Michał Stochmiałek <misto@e-informatyka.pl>

Strumieniowe bazy danych

STREAM: The Stanford Data Stream Management System

Politechnika Wrocławska, Wrocław, styczeń 2005

Streszczenie

Artykuł wprowadza czytelnika w teoretyczne podstawy strumieniowych baz danych oraz systemów zarządzania strumieniami danych (ang. *Data Stream Management System*). Jest to innowacyjne podejście to zarządzania danymi. W odróżnieniu od modelu przyjętego w relacyjnych bazach danych, model danych w DSMS przyjmuje postać ciągłych, dynamiczny, zmiennych w czasie strumieni danych. Artykuł opisuje również idee ciągłych zapytań (ang. *continuous queries*) i prezentuje ją na konkretnym przykładzie.

1. Wprowadzenie

Przetwarzanie i analiza danych w tradycyjnych bazach danych jest zwykle rozwiązywana w następujący sposób. Potrzebne dane są ładowane i utrwalane w trakcie atomowych i niepodzielnych transakcji, następnie wykonywane są odpowiednie zapytania, które wykonują zasadnicza część analizy.

Rozważmy jednak pewną klasę aplikacji, do których nowe dane napływają ciągle z różnych źródeł. Dodatkowo wymagana jest ciągła i dynamiczna ich analiza oraz prezentacja jej wyników. Przykładem takich systemów może być monitorowanie ruchu sieciowego lub drogowego, analiza finansowa, analiza logów serwerów internetowych czy kliknięć użytkownika w portalu.

Wymagania powyższej klasy systemów mogą być zabójcze dla tradycyjnych baz danych, które nie zostały zaprojektowane do przetwarzania ciągłych strumieni danych.

Rozpoznanie powyższej klasy systemów zapoczątkowało szeroko zakrojone badania nad przetwarzaniem strumieni danych. Są one prowadzone między innymi na uniwersytecie w Stanford (projekt STREAM [WWW2005a]), ale również w ośrodkach Brandeis University, Brown University oraz MIT (projekt Aurora [WWW2005b]) czy nawet w Berkeley (projekt Telegraph [WWW2005c]). Wielokrotnie temu tematowi przeznaczane były całe sesje plenarne na konferencjach *Very Large Data Bases* czy *Management of Data*. Badania zaowocowały wieloma prototypami *systemów zarządzania strumieniami danych* (ang. *Data Stream Management System*), które aktualnie stają się coraz bardziej dojrzałe.

System zarządzania strumieniami danych ma na celu przetwarzanie ciągłych, nieskończonych, zmiennych w czasie strumieni danych. Wiąże się to nierozłącznie z ideą ciągłych zapytań, które często przetwarzane są przez cały okres działania aplikacji. Dostarczają aktualnych wyników analiz czy innych rezultatów przetwarzania danych.

Jak już wspomniano, badania zaowocowały wieloma projektami DSMS. Powyższy artykuł skupi się na koncepcjach wypracowanych przez zespół z uniwersytetu z Stanford, czyli na projekcie STREAM.

2. STREAM: The Stanford Data Stream Management System

Najważniejsza ideą wypracowaną przez zespół projektu STREAM jest model strumieni danych, jak również język CQL (*continuous query language*), w którym rejestruje się ciągłe zapytania w systemie.

2.1. Model danych

Dane strumieniowych bazach danych[1] mogą być modelowane jako **strumienie** lub **relacje**. Podobnie jak w tradycyjnych bazach danych każdy strumień i relacja posiada określony schemat definiujący jego atrybuty. Dla oznaczenia czasu utworzenia (powstania) elementów strumieni lub relacji użyta zostanie dyskretna i uporządkowana dziedzina czasu $\mathcal{T}$. Dla uproszczenia przyjęto: $\mathcal{T} = \{0, 1, \ldots\}$, gdzie $\tau = 0$ oznacza najwcześniejszą *chwilę*.

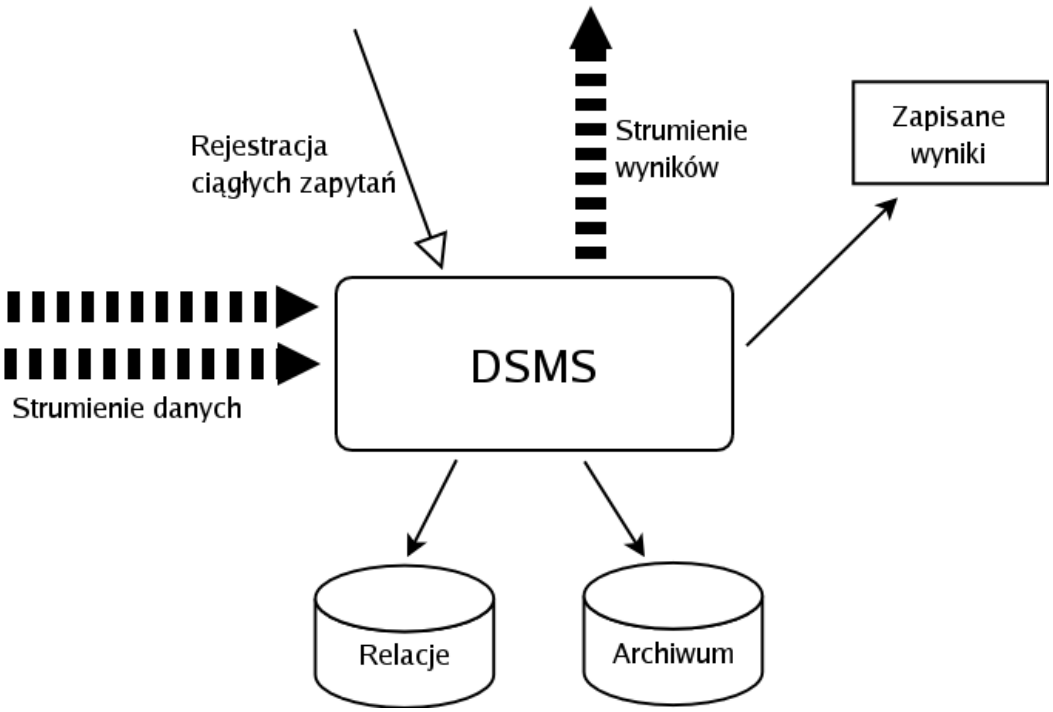
Strumień S jest to wielozbiór elementów postaci $\langle s, \tau \rangle$, gdzie s jest krotką, a $\tau \in \mathcal{T}$ jest czasem pojawienia się elementu (ang. *timestamp*). Strumień S może być nieskończony. Wartości τ w elementach strumienia mogą się powtarzać (elementy pojawiły się w tym samym czasie), jak również mogą w ogóle nie występować. Implementacja projektu STREAM modeluje strumień jako sekwencję elementów $\langle s, \tau \rangle$ uporządkowaną względem wartości τ.

[1] Autor używając terminu 'strumieniowa baza danych' odnosi się do systemu DSMS opracowanego w projekcie STREAM.

Relacja R jest to odwzorowanie z $\mathcal{T}$ na skończony (ale nieograniczony) wielozbiór krotek s. Należy zwrócić uwagę na różnicę pomiędzy definicją relacji z tradycyjnych baz danych a strumieniowych. W standardowym modelu relacyjnym relacja to zwykły zbiór (lub wielozbiór) krotek, który nie bierze pod uwagę zmiennej czasowej. Natomiast w strumieniowym modelu, relacja jest zmienna w czasie. Tak więc, jeżeli określimy chwilę τ, to wartość $R(\tau)$ można interpretować już w standardowym, relacyjnym ujęciu.

2.2. Zasada działania

Na rysunku 1 przedstawiony został poglądowy schemat działania aplikacji opartej na strumieniowej bazie danych.



Rys. 1. Prosty schemat działania systemu opartego na strumieniowej bazie danych

Danymi wejściowymi bazy strumieniowej są strumienie o zdefiniowanym schemacie danych. System zarządzania przetwarza je na podstawie zarejestrowanych ciągłych zapytań. Wyniki tych zapytań mogą być przesyłane jako strumień do aplikacji albo zapisywane. Dodatkowo przetwarzanie zapytania może korzystać z danych zapisanych w relacjach oraz z archiwum danych ze strumieni.

2.3. Język CQL

Język CQL rozszerza język SQL, którego semantyka jest ogólnie znana. W rezultacie uzyskaliśmy język deklaratywny, który jest bardzo łatwy w opanowaniu, a zarazem potężny w definiowaniu operacji na strumieniach.

Język CQL (podobnie jak SQL) transformuje się do planu zapytania składającego się z operatorów. Operatory te można podzielić na trzy klasy:

- operatory **strumień-relacja** produkujące na podstawie strumienia S relację R o takim samym schemacie. W każdej chwili τ, relacja $R(\tau)$ powinna być obliczalna z elementów strumienia S aż do chwili τ.

- operatory **relacja-relacja** produkujące relację R na podstawie relacji $R_1, \ldots, R_n$. W każdej chwili τ, relacja $R(\tau)$ powinna być obliczalna z relacji $R_1(\tau), \ldots, R_n(\tau)$.
- operatory **relacja-strumień** produkujące na podstawie relacji R strumień S o takim samym schemacie. W każdej chwili τ, strumień S w chwili τ powinien być obliczalny z elementów relacji R aż do chwili τ.

2.3.1. Operatory strumień-relacja

W projekcie STREAM wszystkie operatory strumień-relacja bazują na idei przesuwającego się okna (ang. *sliding window*). Polega ona na ekstrakcji tylko najnowszych danych ze strumienia.

Okno oparte na przedziale czasowych Pierwszym typem przesuwającego się okna jest okno oparte na przedziale czasowym (ang. *time-based sliding window*). Wynikowa relacja '`S [Range T]`' zawiera w chwili τ wszystkie elementu strumienia S, które pojawiły się do chwili τ po rozpoczęciu przedziału czasowego `T`.

Przykładowo '`S [Range 40 Seconds]`' zawiera wszystkie krotki, które pojawiły się w strumieniu S w czasie ostatnich czterdziestu sekund. Szczególnym przypadkiem tego operatora jest '`S [Now]`', który zawiera wszystkie krotki, które pojawiły się w tej chwili oraz '`S [Range Unbounded]`', ktora zawiera wszystkie krotki, które pojawiły się kiedykolwiek w strumieniu.

Okno oparte na ilości krotek Drugim typem przesuwającego się okna jest okno oparte na ilości krotek (ang. *tuple-base sliding window*). Wynikowa relacja '`S [Rows N]`' zawiera `N` najnowszych elementów strumienia S (należy zwrócić uwagę, że wynik tego okna może być niedeterministyczny, jeżeli w strumieni znajduje się więcej niż `N` najnowszych elementów).

Przykładowo '`S [Rows 1]`' zwraca najnowszą krotkę ze strumienia. Natomiast wynik wyrażenia '`S [Rows Unbounded]`' jest identyczny z '`S [Range Unbounded]`'.

Okno podzielone Ostatnim typem przesuwającego się okna jest okno podzielone (ang. *partitioned sliding window*). Wynikowa relacja '`S [Partition By` $\mathtt{A}_1, \ldots, \mathtt{A}_k$ `Rows N]`' jest tworzona w następujący sposób. Strumień S jest dzielony na podstrumienie zawierające elementy o identycznych wartościach atrybutów $\mathtt{A}_1, \ldots, \mathtt{A}_k$, następnie z tych podstrumieni wybierane jest `N` najnowszych elementów i tworzona jest relacja wynikowa.

Przykładowo relacja '`Wyplaty [Partition By pracownik_id Rows 5]`' zawiera dane dotyczące pięciu ostatnich wypłat każdego pracownika w pewnej korporacji.

2.3.2. Operatory relacja-relacja

Operatory relacja-relacja w języku CQL wywodzą się z języka SQL. Ich działanie jest analogiczne, przy czym trzeba przypomnieć, że operują one na relacjach tylko w tych samych chwilach τ.

```
SELECT P.pracownik_id, P.imie, P.nazwisko
FROM wchodzacyDoBudynkuStream [Range 30 Seconds] as W, pracownicy as P
WHERE W.pracownik_id = P.pracownik_id
```

Rozważmy powyższe przykładowe zapytanie, które w każdej chwili τ zwraca listę pracowników, którzy weszli do budynku firmy w ostatnich trzydziestu sekundach. Wykorzystane zostały następujące operatory klasy relacja-relacja: operator złączenia oraz operator projekcji.

2.3.3. Operatory relacja-strumień

Język CQL zawiera trzy operatory klasy relacja-strumień: strumień elementów dodanych `IStream`, strumień elementów usuniętych `DStream` oraz strumień elementów relacji `RStream`.

Strumień elementów dodanych Operator `IStream` tworzy, na podstawie relacji R, strumień, który w chwili τ zawiera elementy postaci $\langle s, \tau \rangle$, gdzie s należy do zbioru elementów dodanym do relacji R w chwili τ.

```
SELECT IStream(*)
FROM wchodzacyDoBudynkuStream [Range Unbounded]
WHERE wejscie = 'tylne'
```

Dla przykładu powyższe zapytanie tworzy filtr na strumieniu `wchodzacyDoBudunkuStream`, który do wynikowego strumienia przepuszcza tylko krotki dotyczące wejść tylnym wejściem.

Strumień elementów usuniętych Operator `DStream` tworzy, na podstawie relacji R, strumień, który w chwili τ zawiera elementy postaci $\langle s, \tau \rangle$, gdzie s należy do zbioru elementów usuniętych z relacji R w chwili τ.

```
SELECT DStream(*)
FROM wchodzacyDoBudynkuStream [Range 30 Minutes]
```

Przykładowo powyższe zapytanie jest filtrem na strumieniu `wchodzacyDoBudunkuStream`. Produkuje strumień wynikowy, który zawiera krotki dotyczące wejść trzydzieści minut temu i później.

Strumień elementów relacji Operator `RStream`, tworzy na podstawie relacji R, strumień, który w chwili τ zawiera elementy postaci $\langle s, \tau \rangle$, gdzie s należy do relacji $R(\tau)$.

```
SELECT RStream(*)
FROM wchodzacyDoBudynkuStream [Now]
WHERE wejscie = 'tylne'
```

Powyższe przykładowe zapytanie produkuje identyczny wynik jak przykład dotyczący operatora `IStream`.

2.4. Konkretny przykład

Rozważmy bardziej konkretny przykład. Jest to uproszczona wersja *benchmarku Linear Road* dla systemów zarządzających strumieniami danych [TCA03][ABW03].

System Linear Road jest systemem naliczającym wysokość opłat za przejazd autostradą. System bazuje na zmiennej opłacie w zależności od natężenia ruchu. Każdy samochód jest wyposażony w nadajnik, który co trzydzieści sekund emituje do centrali swoją aktualną pozycje i prędkość. Następnie dane ze wszystkich pojazdów są analizowane i naliczane są opłaty.

System Linear Road działa na autostradzie podzielonej na pewną liczbę odcinków. Opłata jest nalicza tylko gdy samochód jest na zatłoczonym odcinku autostrady, czyli taki na którym samochody od pięciu minut poruszają się prędkością niższą od 60 km/h. Wysokość opłaty zależy od ilości samochodów na danym odcinku i jest obliczana według wzoru: $bazowa\,opłata \times (ilość\,samochodów - 150)$.

W terminologii strumieniowych baz danych można wyrazić, że system Linear Road składa się z:

- pojedynczego strumienia wejściowego zawierającego informacje przesyłane z samochodów o schemacie:
  `predkoscSamochodowStream(samochodId, odcinekId, predkosc)`
- pojedynczego ciągłego zapytania, które oblicza aktualną wysokość opłat
- pojedynczego strumienia wyjściowego zawierającego aktualne wysokości opłat o schemacie:
  `oplatyZaAutostrade(samochodId, oplata)`

Docelowe zapytanie obliczające opłaty zostało podzielone na trzy mniejsze. Poniżej zostały one po kolei przedstawione.

Zapytanie 1 Pierwsze zapytanie oblicza w każdej chwili aktualny zbiór zatłoczonych odcinków autostrady. Na strumieniu wejściowym nałożone jest okno, którego wynikiem jest relacja zawierająca informacje z ostatnich pięciu minut. Następnie są one grupowane względem identyfikatora odcinka autostrady. Dodatkowo nałożony jest warunek na średnią prędkość samochodów w tym odcinku. Wynikowa relacja posiada schemat `zatloczoneOdcinki(odcinekId)`.

```
SELECT odcinekId
FROM  predkoscSamochodowStream [Range 5 Minutes]
GROUP BY odcinekId
HAVING AVG(predkosc) < 60
```

Zapytanie 2 Drugie zapytanie oblicza liczbę samochodów na każdym odcinku autostrady. Zakładamy, że samochód opuścił autostradę, jeżeli od 30 sekund nie przesłał informacji o swojej aktualnej pozycji.

Na strumieniu wejściowym zostało nałożone okno produkujące relację zawierającą informacje z ostatnich trzydziestu sekund. Następnie jest obliczana liczba samochodów na danym odcinku.

```
SELECT odcinekId, COUNT(samochodId) as liczba
FROM  predkoscSamochodowStream [Range 30 Seconds]
GROUP BY odcinekId
```

Zapytanie 3 Trzecie zapytanie produkuje strumień wynikowy na podstawie danych obliczonych w poprzednich zapytaniach. Jest on przesyłany z powrotem do nadajników w samochodach.

W każdej chwili, okno `Now` na strumieniu wynikowym zwraca aktualne dane o pozycjach samochodów. Są one łączone z aktualnym zbiorem zatłoczonych odcinków autostrady oraz z relacją zawierającą liczbę samochodów na każdym odcinku w danej chwili. Następnie liczona jest wysokość opłaty na podstawie przedstawionego wcześniej wzoru. Ostatecznie za pomocą operatora `RStream` tworzony jest strumień wynikowy.

```
SELECT RStream(S.samochodId,
           oplataBazowa * (LS.liczba - 150) as oplata
FROM  predkoscSamochodowStream [Now] as S,
      zatloczoneOdcinki as ZO, liczbaSamochodow as LS
WHERE S.odcinekId = ZO.odcinekId
      and  ZO.odcinekId = LS.odcinekId
```

3. Podsumowanie

Strumieniowe bazy danych należy na razie traktować jako nowinkę bazodanową. Wszystkie wspomniane w artykule projekty są w fazach prototypu i rozwijane są w kręgach akademickich.

Są to raczej pola eksperymentalne, gdzie ich autorzy testują swoje pomysł. Strumieniowe bazy danych powinny zostać przeniesione na grunt komercyjny, aby mogły zostać ogólnie zaakceptowane przez przemysł informatyczny.

Druga kwestia, to nietrudno nie zauważyć, że aktualnie monopol na rynku bazodanowym prowadzą relacyjne bazy danych. Jest to nie tylko monopol rynkowy, ale dominacja w świadomości informatyków. Aktualnie prawie każdy system wymagający trwałości danych biznesowych jest implementowany za pomocą relacyjnych baz danych, nawet gdy ich użycie nie jest najlepszym rozwiązaniem. Przykładowo, nieduże systemy oparte na językach obiektowych często stosują relacyjne bazy danych, pomimo tego że muszą radzić sobie z mapowaniem obiektowo-relacyjnym, które wnosi spore obniżenie wydajności (duża ilość alokacji obiektów podczas tworzenia wyników zapytań). Nie jest to opłacalne dla małej ilości danych (np. mniejszej niż 100 mb). W takich przypadkach lepiej użyć obiektową bazę danych, która wydaje się bardziej naturalna dla systemów obiektowych.

Niestety obiektowe bazy danych wydają się przegrywać z monopolem baz relacyjnych i nie jest to już tylko problem rynkowy, lecz również bezgranicznego (czasami irracjonalnego) zaufania w paradygmat relacyjny. Podobnie można obawiać się, że przyszłość strumieniowych baz danych będzie wyglądać analogicznie.

Literatura

[ABW03] A. Arasu, S. Babu, J. Widom, *The CQL Continuous Query Language: Semantic Foundations and Query Execution*

[BDMW03] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, *Models and Issues in Data Stream Systems*, czerwiec 2002

[STREAM03] The STREAM Group, *Stanford Data Stream Management System, Overview*, marzec 2003

[TCA03] R. Tibbetts, M. Cherniack, A. Arasu, *Linear road: A stream data management benchmark*, lipiec 2003, `http://www.cs.brown.edu/research/aurora/linear-road.pdf`.

[WWW2005a] Strona projektu STREAM, `http://www-db.stanford.edu/stream/`

[WWW2005b] Strona projektu Aurora, `http://www.cs.brown.edu/research/aurora/`

[WWW2005c] Strona projektu Telegraph, `http://telegraph.cs.berkeley.edu/`