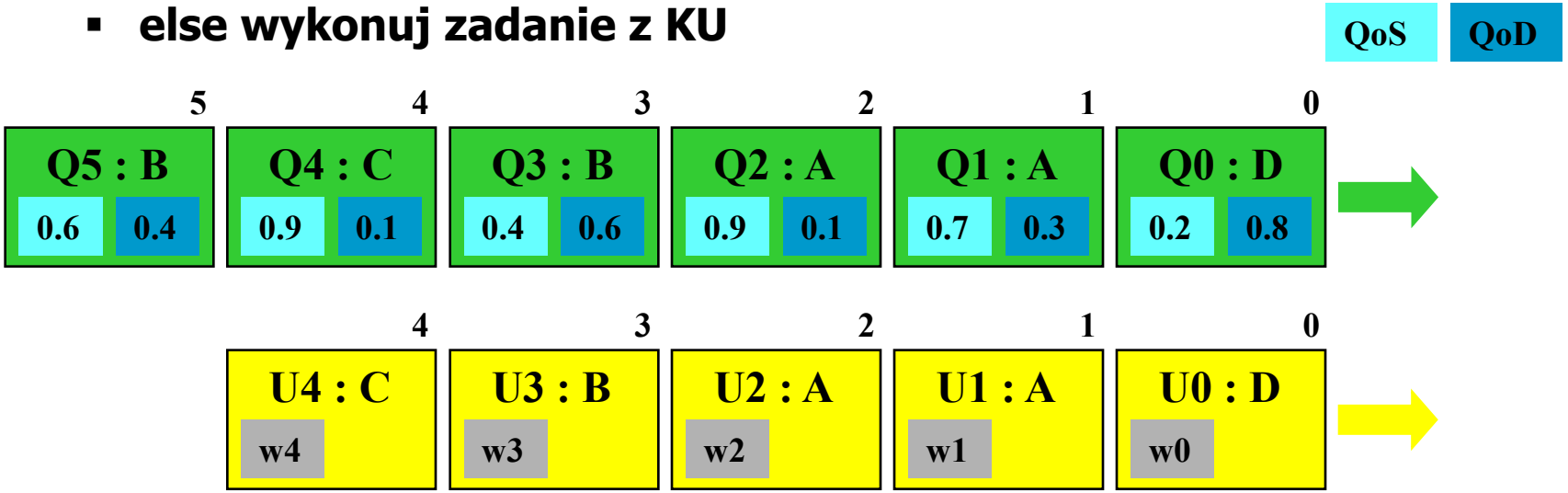
$$QoD(W) = \frac{1}{|W_q|} \sum_{q_i \in W_q} QoD(q_i)$$

WINE (2)

- **Cel optymalizacji**
  - **minimalizacja czasu przebywania zapytania w systemie**
  - **maksymalizacja aktualności danych**
- **Kolejka zapytań (KQ) i kolejka odświeżeń (KU)**
  - **każde zadanie w kolejce ma wartość QoD i QoS**
- **2 poziomowe szeregowanie operacji**
  - **poziom 1:**
    - **alokowanie zasobów albo dla KQ albo dla KU (suma preferencji QoS lub QoD dla kolejki)**
  - **poziom 2:**
    - **szeregowanie zapytań w KQ zgodnie z QoS**
    - **szeregowanie odświeżeń w KU zgodnie z QoD**

WINE (3)

- **Poziom 1**
  - **if** $\sum QoS > \sum QoD$ **then wykonuj zadanie z KQ**
  - **else wykonuj zadanie z KU**



- **Warunek** $\sum QoS > \sum QoD$ **jest sprawdzany po wykonaniu każdego zadania**

WINE (4)

- 2 poziom - kolejka zapytań
  - sortowanie zapytań:
    - order by QoS desc, timestamp asc
  - aby zapobiec blokowaniu zapytań z niską wartością QoS ⇨ zwiększanie wartości QoS zapytań znajdujących się już w kolejce o wartość d, po każdym wykonaniu zapytania z kolejki
    - |Q| - długość kolejki
    - r_{max} - parametr systemowy

$$d = \frac{1}{|Q| \cdot r_{max}}$$

WINE (5)

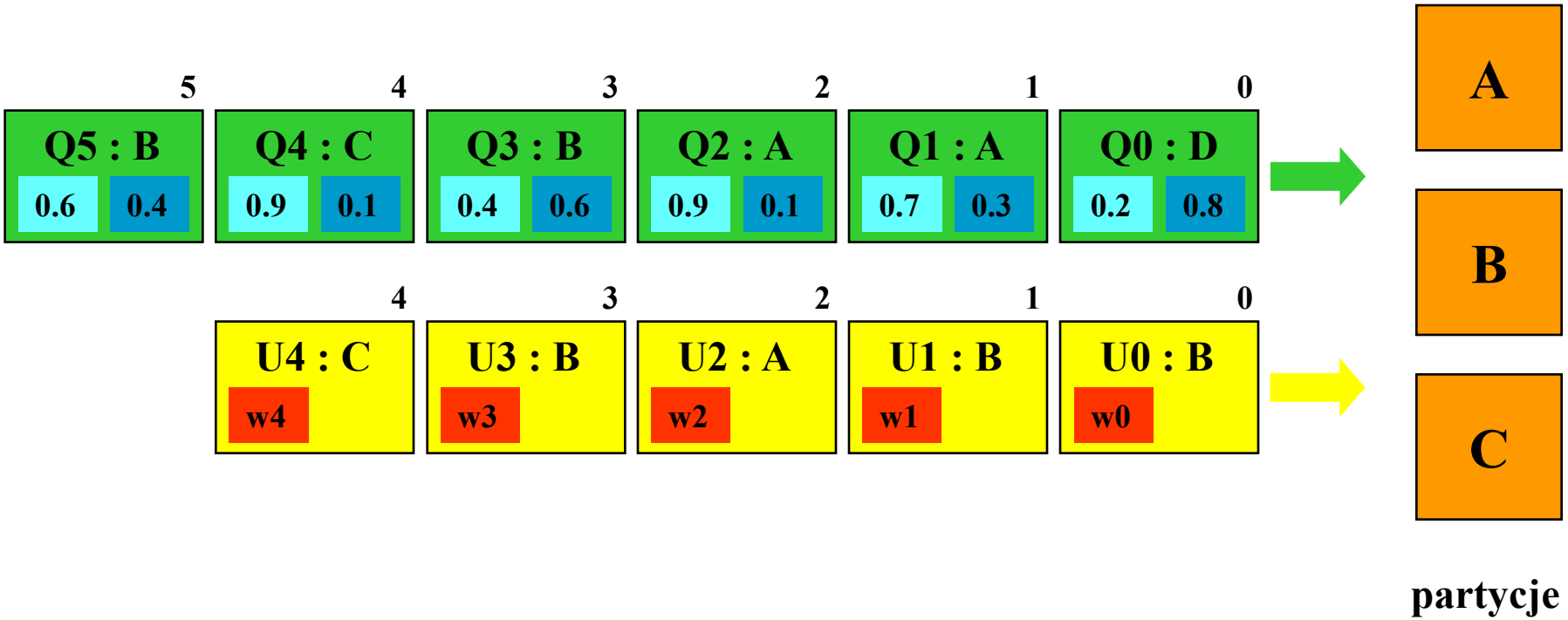
- 2 poziom - kolejka odświeżeń
- Preferowane są odświeżenia, z których będą korzystały zapytania uruchamiane za chwilę (na pocz. KQ) ⇨ w tym celu wprowadza się wagę odświeżenia

$$w(u_i) = \sum_{\forall q_i, P_{q_i} \cap P_u \neq \emptyset} \frac{QoD_{Q_i}}{1 + pos_{Q_i}}$$

- Zapytanie i uaktualnienie adresują tę samą partycję danych
- Na QoD uaktualnienia partycji P_i ma wpływ pozycja zapytania w kolejce odwołującego się do P_i
  - im zapytanie bliżej początku kolejki tym QoD uaktualnienia otrzymuje wyższą wartość

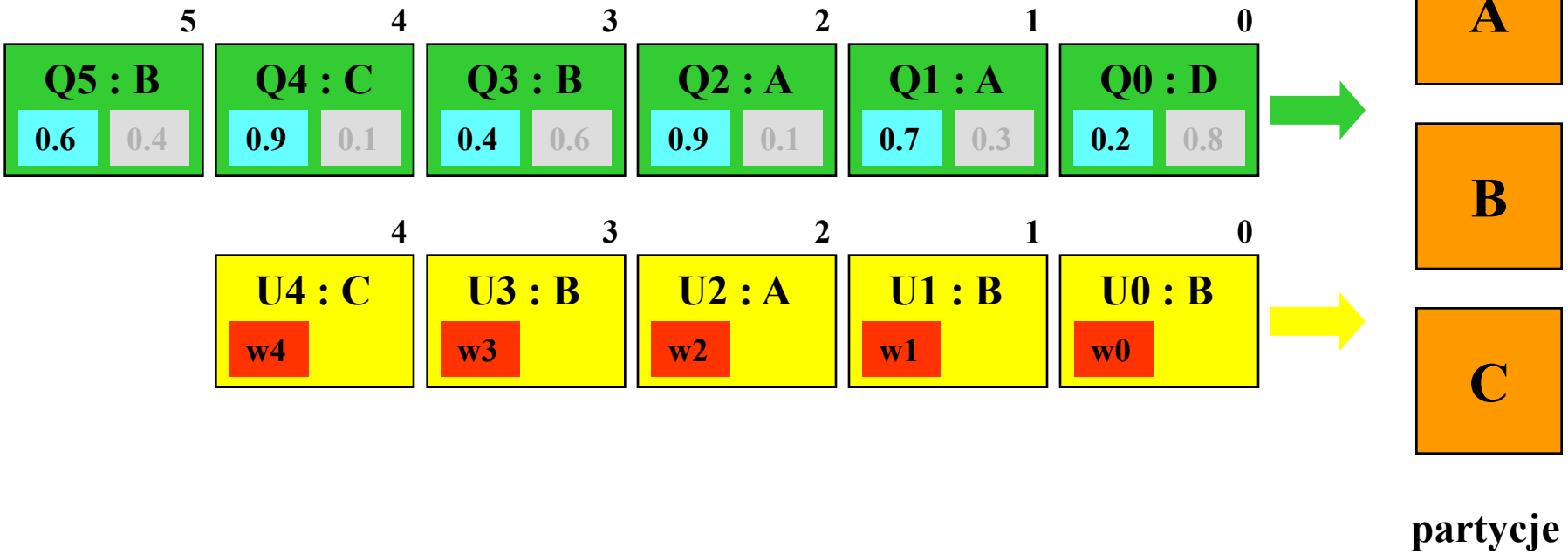
WINE (6)

- suma QoS=3,7
- suma QoD=2,3



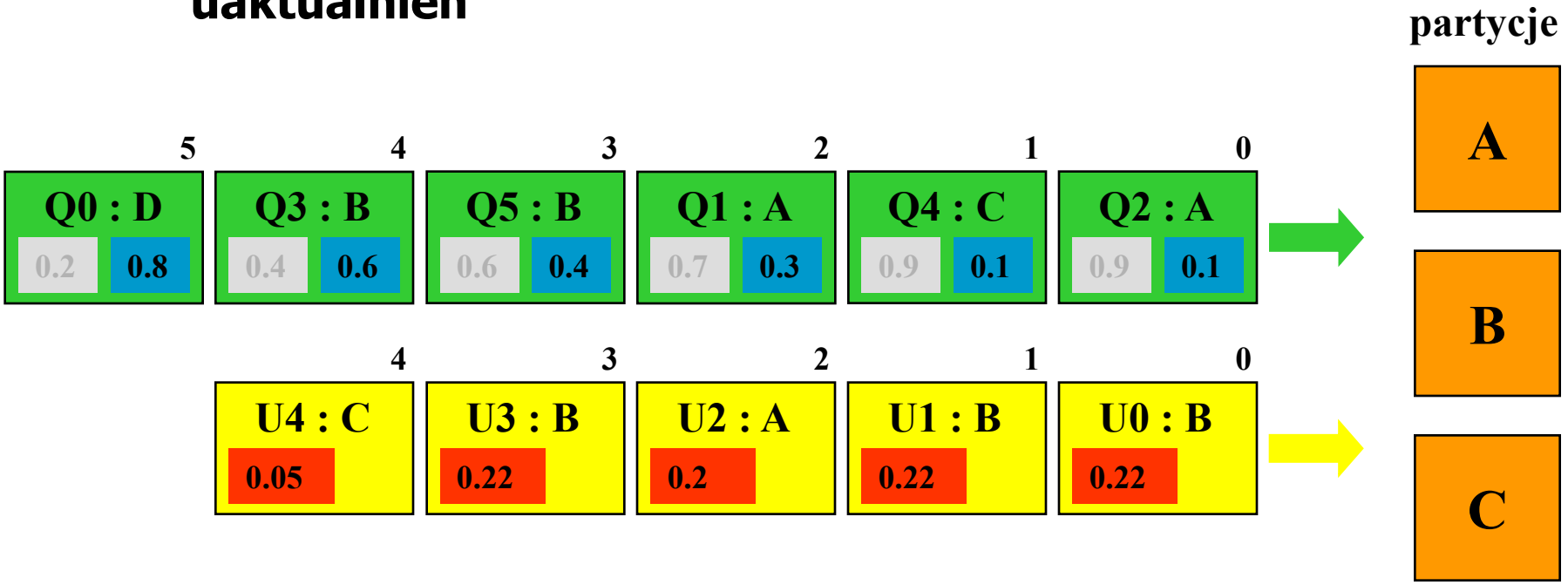
WINE (7)

- suma QoS=3,7
- suma QoD=2,3
- sortowanie KQ: order by QoS desc, timestamp asc



WINE (8)

- **Obliczanie współczynnika w dla zadań z kolejki uaktualnień**

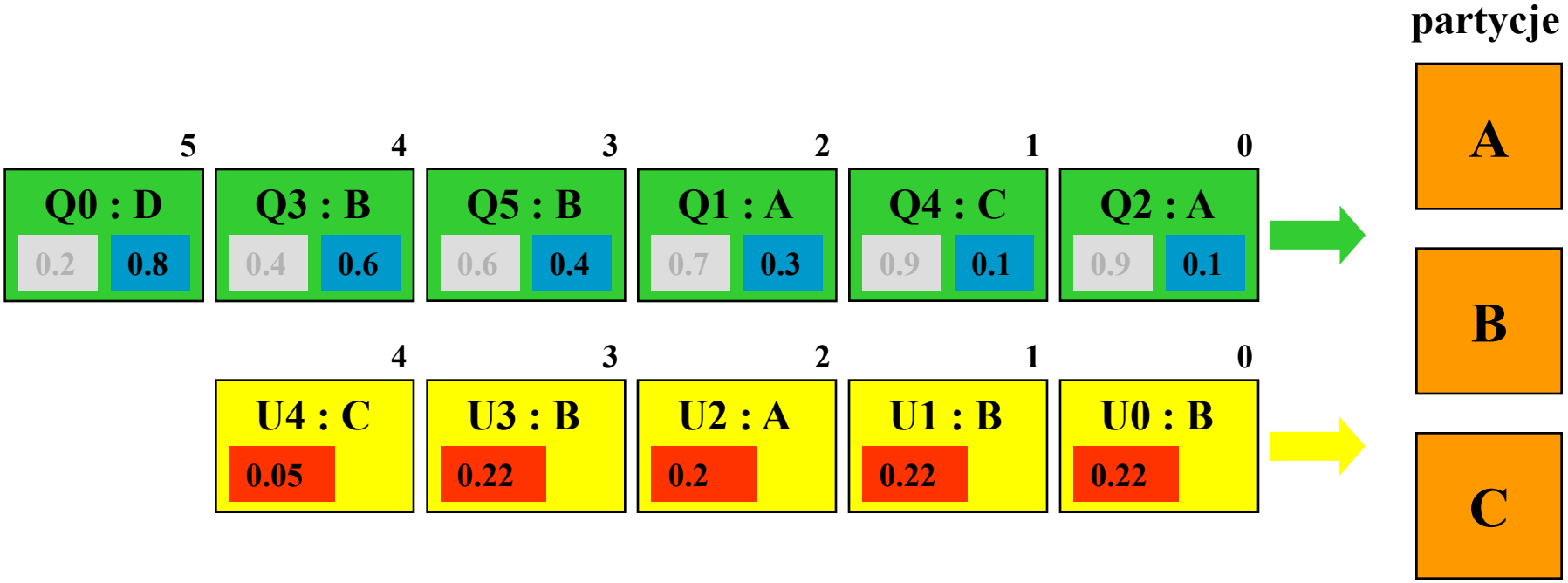


QoS QoD

$$w(U_1) = \frac{QoD_{Q_5}}{1 + pos_{Q_5}} + \frac{QoD_{Q_3}}{1 + pos_{Q_3}} = \frac{0.4}{1+3} + \frac{0.6}{1+4} = 0.22$$

WINE (9)

➲ **Sortowanie uaktualnień: order by w desc, timestamp asc**



QoS QoD