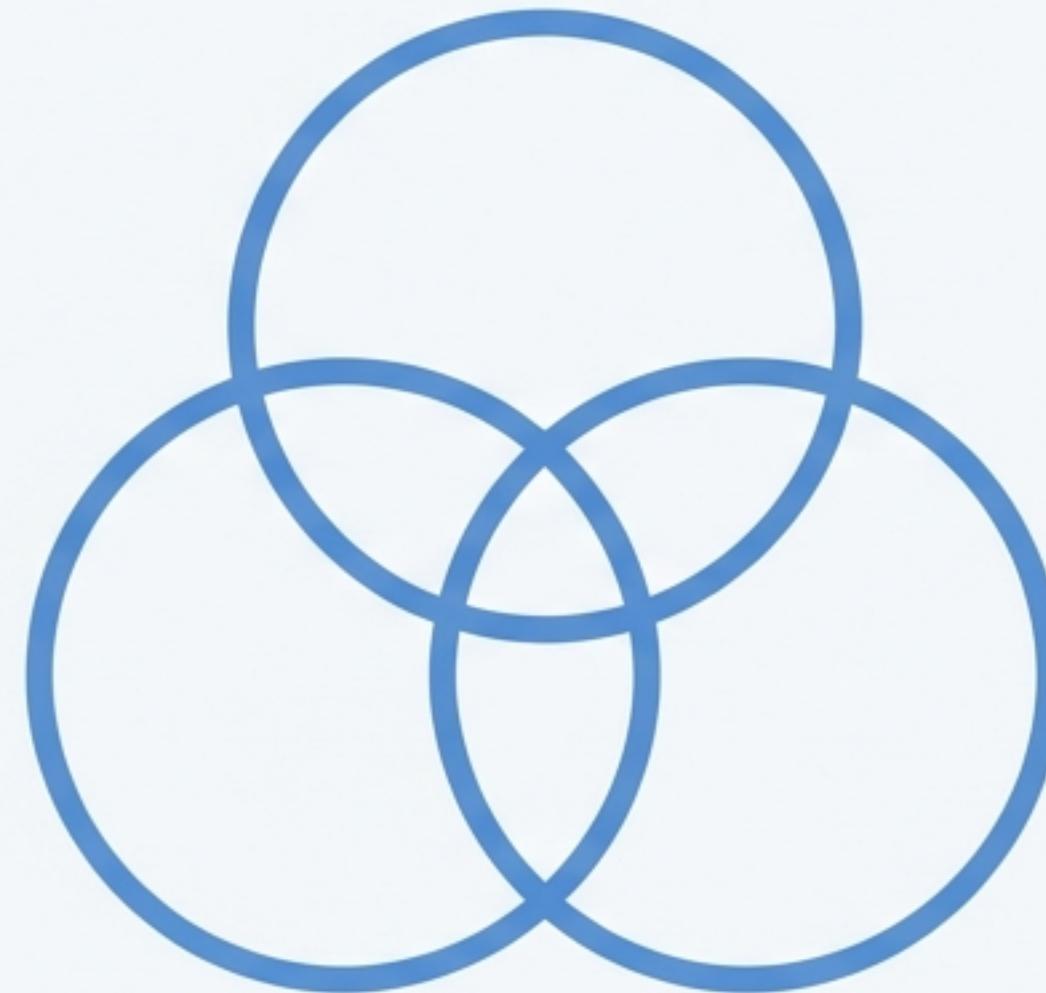


Twierdzenie CAP Dwanaście Lat Później

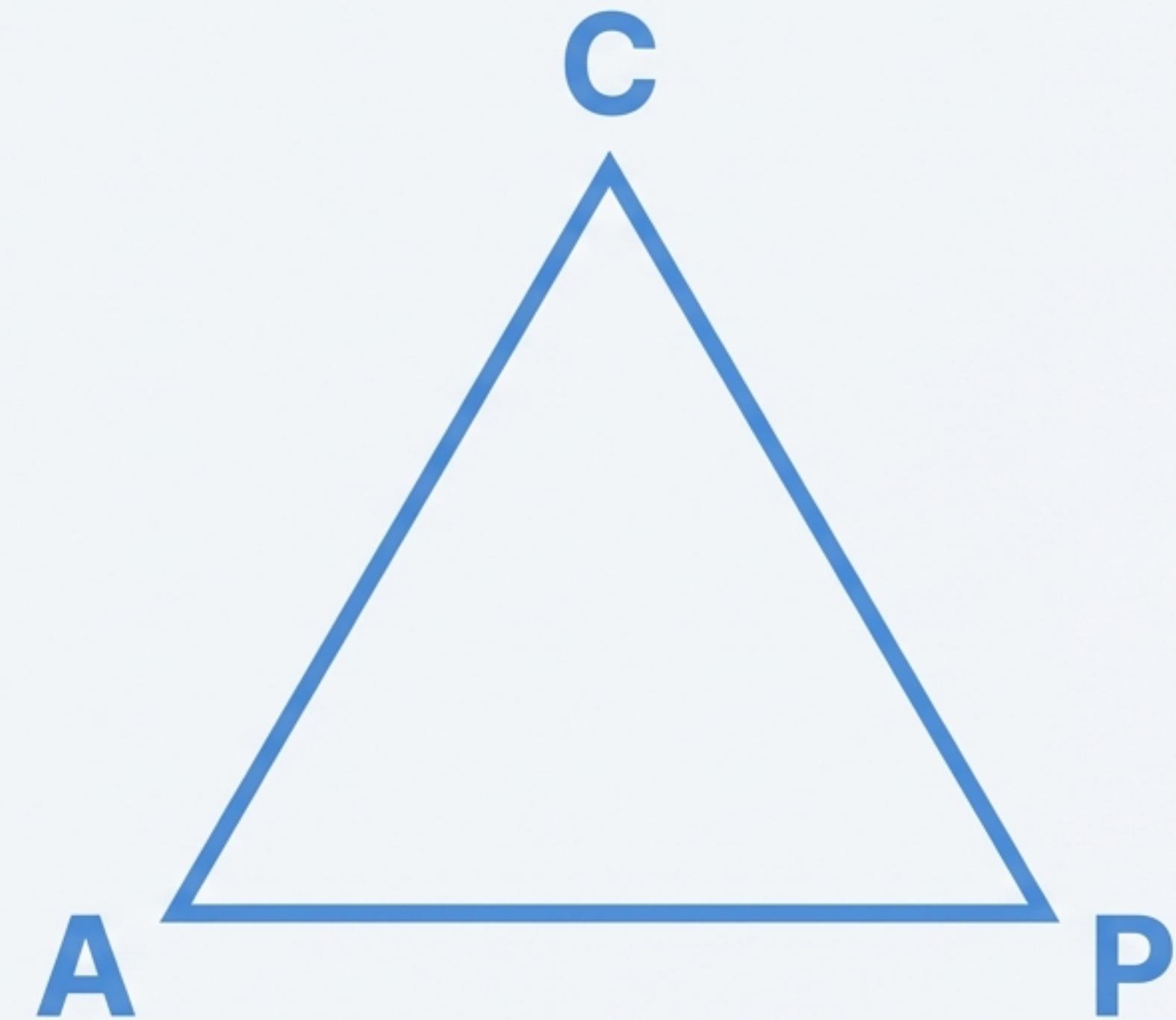
Jak Zmieniły Się 'Zasady' Projektowania Systemów Rozproszonych

Na podstawie eseju Erica Brewera

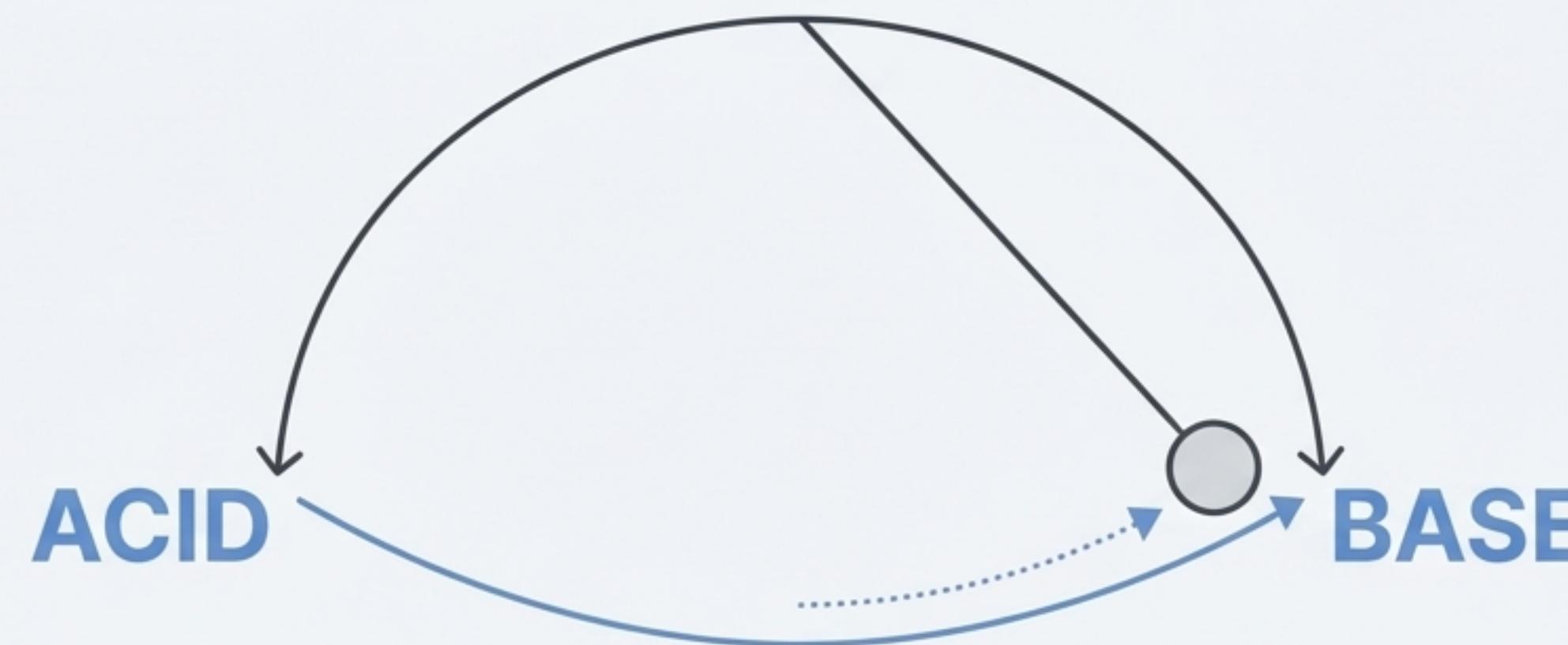


Fundamenty Twierdzenia CAP

- **C (Consistency / Spójność)**: Równoważne z posiadaniem jednej, zawsze aktualnej kopii danych.
- **A (Availability / Dostępność)**: Gwarancja, że system zawsze odpowiada na żądania (szczególnie dotyczące aktualizacji).
- **P (Partition Tolerance / Tolerancja na Podział Sieci)**: System kontynuuje działanie pomimo awarii komunikacji (podziału) między węzłami.
- **Klasyczna interpretacja**: 'Wybierz 2 z 3'. W rozproszonym systemie sieciowym (gdzie nie można zrezygnować z P), projektanci stają przed trudnym wyborem między spójnością (C) a dostępnością (A).
- **Wpływ**: Ten binarny wybór ukształtował architekturę systemów internetowych i stał się argumentem przeciwko tradycyjnym bazom danych.



Od ACID do BASE: Rewolucja NoSQL

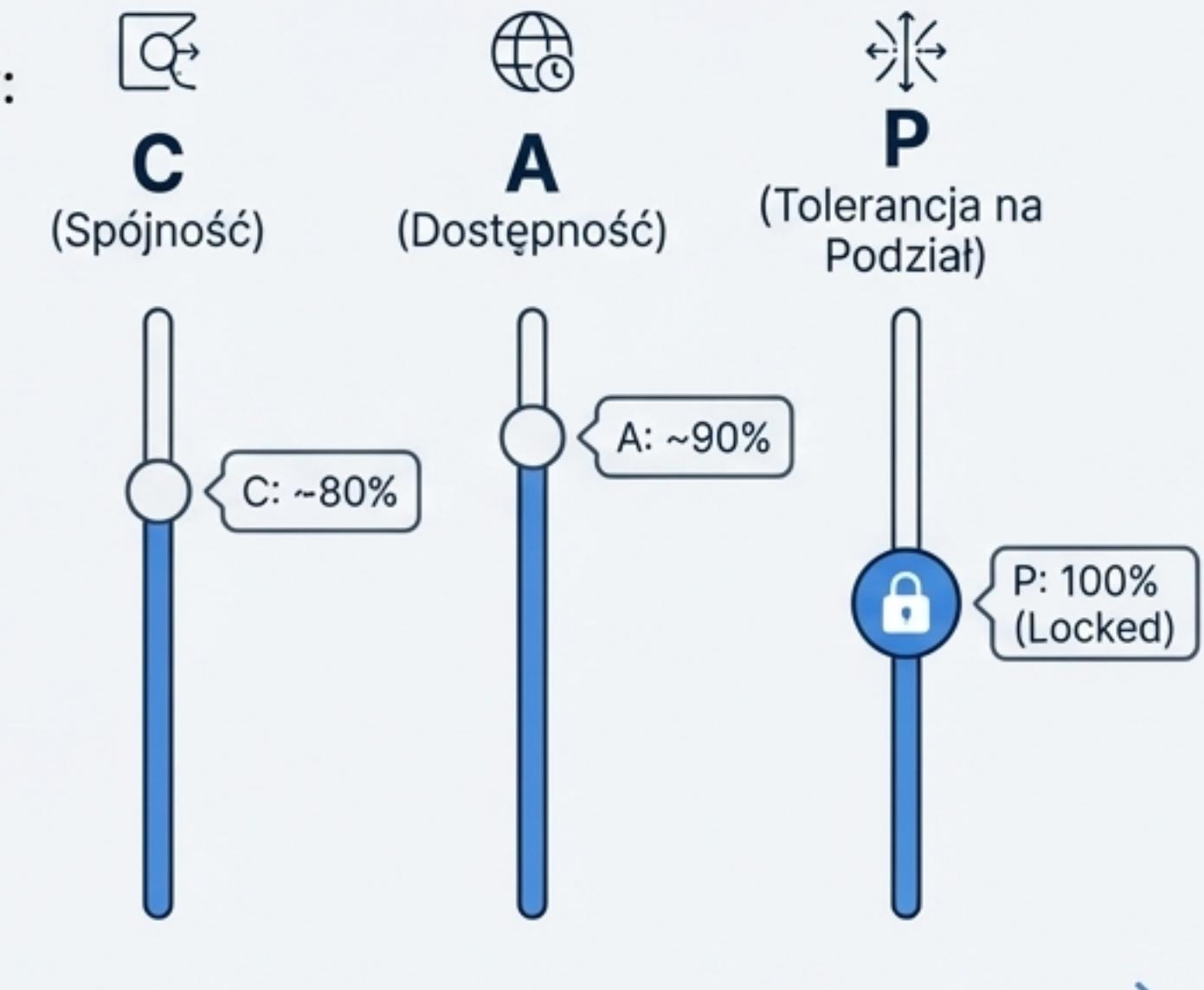


- **Świat ACID (tradycyjne bazy danych):**
Atomicity, Consistency, Isolation, Durability. Filozofia skupiona na rygorystycznej spójności.
- **Filozofia BASE (ruch NoSQL):**
Basically Available, Soft state, Eventually consistent. Mnemotechnika stworzona, by opisać nowe podejście stawiające dostępność na pierwszym miejscu.

- **Katalizator:** Twierdzenie CAP posłużyło do uzasadnienia potrzeby eksploracji szerszego spektrum systemów, faworyzujących dostępność kosztem spójności.
- **Problem:** Uproszczenie '2 z 3' było mylące.
Doprowadziło do nadmiernego skupienia się na wyborze, a nie na zarządzaniu kompromisami.

Wyjaśnienie Brewera (2012): CAP to nie jest wybór '2 z 3'

- **Kluczowy wniosek:** Ujęcie '2 z 3' jest mylące z kilku powodów:
 - Podziały sieci są rzadkie. W normalnych warunkach nie ma powodu, by rezygnować z C lub A.
 - Wszystkie trzy właściwości są bardziej spektrum niż wartościami binarnymi (0/1). Dostępność może wynosić od 0 do 100%, a spójność ma wiele poziomów.
 - Wybór między C a A może być dokonywany wielokrotnie w tym samym systemie, w zależności od operacji, danych czy nawet użytkownika.
- **Nowoczesny cel:** Zamiast dokonywać trwałego wyboru architektonicznego, należy maksymalizować kombinacje spójności i dostępności, które mają sens dla konkretnej aplikacji.



Od statycznego wyboru do dynamicznego zarządzania

Aktywne Zarządzanie Podziałem Sieci

- Nowoczesne podejście wymaga jawnnej strategii, która składa się z trzech kroków:

- 1. Wykryj podział:**

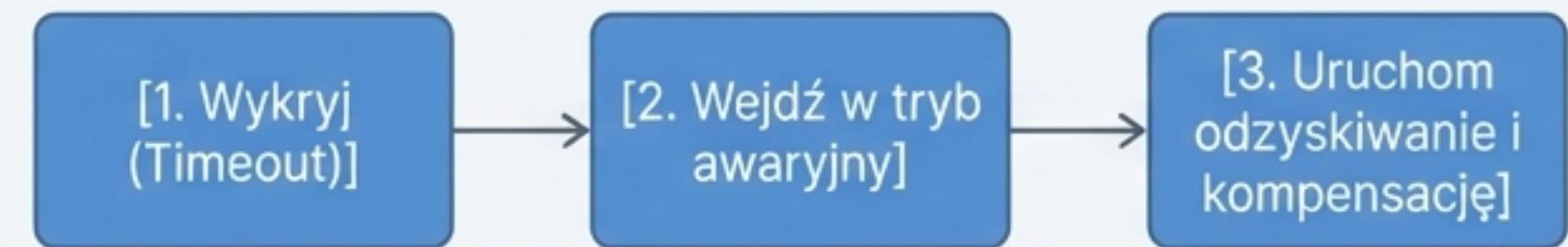
W praktyce, podział to przekroczenie limitu czasu (timeout) w komunikacji. Nie jest to globalny stan; różne węzły mogą go postrzegać inaczej.

- 2. Wejdź w tryb awaryjny (Partition Mode):**

System jawnie ogranicza niektóre operacje, aby chronić niezmienniki (faworyzując C) lub rejestruje dodatkowe informacje, aby umożliwić późniejszą naprawę (faworyzując A).

- 3. Uruchom odzyskiwanie (Recovery):**

Po przywróceniu komunikacji, system scala rozbieżne stany i kompensuje błędy popełnione w trakcie podziału.



Spójność Kontekstowa: Różne Strategie dla Różnych Danych

Nie trzeba wybierać jednej strategii dla całego systemu. Poziom spójności można dostosować do operacji.

Priorytet: Niski czas odpowiedzi (kosztem spójności)



- Przykład: *Yahoo! PNUTS* - dane użytkownika są replikowane asynchronicznie. Kopia główna (master) jest utrzymywana blisko 'domowej' lokalizacji użytkownika, co minimalizuje opóźnienia dla jego operacji.

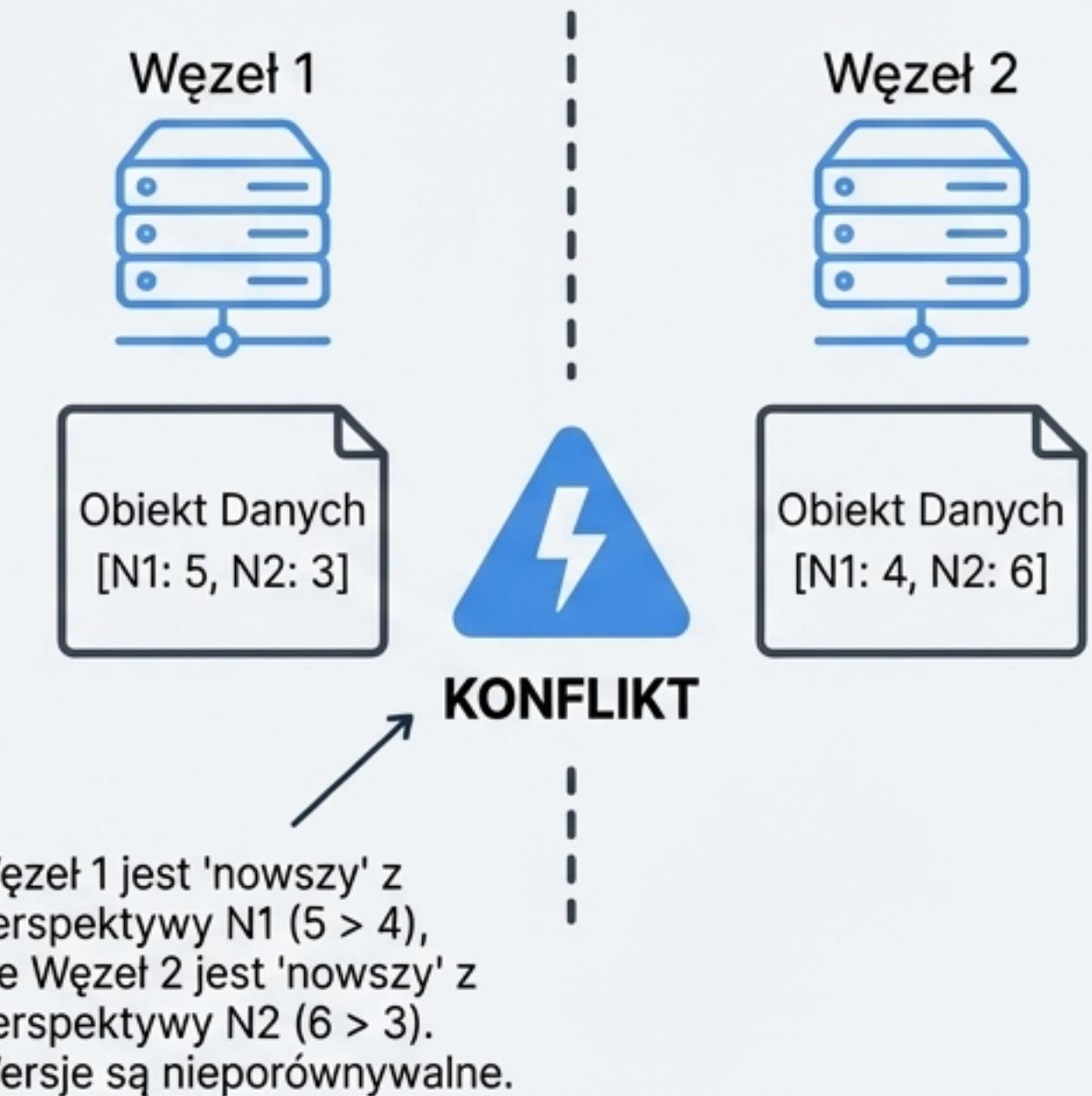
Priorytet: Spójność (kosztem czasu odpowiedzi)



- Przykład: *Facebook* - kopia główna jest zawsze w jednej lokalizacji. Gdy użytkownik z innej części świata dokonuje aktualizacji, jego żądanie jest kierowane do odległego mastera (wyższe opóźnienie), a jego odczyty przez krótki czas (20s) również trafiają do mastera, aby zapewnić, że widzi swoje zmiany.

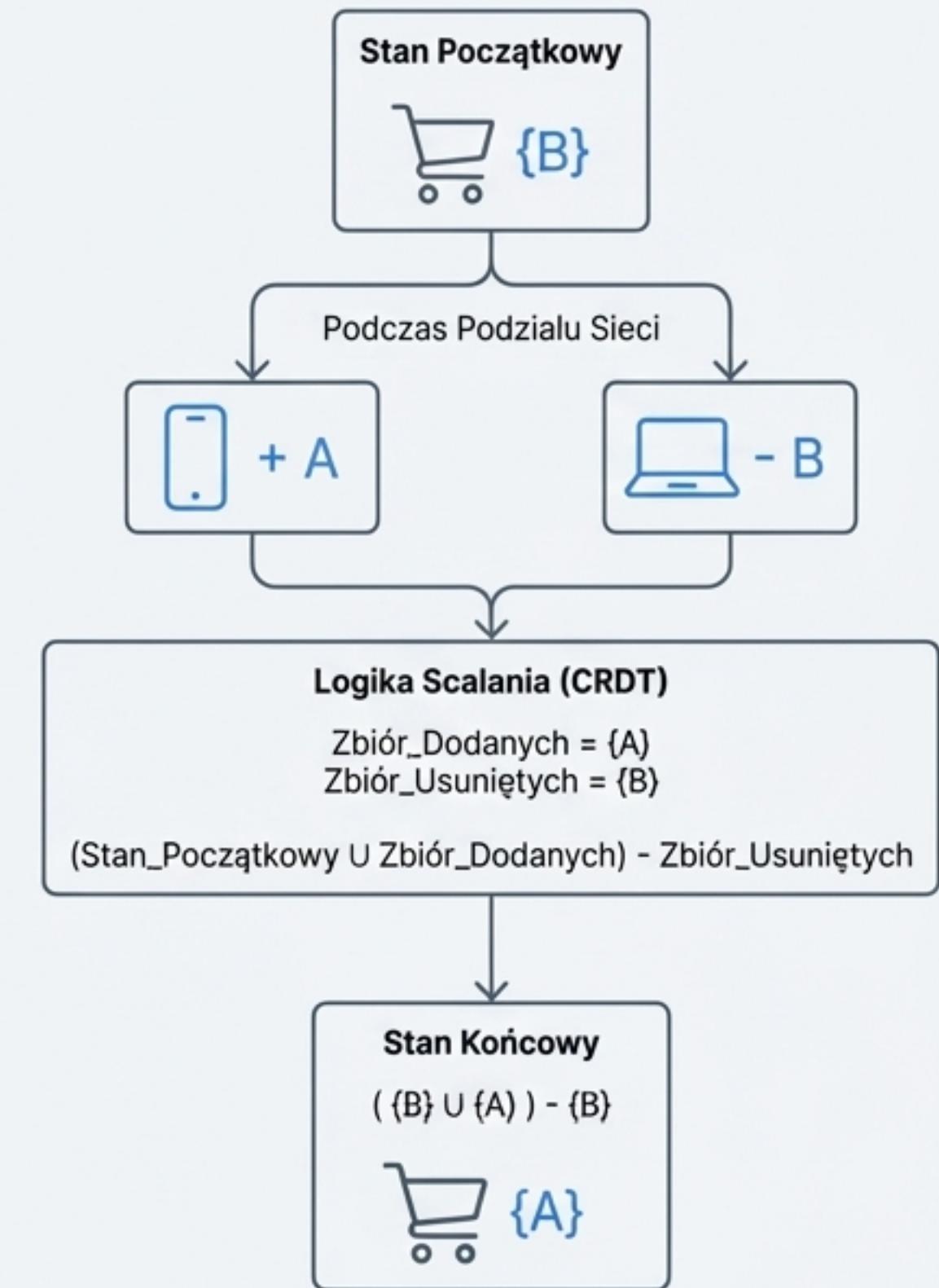
Narzędzie 1: Wektory Wersji do Wykrywania Konfliktów

- **Problem:** Jak po zakończeniu podziału system może stwierdzić, które operacje odbyły się równolegle i potencjalnie są w konflikcie?
- **Rozwiążanie: Wektory Wersji (Version Vectors)**
 - Przechwytują zależności przyczynowe między operacjami.
 - Każdy obiekt posiada wektor par '(węzeł, czas_logiczny)', który śledzi ostatnią aktualizację z każdego węzła.
- **Jak to działa:**
 - Po podziale, system porównuje wektory wersji z obu stron.
 - Jeśli nie da się ustalić, który wektor jest jednoznacznie 'nowszy' (tzn. obie strony mają wyższe wartości dla różnych węzłów), oznacza to, że aktualizacje były współbieżne i wystąpił konflikt, który wymaga rozwiązania.



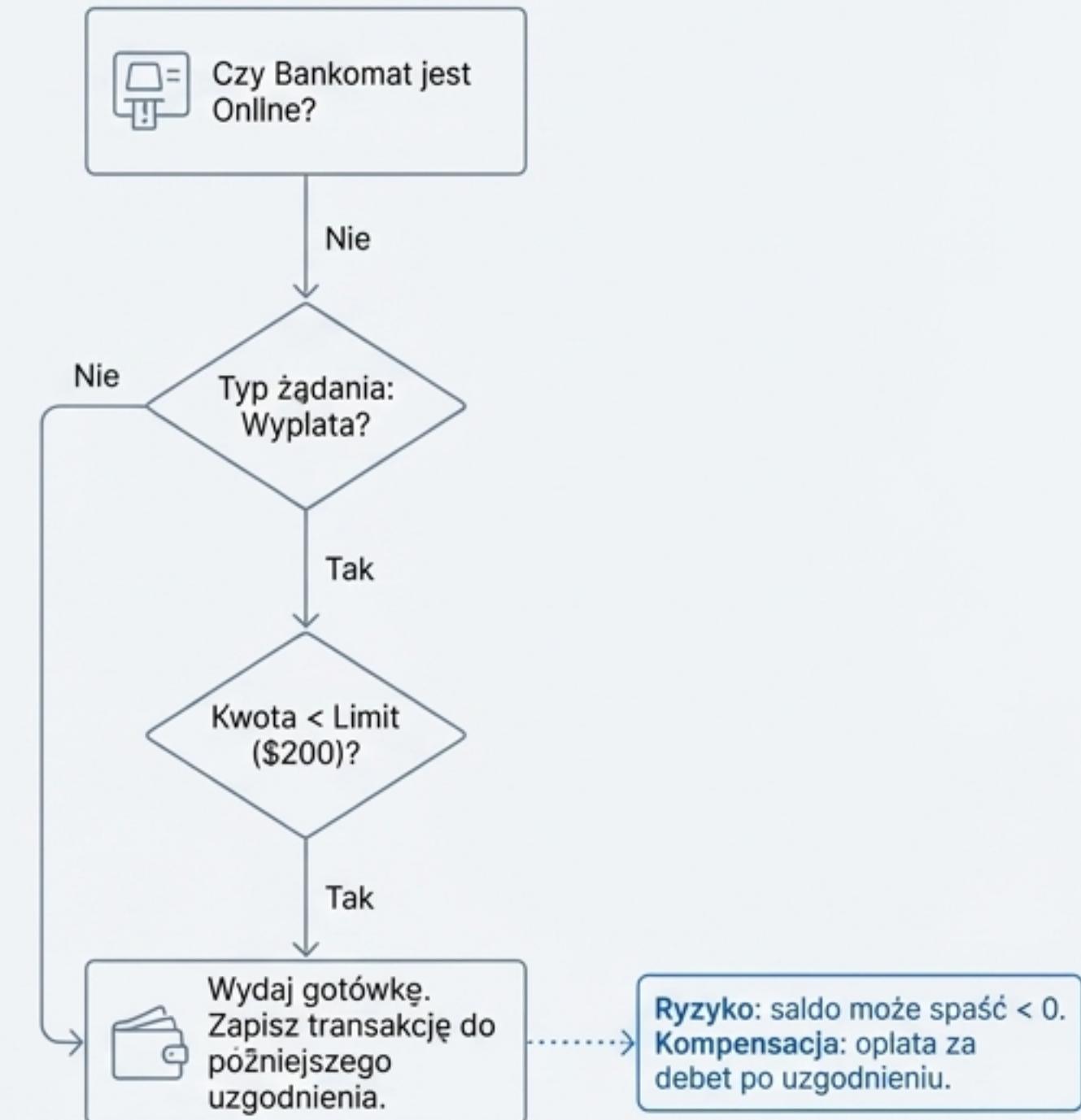
Narzędzie 2: CRDTs do Bezkonfliktowego Scalania

- **Cel:** Zaprojektowanie struktur danych, które z definicji można bezpiecznie scalać, unikając konfliktów.
- **Podejście:** Commutative Replicated Data Types (CRDTs).
- **Przykład: Koszyk w sklepie Amazon**
 - Tradycyjnie, jeśli użytkownik dodaje przedmiot A na telefonie i usuwa B na laptopie, scalenie jest problemem.
 - Strategia CRDT: Zamiast jednego zbioru, utrzymuj dwa: zbiór `dodanych` i zbiór `usuniętych` przedmiotów.
 - Oba zbiory tylko rosną (są monotoniczne), więc ich scalenie to prosta suma (union).
 - **Finalny koszyk = (suma wszystkich dodanych) - (suma wszystkich usuniętych).**
- Gwarancja: Żaden dodany produkt nie zostanie utracony, a usunięte mogą co najwyżej 'pojawić się' ponownie, co jest akceptowalnym kompromisem biznesowym.



Studium Przypadku: Logika Bankomatu (ATM)

- Bankomaty często działają w trybie offline (podziału sieci), ale muszą być dostępne.
- Wybór biznesowy: Dostępność (A) ponad spójnością (C), ponieważ dostępność generuje przychód.
- Zarządzanie ryzykiem w trybie awaryjnym (stand-in mode):
 - Operacje takie jak depozyt i sprawdzenie salda są zawsze dozwolone.
 - Wypłaty, które mogą naruszyć niezmiennik ($\text{saldo} \geq 0$), są dozwolone, ale ograniczone do pewnej kwoty (np. \$200). System świadomie ryzykuje powstanie debetu.
- Mechanizm kompensacji:
 - Po przywróceniu połączenia, system uzgadnia stan.
 - Jeśli wykryje, że saldo spadło poniżej zera, uruchamia procedurę kompensacyjną: nalicza opłatę za debet i oczekuje spłaty od klienta.



Niezmienniki Systemowe vs. Elastyczne Reguły

Kluczowym zadaniem projektanta jest sklasyfikowanie reguł biznesowych systemu. Wybór A (dostępność) wymaga jawnego zdefiniowania wszystkich niezmienników.



Niezmienniki (Invariants): Reguły, które NIGDY nie mogą być złamane.

- Są to fundamentalne prawdy biznesowe (np. saldo konta bankowego nie może być ujemne bez procedury debetowej).
- Operacje, które mogą je naruszyć w trakcie podziału, muszą być **blokowane, opóźniane lub modyfikowane**.

Elastyczne reguły: Mogą być tymczasowo naruszone.

- Są to reguły, dla których system potrafi naprawić niespójność (np. unikalność kluczy w tabeli, stan koszyka).
- Wymagają zaprojektowania jawnych scenariuszy **odzyskiwania i kompensacji**.

Podsumowanie: Nowoczesne Spojrzenie na CAP

- ✓ Mit 'wybierz 2 z 3' jest mylącym uproszczeniem. W normalnych warunkach systemy powinny zapewniać zarówno C, jak i A.
- ✓ Podział sieci to rzadki stan awaryjny, którym należy aktywnie zarządzać, a nie statyczny wybór architektoniczny, którego dokonuje się raz.
- ✓ Stosuj różne strategie spójności w zależności od krytyczności danych i operacji.
- ✓ Wykorzystuj narzędzia takie jak Wektory Wersji i CRDTs do zarządzania stanem i automatyzacji odzyskiwania spójności.
- ✓ Główny cel projektowy: **Zamiast poświęcać spójność lub dostępność, projektuj jawne strategie wykrywania podziału, działania w jego trakcie i odzyskiwania spójności po jego zakończeniu.**