Zaawansowane Systemy Baz Danych – ZSBD

Aktywne bazy danych

Wykład prowadzi:
Tomasz Koszlajda

Niniejszy wykład jest poświęcony aktywnym bazom danych. W przeciwieństwie do klasycznych – biernych baz danych, systemy zarządzania aktywnymi bazami danych mogą podejmować autonomiczne działania związane z procesami informacyjnymi użytkowników bazy danych. Mechanizmy aktywności baz danych pozwalają w prosty sposób rozszerzać funkcjonalność systemów zarządzania bazami danych.

Plan wykładu

- Model ECA
- Dziedziny zastosowania
- Schematy aktywności
- Zdarzenia elementarne
- Definiowanie aktywnych reguł
- Zdarzenia złożone
- Metodyki projektowania

Celem wykładu jest poznanie mechanizmów aktywności oferowanych przez współczesne systemy zarządzania bazami danych.

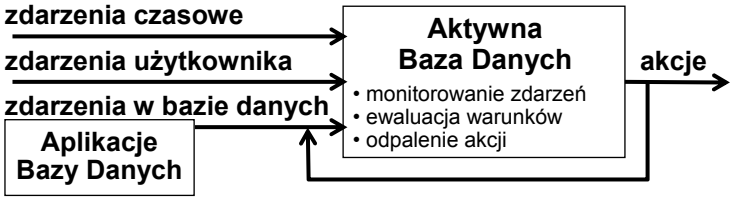
W ramach wykładu zostanie przedstawiony powszechnie akceptowany model aktywności ECA. Pokazane zostaną typowe dziedziny zastosowań dla aktywnych baz danych. Następnie poznamy podstawowe własności aktywnych baz danych: dostępne schematy aktywności, typy zdarzeń elementarnych oraz sposób definiowania zdarzeń złożonych. Definiowanie aktywnych reguł zostanie zilustrowane przykładami z różnych komercyjnych aktywnych systemów baz danych. Na koniec przedstawimy podstawowe metodyki projektowania zbiorów aktywnych reguł.

Funkcjonalność aktywnych bazy danych

Nowa funkcjonalność:

- monitorowanie zdarzeń,
- ewaluacja warunków logicznych,
- autonomiczne uruchamianie akcji,

umożliwia podejmowanie przez system bazy danych autonomicznej aktywności w obszarze zastrzeżonym dotąd dla aplikacji bazy danych.



Aktywne bazy danych (3)

W klasycznych, nieaktywnych bazach danych wszelkie działania wykonywane przez system bazy danych, a związane z realizacją procesów informacyjnych użytkownika są uaktywniane przez aplikacje bazy danych. Autonomiczne w stosunku do aplikacji użytkownika są jedynie procesy realizujące działania systemowe, na przykład takie jak obsługa logów, wykrywanie zakleszczeń, przydział i zwalnianie zasobów, itp.

Aktywne systemy baz danych potrafią same uruchamiać zadania związane z realizacją procesów informacyjnych użytkownika, w sposób niezależny od aplikacji bazy danych. Wymaga to rozszerzenia funkcjonalności klasycznych systemów baz danych o trzy dodatkowe funkcje: monitorowania przez system zarządzania bazą danych zdarzeń zachodzących w bazie danych, ewaluacji warunków przypisanych tym zdarzeniom oraz autonomicznego *„odpalania”* akcji, czyli uruchamiania kodu specjalnych procedur składowanych w bazie danych.

Na slajdzie zilustrowano ideę działania aktywnych baz danych. Pojawiające się w historii życia bazy danych i zdefiniowane przez jej użytkowników zdarzenia są przyczyną autonomicznego *odpalania* akcji. Odpalane akcje mogą generować zdarzenia, które będą przyczyną odpalenia kolejnych akcji.

Większość współczesnych systemów baz danych posiada funkcjonalność aktywnej bazy danych.

Model ECA

Model: Event (i) - **C**ondition (i, ii) - **A**ction (ii)

Definiowanie aktywnych reguł przez trzy elementy:

- wystąpienie zdarzenia
- weryfikacja warunku
- *odpalenie* akcji

Aktywne reguły są wykonywane w dwóch fazach:

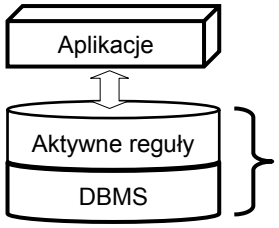
(i) - faza wystąpienie zdarzenia
(ii) - faza odpalenia akcji

Aktywna baza danych pozwala użytkownikom wskazywać zdarzenia, które mają być monitorowane przez system i kojarzyć z wystąpieniem tych zdarzeń określone akcje do uruchomienia. Definicje obejmujące typy zdarzeń, dodatkowe warunki logiczne i akcje są nazywane aktywnymi regułami, wyzwalaczami lub triggerami. Ogólnie przyjętym modelem definiowania aktywnych reguł jest tak zwany model ECA, który to akronim pochodzi od angielskiego Event, Condition, Action, czyli po polsku zdarzenie, warunek i akcja. Dodatkowo model ECA zakłada, że te trzy elementy mogą być wykonywane w dwóch fazach. Pierwsza faza jest związana z momentem wystąpienia zdarzenia, a druga z momentem odpalenia akcji. Weryfikacja warunków logicznych może być wykonywana w pierwszej lub drugiej fazie.

Dziedziny zastosowania

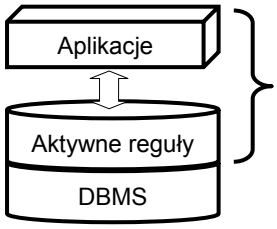
Systemowe – nowe funkcje DBMS

- Weryfikacja złożonych więzów integralności
- Utrzymywanie danych wywiedzionych
- Zarządzanie rozproszonymi bazy danych
- Zarządzanie przepływami pracy



Aplikacyjne – logika biznesowa w bazie danych

- Zarządzanie procesami przemysłowymi
- Systemy giełdowe
- Bazy wiedzy



Można wyróżnić dwie podstawowe klasy zastosowań aktywnych baz danych.

Pierwszą klasą zastosowań są rozwiązania systemowe rozszerzające uniwersalną funkcjonalność systemu zarządzania bazą danych. Zastosowanie aktywnych reguł pozwala wzbogacić i dopasować funkcjonalność systemu bazy danych do potrzeb danego wdrożenia. Przykładem takich zastosowań jest możliwość zdefiniowania nowych rodzajów więzów integralności o bardziej złożonej semantyce niż predefiniowane rozwiązania systemowe. Innym przykładem jest automatyczne utrzymywanie przez system spójności danych wywiedzionych. Spektakularnym polem dla zastosowania takiego mechanizmu są systemy klasy OLAP ze zmaterializowanymi agregatami danych zdefiniowanymi na bazie wielu danych elementarnych. Kolejnym przykładem są rozproszone bazy danych z mechanizmami fragmentacji i replikacji danych, które rzadko są w pełni zaimplementowane przez komercyjnie dostępne systemy zarządzania bazami danych. Ostatnim przykładem niech będą systemy zarządzania przepływami pracy, w których niezbędny jest automatyzm w przekazywaniu informacji i przepływu sterowania między rozproszonymi aplikacjami realizującymi elementarne zadania procesów informacyjnych.

Druga klasa zastosowań polega na przeniesieniu wyspecjalizowanej funkcjonalności systemów informatycznych z aplikacji bazy danych do systemu bazy danych. Naturalnym przykładem takiego zastosowania aktywnych reguł są systemy zarządzania procesami przemysłowymi. Aktywne reguły pozwalają zautomatyzować żmudne monitorowanie przez użytkowników wielu krytycznych parametrów zarządzanych procesów przemysłowych. Innym przykładem są systemy do obsługi giełdy, które są odpowiedzialne za monitorowanie zmian cen akcji i szybkie podejmowanie decyzji o ich sprzedaży lub zakupie. Ostatnim przykładem niech będą bazy wiedzy będące zbiorami złożonych reguł wnioskowania, które w pewnych sytuacjach musza być przeszacowywane.

Od kilku lat znacznym zainteresowaniem informatyków cieszą się tak zwane strumieniowe bazy danych. Są to wyspecjalizowane bazy danych, które mają służyć do sterowania procesami przemysłowymi. Jednym z podstawowych założeń strumieniowych baz danych jest wydajna obsługa tysięcy aktywnych reguł monitorujących strumienie informacji pochodzących z procesów i urządzeń przemysłowych.

Własności aktywnych baz danych

- **Modele aktywności** – zależności czasowe i przyczynowo-skutkowe między zdarzeniami i akcjami
- **Zdarzenia elementarne** – zbiór typów zdarzeń, które mogą być podstawą definiowania aktywnych reguł
- **Operatory zdarzeniowe** – zbiór operatorów umożliwiających specyfikację złożonych wyrażeń zdarzeniowych
- **Kontekst definicji aktywnych reguł** – pojedyncza dana, zbiór danych, baza danych

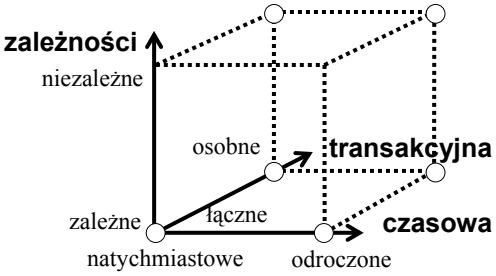
Ze względu na bardzo szeroką klasę zastosowań aktywnych baz danych niezbędna jest możliwość elastycznej konstrukcji aktywnych reguł dla ich dostosowania do zastosowań o różnej charakterystyce. Wyróżnia się cztery podstawowe własności aktywnych baz danych, które decydują o sile modelu poszczególnych systemów.

Pierwszą taką własnością są dostępne w aktywnej bazie danych schematy aktywności określające zależności zachodzące między zdarzaniem, a wyzwalaną przez nie akcją. Duża liczba dostępnych schematów aktywności przyczynia się do uniwersalności modelu danej aktywnej bazy danych. Drugą własnością jest zbiór typów zdarzeń elementarnych, które mogą być użyte do definicji aktywnych reguł. Większy zbiór typów zwiększa zakres zastosowań aktywnej bazy danych. Kolejną własnością jest zbiór dostępnych operatorów zdarzeniowych za pomocą których definiuje się zdarzenia złożone, na które może składać się wiele zdarzeń elementarnych. Ostatnią własnością jest kontekst definiowania aktywnych reguł. Kontekstem definicji może być pojedyncza dana, zbiór danych lub cała baza danych.

Schematy aktywności

Można wyróżnić trzy relacje zachodzące między fazami wystąpienia zdarzenia i uruchomienia akcji:

- Czasowa
- Transakcyjna
- Zależności



Można wyróżnić trzy relacje zachodzące między wystąpieniem zdarzenia zdefiniowanego w aktywnej regule, a odpalaną w związku z tym akcją. Są to relacje czasowa, transakcyjna i zależności.

Relacja czasowa określa, czy moment odpalenia akcji reguły pokrywa się z momentem wystąpienia zdarzenia, czy też akcja jest odroczona w czasie do zakończenia transakcji która uruchomiła zdarzenie odpalające akcję.

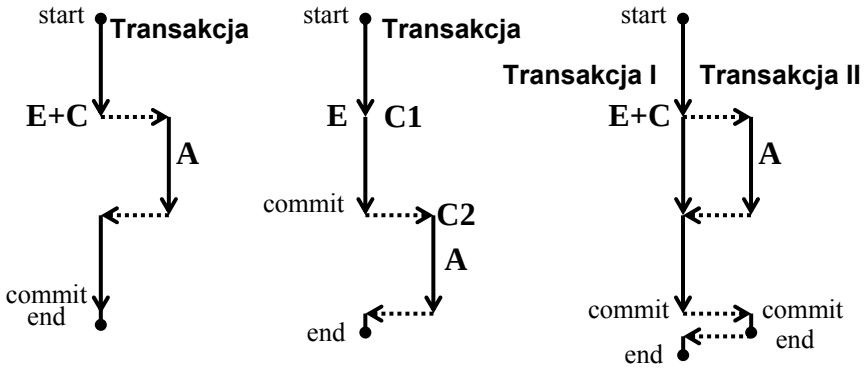
Relacja transakcyjna określa, czy procedura akcji jest uruchamiana jako część transakcji, w ramach której wystąpiło zdarzenie, czy jest osobną transakcją.

Relacja zależności określa, czy zatwierdzenie zakończenia akcji jest zależne od pomyślnego zakończenia transakcji w ramach której wystąpiło zdarzenie.

Te trzy relacje definiują trójwymiarową przestrzeń dostępnych schematów aktywności. Większość systemów komercyjnych wspiera jedynie najprostszy schemat aktywności obejmujący natychmiastowe odpalenie akcji, w ramach tej samej transakcji, która aktywowała zdarzenie. Jednak za pomocą dodatkowych mechanizmów systemowych można nieco większym nakładem pracy budować reguły o innych, bardziej złożonych modelach aktywności.

Przykłady schematów aktywności

a) Natychmiastowe, zależne i łączne
b) Odroczone, zależne i łączne
c) Natychmiastowe, zależne, osobne



Na slajdzie zobrazowano trzy przykładowe schematy aktywności.

W pierwszym przykładzie faza zdarzenia i faza akcji są natychmiastowe, zależne i łączne. Źródłem zdarzenia jest transakcja, która wykonała operację stanowiącą zdarzenie uaktywniające regułę. W momencie, w którym wystąpiło zdarzenie transakcja ta jest przerywana i odpalana jest akcja reguły. Akcja wykonuje się jako część transakcji, która była przyczyną jej uruchomienia. Efektem jest zależność wykonania akcji od tej transakcji, bowiem wycofanie transakcji wycofa również wszystkie odwracalne działania, które wykonała akcja. Po zakończeniu wszystkich działań składających się na akcję, sterowanie wraca do głównego wątku transakcji, która jest kontynuowana. Taki model aktywności może być wykorzystany na przykład do utrzymywania danych wywiedzionych lub weryfikacji więzów integralności.

Drugi przykład pokazuje model, w którym faza zdarzenia i faza akcji są odroczone, zależne i łączne. W tym wypadku, w momencie wystąpienia zdarzenia może być weryfikowana część warunków logicznych zdefiniowanych w aktywnej regule. Jednak wątek transakcji nie jest przerywany. Odpalenie akcji jest odroczone do osiągnięcia przez transakcję fazy akceptacji. W tym momencie, przed odpaleniem akcji dodatkowo może być weryfikowana druga część warunku logicznego. Po zakończeniu akcji transakcja jest zatwierdzana. Ten model aktywności jest właściwy do zarządzania procesami przemysłowymi. W tym wypadku bowiem pewne działania akcji wykonywane na zarządzanym procesie przemysłowym mogą być nieodwracalne, i w związku z tym powinny być przesunięte na koniec transakcji.

W trzecim przykładzie fazy zdarzenia i akcji są natychmiastowe, zależne i osobne. W momencie, w którym wystąpiło zdarzenie jest tworzona osobna transakcja, która jest przetwarzana równolegle do transakcji powodującej wystąpienie zdarzenia. Przedstawiany model jest zależny, bowiem zatwierdzenie akcji jest zależne od zatwierdzenia właściwej transakcji. Przykładem zastosowania tego modelu aktywności są sytuacje, w których działania akcji muszą być wykonywane na konto innego użytkownika bazy danych niż użytkownik głównej transakcji.

Zdarzenia elementarne

- **Zmiany stanu bazy danych**
  Wstawienie, usunięcie, modyfikacja, odczyt, dostęp
- **Zmiany schematu bazy danych**
  Tworzenie, modyfikacja, usunięcie
- **Zdarzenia w systemie bazy danych**
  LOGIN, LOGOFF, STARTUP, SHUDOWN, ERROR
- **Zdarzenia związane ze zmianami stanu transakcji**
  Rozpoczęcie, punkt akceptacji, zatwierdzenie, wycofanie
- **Zdarzenia temporalne**
  punkt czasu, upływ czasu, periodycznie
- **Zdarzenia zgłaszane przez użytkownika**

Funkcjonalność aktywnych reguł zależy od zbioru dostępnych typów zdarzeń elementarnych, które mogą odpalać akcję reguły. Najbardziej typową grupą zdarzeń są operacje wykonywane przez transakcje na stanie bazy danych. W większości komercyjnych systemów baz danych zbiór ten jest ograniczony do operacji modyfikacji: wstawienia nowych danych do bazy danych, zmiany wartości danych i usunięcia danych. Niektóre systemy prototypowe umożliwiają zastosowanie do definicji aktywnych reguł również operację odczytu danych.

Drugą grupą zdarzeń są operacje modyfikacji schematu bazy danych obejmujące tworzenie nowych struktur danych, ich modyfikacja lub usuwanie, nadawanie lub odbieranie uprawnień użytkownikom bazy danych, przyłączanie i odłączanie do zbiorów danych ich statystyk wykorzystywanych przy optymalizacji zapytań, itp. Przykładem systemu, w którym definicje reguł mogą korzystać z takiej kategorii zdarzeń jest system Oracle.

Kolejną grupą zdarzeń są operacje w systemie zarządzania bazą danych obejmujące przyłączanie się i odłączanie użytkowników od systemu bazy danych, pojawiające się w trakcie pracy błędy systemowe oraz operacje uruchamiania i zatrzymywania systemu. Zdarzenia takie mogą być wykorzystane do diagnostyki systemu bazy danych.

Następna grupa zdarzeń jest związana z przetwarzaniem transakcji i obejmuje operacje rozpoczynania, osiągania punktu akceptacji, zatwierdzania i wycofywania transakcji.

Przedostania grupa dotyczy zdarzeń temporalnych. Obejmuje ona zdarzenia polegające na wystąpieniu określonych punktów czasu, na przykład piątki godzina 17:15. Innym typem zdarzeń temporalnych jest względny upływ czasu od danego momentu, na przykład upływ pięciu minut i piętnastu sekund. Ostatnim typem zdarzenia należącego do tej grupy jest periodyczny upływ czasu, na przykład co dwie godziny.

Ostatnia grupa obejmuje pojedynczy typ zdarzenia, którym jest jawne zgłoszenie zdarzenia przez użytkownika, które może być przesłane z poziomu aplikacji bazy danych. Ostatnie trzy wymienione kategorie zdarzeń występują jedynie w systemach prototypowych.

Definiowanie aktywnych reguł Oracle

```
create trigger Zmodyfikuj_stan_osobowy
   after insert on Pracownicy
   for each row
   begin
      update Zespoły
      set stanOsob = stanOsob + 1
      where idZesp = :new.idZesp;
   end;
```

idPrac	Nazwisko	idZesp
100	Tarzan	10
110	Kowalski	10
120	Nowak	10

zdarzenie: insert

idZesp	stanOsob
10	2

fire

idZesp	stanOsob
10	3

Na slajdzie pokazano przykład aktywnej reguły zdefiniowanej w systemie bazy danych Oracle. Zadaniem reguły jest utrzymywanie wywiedzionego atrybutu „*stanOsobowy”* relacji „*Zespoły”* w wypadku dodawania do relacji „*Pracownicy”* krotek reprezentujących nowo przyjmowanych pracowników. Kompletny przykład powinien jeszcze obejmować przypadki zwalniania pracowników i przenoszenia ich między zespołami.

Zdarzeniem uaktywniającym regułę jest wystąpienie operacji INSERT na relacji „*Pracownicy”*. Słowo kluczowe AFTER użyte w drugiej linii definicji oznacza, że akcja reguły zostanie odpalona bezpośrednio po wykonaniu pojedynczej operacji wstawienia krotki. Fraza "for each row" występująca w trzeciej linii określa, że kontekstem akcji reguły jest pojedyncza krotka. Dla pojedynczego wywołania instrukcji INSERT, która wstawi 100 nowych krotek do relacji, akcja reguły zostanie odpalona stukrotnie, raz dla każdej wstawionej krotki. Akcja reguły jest zdefiniowana między słowami kluczowymi „*begin”* i „*end”*. Prefiks :New poprzedzający nazwę atrybutu „*IdZesp”* w ciele akcji reguły jest odwołaniem do wartości tego atrybutu we wstawianej krotce.

Na rysunku poniżej definicji przedstawiono przykładowy scenariusz odpalenia akcji zdefiniowanej reguły „*Zmodyfikuj_Stan_Osobowy”*. Do relacji „*Pracownicy”* zawierającej dwie krotki reprezentujące pracowników Tarzana i Kowalskiego dodano trzecią krotkę reprezentującą nowego pracownika Nowaka. Bezpośrednio po wstawieniu nowej krotki jest odpalana akcja reguły. W wyniku wykonania akcji stan osobowy zespołu dziesiątego jest zwiększany o jeden.

Definiowanie aktywnych reguł Oracle

```
create trigger Kontrola_płac
   after update of pensja on Pracownicy
   for each row
   when (new.pensja < 1000)
   begin
      raise_application_error(
            -20000,'Poniżej płacy minimalnej');
   end;
```

(0.9*1800 < 1000) → False

idPrac	Nazwisko	pensja
100	Tarzan	1800
110	Kowalski	2500

fire

idPrac	Nazwisko	pensja
100	Tarzan	1620
110	Kowalski	2500

⇧ zdarzenie:

```
update Pracownicy
  set pensja = 0.9 * pensja
  where nazwisko = 'Tarzan'
```


Na slajdzie przedstawiono kolejny przykład definicji aktywnej reguły w systemie Oracle, który zawiera warunek logiczny determinujący odpalenie akcji reguły. Zadaniem pokazanej reguły jest uniemożliwienie obniżenia płacy pracowników poniżej kwoty 1000 zł. Zdarzeniem uaktywniającym regułę jest modyfikacja atrybutu „*pensja”* relacji „*Pracownicy”*. Definicja reguły zawiera dodatkową klauzulę WHEN, w której zdefiniowano warunek konieczny dla odpalenia akcji reguły. Prefiks „*new”* użyty w warunku logicznym służy do odwołania się do wartości atrybutów modyfikowanych krotek. Zdefiniowana w ciele reguły akcja zawiera wywołanie funkcji zgłaszającej błąd i wycofującej modyfikację.

Poniżej definicji przedstawiono przykładowy scenariusz, w którym jest modyfikowana pensja pracownika o nazwisku Tarzan. W tym wypadku weryfikacja warunku skojarzonego z regułą jest negatywna. Pensja po modyfikacji pozostaje na poprawnym poziomie powyżej 1000 zł. W związku z tym, mimo wystąpienia zdarzenia odpalającego akcję, w tym wypadku akcja reguły nie zostanie odpalona.

Definiowanie aktywnych reguł SQLServer

```
create trigger liczba_prac ON PRACOWNICY
after delete as
   update zespoly
   set liczba_prac = liczba_prac - (
      select count(*) from deleted
      where deleted.id_zesp = zespoly.id_zesp)
   where id_zesp in (
      select distinct id_zesp from deleted)
```

idPrac	Nazwisko	idZesp
~~100~~	~~Tarzan~~	~~10~~
110	Kowalski	10

idZesp	stanOsob
10	2

fire

idZesp	stanOsob
10	1

zdarzenie:
```
delete from Pracownicy
   where nazwisko='Tarzan'
```

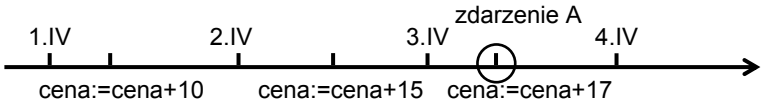

Kolejny przykład jest ilustracją sposobu definiowania aktywnych reguł w systemie SQLServer. Zadaniem aktywnej reguły przedstawionej na slajdzie jest utrzymywanie poprawnej wartości atrybutu „*StanOsobowy"* ze względu na operację usuwania krotek z relacji Pracownicy. W systemie SQLServer jedynym dostępnym kontekstem definiowania akcji reguł jest relacja. W porównaniu z systemem Oracle nie jest dostępny kontekst pojedynczych krotek. Dostęp do wartości modyfikowanych danych jest możliwy przez dwie robocze relacje: „*insertem"* i „*deleted"*, które są widoczne tylko i wyłącznie wewnątrz akcji aktywnych reguł. Schemat tych relacji jest taki sam jak schemat relacji, na której ma miejsce zdarzenie modyfikacji uaktywniające daną regułę. Relacja „*insertem"* przechowuje nowo wstawione krotki lub nowe wartości modyfikowanych krotek. Relacja „*deleted"* przechowuje usunięte krotki i stare wartości modyfikowanych krotek.

W pokazanym przykładzie akcja reguły odejmuje od starej wartości atrybutu „*StanOsobowy"* liczbę usuniętych krotek pracowników dla danego zespołu. Scenariusz obejmuje operację usunięcia z relacji „*Pracownicy"* krotki pracownika z zespołu 10. Uaktywniona przez to zdarzenie aktywna reguła zmniejsza składowaną liczbę pracowników tego zespołu.

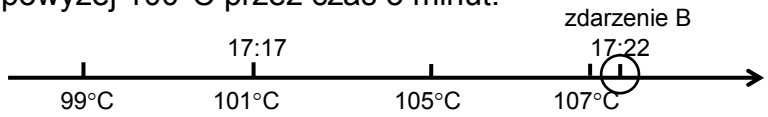
Zdarzenia złożone

Aktywne reguły wyzwalane zdarzeniami złożonymi:

- Zdarzenie A: trzy kolejne obniżki cen akcji giełdowych w czasie trzech dni



- Zdarzenie B: utrzymywanie się temperatury reaktora powyżej 100°C przez czas 5 minut.



Wszystkie pokazane dotąd aktywne reguły były wyzwalane pojedynczymi zdarzeniami elementarnymi. Tymczasem jest wiele zastosowań aktywnych baz danych, które wymagają definiowania aktywnych reguł na bazie zdarzeń złożonych. Na slajdzie pokazano dwa przykłady zdarzeń złożonych.

Pierwszy przykład dotyczy dziedziny notowań giełdowych. Aktywna reguła, której celem jest automatyczny zakup akcji po trzykrotnym spadku jej wartości mającym miejsce w czasie nie dłuższym niż trzy dni. Na dodatek spadki cen akcji nie mogą być przedzielone wzrostem cen. Zdarzenie to zostało zilustrowane na pierwszym wykresie. Punktem czasowym wystąpienia tego zdarzenia złożonego jest trzeci spadek ceny z dnia 3 kwietnia.

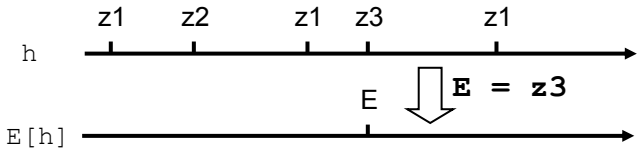
Drugi przykład odwołuje się do dziedziny zarządzania procesami przemysłowymi. Wyobraźmy sobie aktywną regułę, której celem jest awaryjne wyłączenie reaktora jeżeli jego temperatura przez czas dłuższy niż 5 minut będzie przekraczać 100°C. Przykład takiego zdarzenia pokazano na drugim wykresie. Odczyt temperatury reaktora z godziny 17:17 pokazał temperaturę przekraczającą wartość progową 100°C. Kolejne odczyty również pokazywały temperatury wyższe niż 100°C. Po pięciu minutach o godzinie 17:22 wystąpiło określone zdarzenie złożone.

Wyrażenia zdarzeniowe

Ewaluacja zdarzeń złożonych wymaga utrzymywania i analizy historii zdarzeń.

Historia zdarzeń jest skończonym zbiorem zdarzeń, w którym żadne dwa zdarzenia nie mogą mieć tego samego znacznika czasowego.

Wyrażenie zdarzeniowe *E* określone w historii zdarzeń *h* wyznacza podzbiór tych zdarzeń historii *h*, w których zdarzenie *E* jest spełnione.

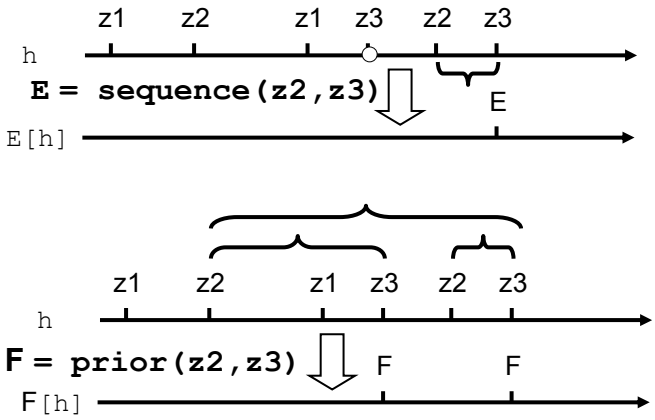


Jak pokazano na przykładach na pojedyncze zdarzenie złożone może się składać wiele wystąpień zdarzeń elementarnych. Wartościowanie zdefiniowanych zdarzeń złożonych wymaga utrzymywania i analizy całych historii zdarzeń elementarnych. Historia zdarzeń jest skończonym zbiorem zdarzeń, w którym każdemu zdarzeniu jest przypisany unikalny znacznik czasowy. Znaczniki czasowe jednoznacznie porządkują zbiór zdarzeń tworzących daną historię. W aktywnej bazie danych z każdą aktywną reguła jest skojarzona osobna historia pamiętająca zdarzenia elementarne użyte w definicji zdarzenia złożonego.

Zdarzenia złożone są definiowane przez wyrażenia zdarzeniowe konstruowane za pomocą zdarzeń elementarnych i operatorów zdarzeń złożonych. Wyrażenie zdarzeniowe wartościowane w ramach danej historii wyznacza podzbiór zdarzeń historii, w których wyrażenie to jest spełnione. Ilustruje to rysunek na dole slajdu. Na historię „*h”* składa się pięć wystąpień trzech różnych typów zdarzeń: „*z1”*, „*z2”* i „*z3”*. Zdefiniowane wyrażenie zdarzeniowe składa się z nazwy typu zdarzenia elementarnego „*z3”*. Zdarzenia elementarne są szczególnym przypadkiem zdarzeń złożonych. Tak zdefiniowane zdarzenie występuje w historii „*h”* tylko raz w momencie wystąpienia zdarzenia „*z3”*.

Operatory zdarzeniowe

Operatory następstwa: **sequence** i **prior**



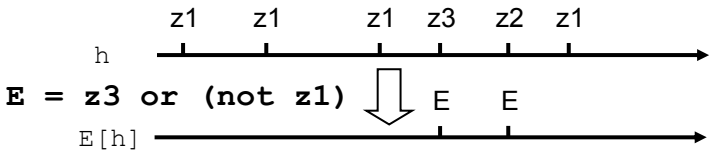
Na kolejnych slajdach zostaną pokazane wybrane operatory zdarzeniowe. Propozycja przedstawionego zbioru operatorów pochodzi z prototypowego systemu aktywnej bazy danych ODE (Object Database and Environment).

Dwa pokazane na slajdzie operatory umożliwiają opis własności następstwa dwóch zdarzeń w historii. Operator sekwencji – „*sequence"* opisuje następstwo polegające na bezpośrednim wystąpieniu jednego zdarzenia po drugim, w taki sposób, że nie są one przedzielone wystąpieniem żadnego innego zdarzenia. Wyrażenie zdarzeniowe pokazane na rysunku opisuje sekwencję wystąpień dwóch zdarzeń elementarnych typu „*z2"* i „*z3"*. Zdarzenie złożone E zdefiniowane za pomocą tego wyrażenia nie zajdzie dla pierwszego wystąpienia zdarzenia „z3" – co symbolizuje czerwona kropka, ponieważ wystąpienie to jest oddzielone od wystąpienia zdarzenia „z2", zdarzeniem „z1". Dopiero drugie wystąpienie zdarzenia „z3" jest wystąpieniem zdarzenia złożonego E.

Drugi operator następstwa – „*prior"* opisuje następstwo dwóch zdarzeń, które mogą być przedzielone dowolną liczbą innych zdarzeń. Zdefiniowane w drugim przykładzie wyrażenie zdarzeniowe opisuje następstwo typu „*prior"* dla zdarzeń typu „*z2"* i „*z3"*. Zdarzenie złożone *F* opisane tym wyrażeniem wystąpi dwukrotnie w danej historii h, w punktach wystąpienia zdarzenia „*z3"*. Zwróćmy uwagę, że drugie wystąpienie zdarzenia *F* miałoby miejsce również w sytuacji, gdyby usunąć z historii „*h"* drugie wystąpienie zdarzenia „*z2"*. Zdarzenie „*z3"* występuje po pierwszym wystąpieniu zdarzenia „*z2"* dwukrotnie. Każde z tych wystąpień jest wystąpieniem zdarzenia złożonego F. Symbolizuje to duża błękitna klamra nad rysunkiem.

Operatory zdarzeniowe

Operatory logiczne: **or**, **and** i **not**

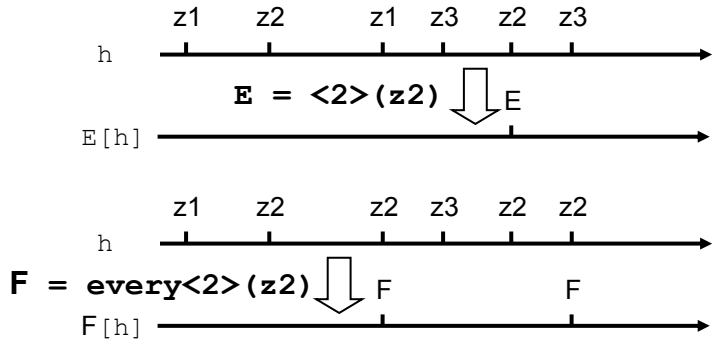


Kolejną grupą operatorów zdarzeniowych są operatory logiczne: sumy, iloczynu i negacji zdarzeń. Suma logiczna dwóch typów zdarzeń obejmuje wszystkie wystąpienia zdarzenia pierwszego lub drugiego typu. Iloczyn logiczny dwóch typów zdarzeń występuje w punktach czasowych, w których jednocześnie występują zdarzenia obydwu typów. Z definicji historii zdarzeń elementarnych wynika, że iloczyn logiczny dwóch różnych typów zdarzeń elementarnych jest dla dowolnej historii pusty. W użytecznych zastosowaniach argumentami operatora iloczynu zdarzeń muszą być zdarzenia złożone, których nie dotyczy ograniczenie unikalności znaczników czasowych. Z kolei negacją zdarzenia danego typu są wystąpienia wszystkich innych typów zdarzeń.

Na slajdzie pokazano przykład wyrażenia, które zwiera operatory sumy i negacji. Zdarzenie E będzie występowało w punktach czasowych, w których występuje zdarzenie „*z3”* lub dowolne inne zdarzenie różne od zdarzenia „*z1”*. W pokazanym konkretnym przypadku wartościami tego wyrażenia będą wystąpienia zdarzeń „*z2”* i „*z3”*.

Operatory zdarzeniowe

Operatory wyliczeniowe: <n>E, every<n>E

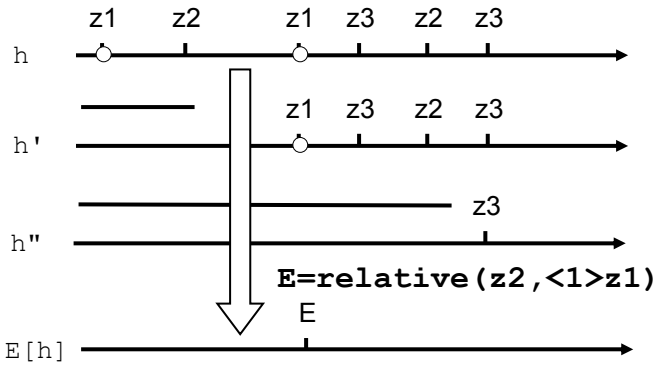


Następna grupa operatorów to operatory wyliczeniowe. Pierwszy z operatorów wyliczeniowych wyznacza dla danego typu zdarzenia „*t*” i danej liczby naturalnej „*n*”, „*n*”-te wystąpienie zdarzenia typu „*t*”. Działanie tego operatora ilustruje pierwszy rysunek. Zdarzenie złożone *E* jest opisane przez wyrażenie wyznaczające drugie wystąpienie zdarzenia typu „*z2*”. W dowolnie długiej historii może wystąpić co najwyżej jedno takie zdarzenie.

Drugi operator jest cyklicznym operatorem wyliczeniowym, który dla danego typu zdarzenia „*t*” i danej liczby naturalnej „*n*” wyznacza każde „*n*”-te wystąpienie zdarzenia typu „*t*”. W drugim przykładzie jest zdefiniowane zdarzenie złożone *F*, jako co drugie wystąpienie zdarzenia typu „*z2*”.

Operatory zdarzeniowe

Operator następstwa: **relative**

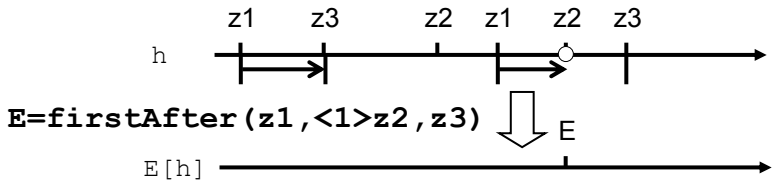


Następnym rozpatrywanym operatorem jest operator względnego następstwa zdarzeń, czyli operator „*relative*”. Operator ten ma odmienną semantykę od poznanych wcześniej operatorów następstwa „*sequence*” i „prior”. Operator „*relative*” jest takim następstwem dwóch zdarzeń „*z1*” i „*z2*”, w którym wystąpienie zdarzenia typu „*z2*” musi wystąpić w historii zdarzeń *h'* uzyskanej ze źródłowej historii „*h*”, przez usunięcie z niej wystąpienia zdarzenia „*z1*” i wszystkich zdarzeń je poprzedzających.

Na rysunku pokazano przykład definicji zdarzenia złożonego E, które jest pierwszym wystąpieniem zdarzenia „*z1*” w historii po wystąpieniu zdarzenia „*z2*”. W pierwszej linii na niebiesko zaznaczono pierwsze wystąpienie zdarzenia „*z1*” w całej historii zdarzeń „*h*”. Druga linia pokazuje historię *h'* uzyskaną przez usunięcie z historii „*h*” pierwszego wystąpienia zdarzenia typu „*z2*” i wszystkich zdarzeń je poprzedzających. Tym razem, niebieska kropka pokazuje pierwsze wystąpienia zdarzenia „*z1*” w tej pod-historii. Punkt ten jest jedynym wystąpieniem zdarzenia złożonego E w historii „*h*”, bowiem w historii *h”* utworzonej przez usunięcie z niej drugiego wystąpienia zdarzenia „*z2*” i wszystkich zdarzeń je poprzedzających, zdarzenie „*z1*” i w konsekwencji zdarzenie E już nie występują.

Operatory zdarzeniowe

Operator okna: **firstAfter**



Ostatnim prezentowanym przykładem operatora zdarzeniowego jest operator okna „*firstAfter”*. Argumentami tego operatora są trzy typy zdarzeń. Pierwszy i trzeci argument definiują okno, w którym oczekiwane jest zdarzenie będące drugim argumentem operatora. Na slajdzie za pomocą operatora „*firstAfter”* jest zdefiniowane zdarzenie złożone E. Okno określają wystąpienia zdarzeń „*z1”* i „*z3”*. Zdarzenie E będzie zachodzić w pierwszym wystąpieniu zdarzeniu „*z2”*, które mieści się w takim oknie. W podanym scenariuszu, pierwsze okno ograniczone przez zdarzenia „*z1”* i „*z3”* jest puste. W drugim oknie znajduje się wystąpienie zdarzenia „*z2”*. Moment jego wystąpienia jest wystąpieniem zdarzenia E.

Przedstawione operatory zdarzeniowe nie są jedynymi zdefiniowanymi w aktywnej bazie danych ODE. System ten zawiera jeszcze operatory rekurencyjne, potokowe i inne odmiany operatora okien. W komercyjnych systemach baz danych zbiór dostępnych operatorów zdarzeniowych jest ograniczony do operatora sumy logicznej.

Aktywna reguła ze zdarzeniem złożonym

Przełącz układ sieci jeżeli przepustowość przez 5 minut jest mniejsza od wartości progowej równej 10

```
#define PoniżejProgu AFTER UPDATE && przepustowość<10
#define PowyżejProgu AFTER UPDATE && przepustowość>10
class Łącze {
private:
      float przepustowość;
public:
      boolean Przełącz();
trigger:
      SpadekPrzepustowości():separate dependent
      firstAtfter( relative(
            ZmianaPowyżejProgu,
                  <1>(PoniżejProgu)),
            AFTER TIME (M=5), PowyżejProgu)
      ==> Przełącz ();};
```


Na slajdzie pokazano przykład kompletnej definicji aktywnej reguły w systemie ODE zawierającej specyfikację zdarzenia złożonego. Definicja wykorzystuje poznane na wykładzie operatory zdarzeniowe. Zadaniem aktywnej reguły jest przełączenie łącza sieci komputerowej w wypadku gdy przepustowość łącza przez czas pięciu minut będzie niższa niż minimalna wartość progowa równa 10.

System bazy danych ODE ma model obiektowy. Definicja aktywnych reguł jest częścią definicji klas. Przykład zawiera definicję klasy „*Łącze*”. Stan atrybutu przepustowość odzwierciedla aktualną przepustowość danego łącza. Nie pokazana na przykładzie metoda klasy „*Łącze*” monitoruje faktyczny stan łącza i odpowiednio modyfikuje wartości atrybutu „*przepustowość*”. Dwie makrodefinicje opisują dwa zdarzenia elementarne modyfikacji stanu obiektów klasy „*Łącze*”. Pierwsze zdarzenie dotyczy modyfikacji, po której stan atrybutu „*przepustowość*” jest mniejszy niż 10, a drugie modyfikacji po której wartość tego atrybutu jest większa niż 10. Operator && służy do definicji warunku reguły. Publiczna metoda „*Przełącz*” zawiera kod akcji reguły.

Schemat aktywności zdefiniowanej reguły o nazwie „*SpadekPrzepustowości*” jest określony przez słowa kluczowe „*separate”* i „*dependent”*. Słowo „*separate”* oznacza, że akcja reguły jest odpalana jako osobna transakcja. Natomiast słowo kluczowe „*dependent*” oznacza, że zatwierdzenie działania akcji jest zależne od zatwierdzenia transakcji modyfikującej atrybut „*przepustowość”*. Do definicji zdarzenia złożonego wykorzystano operator okna „*firstAfter”*. Okno jest ograniczone przez zdarzenia spadku przepustowości poniżej wartości progowej i następującego po nim powrotu powyżej wartości progowej. Zdarzeniem wyszukiwanym w oknie jest upływ 5 minut od otwarcia okna, który odpowiada celowi działania aktywnej reguły.

Metodyka projektowania aktywnych reguł

Projekt dużego zbioru potencjalnie interferujących ze sobą reguł wymaga spełnienia pewnych wymagań:

- **własność stopu** – przetwarzanie akcji reguł uaktywnionych przez pojedyncze zdarzenie zostanie zakończone w skończonym czasie
- **determinizm stanu** – kolejność wykonania akcji reguł uaktywnionych tym samym zdarzeniem nie ma wpływu na końcowy stan bazy danych

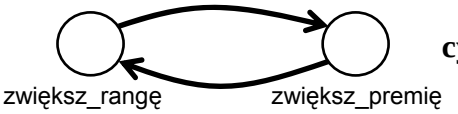
Aktywne reguły są autonomicznymi jednostkami programowymi. Poprawne działanie pojedynczej reguły, nie gwarantuje poprawności działania całego zbioru niezależnie zaprojektowanych i zbudowanych reguł. Aktywne reguły mogą wchodzić we wzajemne interakcje, na przykład gdy dwie różne reguły są uaktywniane tym samym zdarzeniem lub gdy operacje wykonywane w ramach akcji jednej z reguł są zdarzeniami uaktywniającymi inne reguły. Popularną metodyką służąca do zaprojektowania dużego zbioru reguł jest modularyzacja polegająca na grupowaniu reguł w autonomiczne warstwy, na przykład tematyczne. Reguły znajdujące się w różnych warstwach nie powinny wchodzić we wzajemne interakcje. Natomiast problem potencjalnych interakcji reguł znajdujących się w tej samej warstwie powinien być rozwiązany przez wspólny projekt wszystkich reguł z danej warstwy.

Zbiór potencjalnie powiązanych reguł powinien spełniać dwa podstawowe wymagania. Po pierwsze taki zbiór powinien posiadać własność stopu, która wymaga by przetwarzanie akcji reguł wyzwolonych przez zdarzenie zostało zakończone w skończonym czasie. Spełnienie tego warunku dla zbioru autonomicznie projektowanych aktywnych reguł potencjalnie nie powiązanych tematycznie jest trudniejsze niż dla zintegrowanych programów. Drugą wymaganą własnością, którą musi spełniać zbiór aktywnych reguł jest determinizm stanu. Własność ta polega na tym, by kolejność odpalania akcji reguł uaktywnianych tym samym zdarzeniem nie miała wpływu na końcowy stan bazy danych. Trudność w spełnieniu tego wymagania wynika z faktu, że w komercyjnych systemach aktywnych baz danych nie ma mechanizmu do programowego określania kolejności uruchamiania reguł uaktywnionych tym samym zdarzeniem. Oznacza to, że system zarządzania bazą danych odpala akcje takich reguł w arbitralnie ustalonej kolejności.

Własność stopu zbioru reguł

```
create trigger zwiększ_rangę
   after update of wysokość on Premie
   for each row
   when new.wysokość – old.wysokość > 400
   begin
      update Pracownicy set ranga = ranga+1
      where nr = :new.nr_prac;
   end
```

```
create trigger zwiększ_premię
   after update of ranga on Pracownicy
   for each row
   begin
      update Premie set wysokość=wysokość+500*:new.ranga
      where nr_prac = :new.nr;
   end
```



cykl

Na slajdzie pokazano przykład zbioru dwóch reguł, który nie posiada własności stopu. Przedstawione reguły obsługują bazę danych pracowników. Pozycja każdego pracownika jest określona przez jego rangę. Im większa ranga danego pracownika tym większe premie on otrzymuje. Awans o jedną rangę oznacza podwyżkę premii o 500 zł. Ta zależność jest utrzymywana automatycznie przez regułę „*zwiększ_premię”*. Premie mogą być podnoszone również w wyniku innych zasług pracownika. Podniesienie premii o ponad 400 zł skutkuje podwyższeniem rangi pracownika. Ta zależność jest utrzymywana przez regułę o nazwie „*zwiększ_rangę”*.

Pojawienie się z zewnątrz akcji reguł zdarzenia zwiększenia rangi pracownika pociąga za sobą cykliczne wywołania dwóch przedstawionych reguł. Podniesie rangi pociąga za sobą podniesienie premii, które z kolei jest przyczyną podniesienia rangi, itd. Podany zbiór reguł nie posiada więc własności stopu.

Własność stopu

```
create trigger kotwica_budżetowa
   after insert or update of budżet on Zespoły
   declare
         suma float;
   begin
         select sum(budżet) into suma from zespoły
         if suma > 100 then
               update Zespoły
               set budżet = 0.9*budżet;
         end if;
   end
```

idZesp	budżet
10	50
20	50

insert

30	20

fire

idZesp	budżet
10	45
20	45
30	18

fire

idZesp	budżet
10	40,5
20	40,5
30	16,2

stop

Wzajemne wywoływanie się reguł nie musi wiązać się z brakiem własności stopu. Po kilku iteracjach taki cykl może zostać zatrzymany. Przypadek taki pokazano na przykładzie pojedynczej reguły, której akcja obejmuje operacje, które są zdarzeniem rekurencyjnie odpalającym akcję reguły.

Zadaniem przedstawionej reguły jest uniemożliwienie przekroczenia zadanego górnego pułapu budżetu firmy ustalonego na 100 jednostek. Modyfikacje budżetu poszczególnych zespołów firmy są weryfikowane przez regułę „kotwica_budżetowa". W wypadku przekroczenia zadanego progu budżetu całej firmy, budżety poszczególnych zespołów są redukowane do 90% poprzedniej wartości.

Pokazany na slajdzie scenariusz, w którym do dwóch istniejących zespołów dodano trzeci o budżecie w wysokości 20 jednostek wyzwolił akcję reguły. Ponieważ sumaryczny budżet wynosi teraz 120 jednostek, akcja reguły redukuje budżety zespołów. Modyfikacja budżetów cyklicznie uruchamia akcję reguły. Ograniczony budżet do wysokości 108 jednostek w dalszym ciągu przekracza dopuszczalną wartość. Akcja rekurencyjnie wywołanej reguły dokonuje kolejnej redukcji budżetu, tym razem do akceptowalnego poziomu. W tym momencie, cykl wywoływania reguły zastaje zatrzymany.

Statyczna walidacja własności stopu

Graf wyzwalania (GW)

Węzły grafu reprezentują zbiór aktywnych reguł. Dwa węzły grafu GW, reprezentujące reguły R1 i R2 są połączone krawędzią skierowaną od węzła R1 do R2, jeżeli kod akcji reguły R1 zawiera instrukcje manipulacji danymi na relacji, której modyfikacja jest zdarzeniem uaktywniającym regułę R2.

Brak cyklu w grafie wyzwalania oznacza, że analizowany zbiór reguł posiada własność stopu. Występowanie cyklu w grafie wyzwalania oznacza, że zbiór reguł może nie posiadać własności stopu.

Jak wiadomo z teorii algorytmów formalna analiza własności stopu jest w ogólności problemem nierozwiązywalnym. W praktyce dla weryfikacji własności stopu stosuje się składniową analizę zbioru reguł. Analiza składniowa jest zachowawcza, co oznacza, że istnieją zbiory reguł posiadające własność stopu, które zostaną uznane w wyniku analizy składniowej jako niepoprawne.

Analiza składniowa opiera się na badaniu własności grafu wyzwalania. Graf ten jest konstruowany w następujący sposób. Węzłami grafu są aktywne reguły z analizowanego zbioru reguł. Krawędzie w grafie reprezentują składniowe zależności między regułami. Między węzłami, które reprezentują reguły R1 i R2 jest wstawiana krawędź skierowana od R1 do R2, jeżeli kod akcji reguły R1 zawiera operacje, które są zdarzeniem wywołującym regułę R2. Na poprzednich dwóch slajdach takie zależności zostały pokazane za pomocą strzałek. Brak cyklu w grafie wyzwalania oznacza, że badany zbiór reguł posiada własność stopu. Występowanie cyklu w grafie oznacza, że dany zbiór reguł może nie posiadać własności stopu.

Na rysunku pokazano grafy wyzwalania dla zbioru reguł „*zwiększ_rangę”* i „*zwiększ_premię”* oraz dla reguły „*kotwica_budżetowa”*. Ze względu na zachowawczość analizy składniowej mimo, że obydwa grafy są cykliczne, jednak w praktyce tylko pierwsza para reguł nie posiada własności stopu.

Komercyjne aktywne systemy baz danych zazwyczaj dokonują składniowej analizy własności stopu tylko podczas kompilacji pojedynczych reguł. W przypadku zbioru reguł brak własności stopu jest diagnozowany przez system dopiero podczas aktywowania reguł. Dla bardziej złożonych przypadków błąd taki może być wykryty dopiero po długim okresie eksploatacji systemu.

Własność determinizmu stanu

```
create trigger zwiększ_premię_1
   after update of wysokość on Sprzedaż
   for each row
   when new.wysokość – old.wysokość > 100
   begin
      update Pracownicy
      set premia = premia + 1000
      where nr = new.nr_prac;
   end

create trigger zwiększ_premię_2
   after update of wysokość on Sprzedaż
   for each row
   when new.wysokość – old.wysokość > 500
   begin
      update Pracownicy
      set premia = 1.5*premia
      where nr = new.nr_prac;
   end
```

Premia=5000
↓
zwiększ_premię_1
↓
zwiększ_premię_2
↓
Premia=**9000**

Premia=5000
↓
zwiększ_premię_2
↓
zwiększ_premię_1
↓
Premia=**8500!!!**

Następną wymaganą własnością zbioru aktywnych reguł jest determinizm stanu. Problem determinizmu stanu jest konsekwencją występowania w zbiorze, reguł uaktywnianych tym samym zdarzeniem i modyfikujących ten sam fragment bazy danych. Kolejność odpalania akcji dwóch reguł uaktywnianych tym samym zdarzeniem jest w komercyjnych bazach danych ustalana arbitralnie przez system zarządzania bazą danych. Jeżeli akcje takich reguł nie są komutatywne, to końcowy stan bazy danych jest zależny od arbitralnie ustalonej przez system kolejności ich uruchomienia.

Przykład taki pokazano na slajdzie. W bazie danych pracowników, którzy są akwizytorami zdefiniowano dwie aktywne reguły automatycznie ustalające wysokość premii na podstawie wyników sprzedaży danego pracownika. Pierwsza reguła pracownikowi, który zwiększył sprzedaż o 100 jednostek podnosi premię o 1000 zł. Druga reguła w wypadku zwiększenia sprzedaży przez pracownika o ponad 500 jednostek podnosi mu pensję o 50%. W sytuacji gdy wzrost sprzedaży przekroczy 500 jednostek odpalone zostaną akcje obydwu reguł. Ponieważ akcje te nie są komutatywne, końcowy stan bazy danych zależy od kolejności ich odpalenia. Wypadek ten ilustrują scenariusze z prawej strony slajdu. Dla początkowego stanu premii pracownika wynoszącego 5000 zł odpalenie akcji w kolejności najpierw reguła pierwsza, a potem reguła druga ustala ostatecznie premię w wysokości 9000 zł. Odwrotna kolejność odpalenia reguł daje w efekcie inną kwotę premii wynoszącą w tym wypadku 8500 zł.

Silniejszym wymaganiem niż determinizm stanu bazy danych jest determinizm zachowania systemu informatycznego, obejmujący poza stanem bazy danych dodatkowo informacje wychodzące na zewnątrz systemu w postaci zawartości ekranów komputerów, wydruków czy innych wysyłanych na zewnątrz systemu komunikatów.

Podobnie jak w wypadku własności stopu również determinizm stanu bazy może być weryfikowany przez zachowawczą analizę składniową weryfikującą komutatywność akcji reguł.