ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2005
Seria: TRANSPORT z. 58 Nr kol. 1688

Grzegorz KAROŃ[1], Renata ŻOCHOWSKA[2]

# PRZETWARZANIE INFORMACJI W SYSTEMACH TRANSPORTOWYCH - STRUMIENIOWE BAZY DANYCH

**Streszczenie.** W artykule scharakteryzowano podstawowe modele baz danych ze szczególnym uwzględnieniem najnowszego modelu: strumieniowych systemów baz danych DSMS. Podstawowe definicje DSMS oraz operatory języka CQL przedstawione zostały z wykorzystaniem przykładów mających zastosowanie w przetwarzaniu informacji w systemach transportowych.

# DATA PROCESSING IN TRANSPORTATION SYSTEMS – DATA STREAM SYSTEMS

**Summary.** The basic database models have been described in the article. The most recent model: Data Stream Systems DSMS has been taken into special consideration. Denotational semantics of DSMS and oparators of CQL have been presented by application in transportation systems data processing.

## 1. WPROWADZENIE

Planowanie systemów transportowych oraz modelowanie procesów transportowych odbywa się na podstawie przeprowadzanych różnorodnych badań. Są to między innymi badania związane z: pomiarami natężenia potoków ruchu samochodowego, potoków pieszych, wykorzystaniem powierzchni parkingowej, napełnieniem pojazdów komunikacji zbiorowej, poziomem swobody ruchu na skrzyżowaniach i wiele innych. Podczas pomiarów zachodzi konieczność rejestracji danych a następnie zebrania ich w określonej formie i opracowania na potrzeby określonych badań.

Odpowiednie zorganizowanie danych w postaci bazy danych a następnie efektywne ich przetwarzanie poprzez system zarządzania bazą danych umożliwia szybkie i sprawne dostarczanie danych do badań oraz otrzymywanie niezbędnych wyników. Dzięki systemom baz danych, a szczególnie tak zwanym hurtowniom danych [8,20], możliwa jest zamiana danych w wiedzę, a przez to osiąganie następujących korzyści:

- otrzymywanie na czas aktualnej informacji,
- możliwość tworzenia przekrojowych analiz danych,
- szybsze dostrzeganie pojawiających się problemów i możliwości ich rozwiązania,

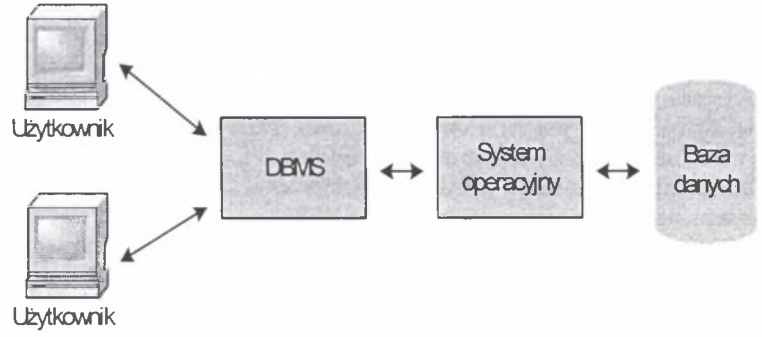
[1] Wydział Transportu, Politechnika Śląska, ul. Krasińskiego 8, 40-019 Katowice, tel/fax (+48 32) 2554159, grzegorz.karon@polsl.pl
[2] Wydział Transportu, Politechnika Śląska, ul. Krasińskiego 8, 40-019 Katowice, tel/fax (+48 32) 2554121, renzoc@polsl.katowice.pl

- "zagłębianie się" w dane, drążenia ich od ogólnego spojrzenia na całość do szczegółowej analizy wybranych, interesujących składników,
- możliwość prowadzenia prognoz – analiz,
- sporządzanie dowolnych raportów, na przykład raportów porównujących wartości bieżące ze wskaźnikami z poprzednich okresów,
- integracja danych, dbanie o ich spójność i zgodność z przyjętymi standardami,
- zasilanie danymi odbywa się automatycznie, co zmniejsza ryzyko powstania błędów w danych.

Powstała już tak zwana inżynieria informacyjna - nauka zajmująca się identyfikacją potrzeb informacyjnych oraz rozwojem systemu informatycznego, który udostępnia informacje odbiorcom. Inżynieria informacyjna jest odpowiedzialna za proces filtrowania, który redukuje masy zbędnych danych do komunikatu zawierającego wymaganą informację, a następnie dostarcza go do właściwych osób, w każde wymagane miejsce na czas, w czytelnej formie i po ustalonym koszcie, tak żeby możliwe było podjęcie właściwych decyzji.

## 2. ROZWÓJ BAZ DANYCH

Przez bazę danych rozumie się uporządkowany zbiór danych z nałożoną na niego wewnętrzną strukturą, przechowywany w pamięci komputera. Jest to zestaw: danych, programów i innych środków pozwalających na utrzymywanie, zabezpieczanie, przetwarzanie i udostępnianie danych dla użytkowników [7,9,16]. Bazy danych są komputerowymi reprezentacjami fragmentów istniejącego świata rzeczywistego. Informacje (dane) mają wartość tylko wtedy, gdy są dokładne i dostępne na czas. Specjalizowane oprogramowanie do obsługi baz danych, zapewniające użytkownikom możliwości definiowania, zapisywania, wyszukiwania i aktualizacji danych zwartych w bazie, to system zarządzania bazą danych DBMS (ang. *Database Management System*). Na rysunku 1 pokazano schemat przetwarzania informacji w systemach bazodanowych.




Rys. 1. Przetwarzanie informacji w systemach z bazą danych
Fig. 1. Data processing in Database Systems

Przed powstaniem systemów zarządzania bazami danych, informacje przechowywane były w wielu różnych formatach i nie istniał jeden spójny system do zarządzania nimi. W roku 1960 firma IBM wprowadziła prototyp bazy danych w celu pokazania, jak dane mogą być przechowywane, przetwarzane i aktualizowane z wykorzystaniem jednolitego formatu.

### 2.1. Hierarchiczny model danych

System Zarządzania Informacją – IMS (ang. *Information Management System*), opracowany w roku 1960 przez IBM, był rewolucyjnym pomysłem, gdyż umożliwiał dostęp do danych przez kilka programów napisanych w różnych językach, a także pozwalał na obsługę przez wielu użytkowników jednocześnie [9]. IMS wykorzystywał model hierarchiczny polegający na organizacji danych w schodzące w dół relacje jeden-do-wielu, gdzie każdy poziom hierarchii ma pierwszeństwo w stosunku do znajdującego się poniżej. Dane tworzyły strukturę drzewa z węzłami połączonymi między sobą za pomocą tak zwanych gałęzi. Główne zalety hierarchicznych baz danych to:
- łatwość zastosowania modelu hierarchicznego,
- bardzo duża szybkość działania,
- są to systemy, które okazały się na tyle dobre, że są nadal wykorzystywane.

Do wad zaliczyć można:
- ścisłe reguły dotyczące relacji,
- wstawianie i usuwanie danych może okazać się bardzo skomplikowane,
- dostęp do niższych warstw jest możliwy tylko poprzez warstwy nadrzędne,
- trudności w modelowaniu relacji typu wiele-do-wielu.

### 2.2. Sieciowy model danych

W latach sześćdziesiątych XX wieku powstały również systemy zarządzania bazami danych, wykorzystujące model sieciowy, stworzony przez organizację CODASYL (ang. *Conference On Data Systems Languages*). Specyfikacje CODASYL zawierały definicje schematu bazy danych, języka kontroli urządzeń DMCL (ang. *Device Media Control Language*) oraz języka manipulacji danymi DML (ang. *Data Manipulation Language*). Logiczna struktura bazy danych definiowana była za pomocą języka definiowania danych DDL (ang. *Data Definition Language*).

W modelu sieciowym dane posiadają wewnętrzne identyfikatory (lub adresy), natomiast związki semantyczne pomiędzy danymi reprezentowane są poprzez powiązania. Struktura danych tworzy więc graf (sieć). Wady tego modelu to:
- duża złożoność i związana z tym trudność w użyciu,
- niezwykle trudna implementacja zmian strukturalnych.

### 2.3. Relacyjny model danych

W roku 1970 dr Edgar Ted Codd z firmy IBM w artykule artykuł pt. "A relational model of data for large shared data banks" zaprezentował relacyjny model danych, w którym dane przechowywane są w tabelach (relacjach). Jedną z największych zalet modelu relacyjnego w stosunku do poprzedników jest jego prostota. Zamiast konieczności poznawania mnóstwa komend języka DML model relacyjny wprowadził język SQL (ang. *Structured Query Language*) w celu ułatwienia dostępu do danych i ich modyfikacji, a całą bazą zarządza system RDBMS (ang. *Relational DataBase Management System*) [10,11,13].

Tabele to dwuwymiarowe tablice, w których wiersze nazywane są krotkami (ang. *tuples*) lub rekordami, natomiast kolumny atrybutami (ang. *attributes*) lub polami. Tabela posiada zawsze pole lub kilka pól, tworzące dla niej klucz główny (ang. *primary key*), który jednoznacznie identyfikuje wiersz w tabeli. Klucz obcy (ang. *foreign key*), którego wartości pochodzą z klucza głównego innej tabeli, pozwala na łączenie tabel między sobą [9,10,22].

Relacyjne bazy danych wprowadziły następujące ulepszenia w stosunku do hierarchicznych i sieciowych baz danych:

- wykorzystują tabele z wierszami i kolumnami, które dla użytkownika końcowego są proste i łatwe do zrozumienia,
- nie występują złożone diagramy, wykorzystywane w przypadku modeli hierarchicznych i sieciowych,
- tabele nie są ze sobą połączone na sztywno, do tabel mogą być dodawane kolumny, tabele mogą być dołączane do bazy danych, a nowe relacje mogą być tworzone bez konieczności wprowadzania istotnych zmian,
- dzięki wykorzystaniu języka SQL użytkownik określa jedynie warunki odnośnie do poszukiwanych danych - pobraniem odpowiednich danych zajmuje się RDBMS.

### 2.4. Strumieniowy model danych

Strumieniowe bazy danych powstały w oparciu o relacyjne bazy danych. Są to bazy, do których dane napływają ciągle z różnych źródeł. Tradycyjne bazy nie są zaprojektowane do przetwarzania ciągłych strumieni danych. System zarządzania strumieniami danych DSMS (ang. *Data Stream Management System*) ma na celu przetwarzanie ciągłych, nieskończonych, zmiennych w czasie strumieni danych. Wiąże się to z ideą ciągłych zapytań wykonywanych w nowo opracowanym języku CQL (ang. *Continous Query Language*), które często przetwarzane są przez cały okres działania aplikacji [1, 3].

Przykładem takich systemów może być monitorowanie ruchu drogowego, analiza finansowa i wiele innych. Szeroko zakrojone badania nad przetwarzaniem strumieni danych są prowadzone między innymi na uniwersytecie w Stanford, w ośrodkach Brandeis University, Brown University, MIT oraz Berkeley.

## 3. STRUMIENIOWE BAZY DANYCH

Dane w bazach strumieniowych są modelowane jako strumienie lub relacje. Podobnie jak w tradycyjnych bazach danych każdy strumień i relacja posiadają określony schemat definiujący jego atrybuty (pola). Elementem wyróżniającym model strumieniowy od wcześniejszych modeli jest zmienna oznaczająca czas utworzenia krotek w strumieniu: $t = \{0,1,...\}$, gdzie $t = 0$ oznacza chwilę początkową.

**Strumień** $S$ to ciąg elementów postaci $(s,t)$ uporządkowanych względem wartości $t$, gdzie $s$ jest krotką, a $t$ jest czasem pojawienia się elementu (ang. *timestamp*).

**Relacja** $R$ to odwzorowanie z $t$ na skończony (ale nieograniczony) wielozbiór krotek $s$. Różnica pomiędzy definicją relacji w tradycyjnych bazach danych a strumieniowych polega na tym, że w standardowym modelu relacyjnym relacja to zbiór krotek (wierszy w tabeli), który nie uwzględnia zmiennej czasowej. Natomiast w strumieniowym modelu relacja jest zmienna w czasie. Tak wiec, jeżeli określona zostanie chwila $t$, to wartość $R(t)$ można interpretować już w standardowym, relacyjnym ujęciu [3, 15].

### 3.1. Język CQL

Przetwarzanie strumieniowej bazy danych odbywa się poprzez wykonywanie zapytań w języku CQL (ang. *Continous Query Language*). Jest on rozwinięciem języka SQL stosowanego w bazach relacyjnych. Występują w nim trzy klasy operatorów [1,5]:

- operatory strumień-relacja: ruchome okno czasowe, ruchome okno krotek, ruchome okno podzielone,
- operatory relacja-relacja,

- operatory relacja-strumień: strumień elementów dodanych, strumień elementów usuniętych i strumień elementów relacji.

### 3.2. Operatory strumień-relacja

Operatory strumień-relacja generują na podstawie strumienia $S$ relację $R(t)$ bazując na idei przesuwającego się okna (ang. *sliding window*), która polega na analizie tylko danych znajdujących się w określonym przedziale.

**Ruchome okno czasowe** (ang. *time-based sliding window*), oznaczane jako `S[Range` $\varpi$ `]`, generuje ze strumienia $S$ w chwili $t$ relację $R(t)$ zawierającą wszystkie krotki strumienia $S$, które pojawiły się od chwili $\varpi - t$ do $t$.

**Przykład:**

```
Strumien_pojazdow[Range 5 Minutes]
```

Wynikiem operatora `Range` jest relacja zawierająca wszystkie krotki (pojazdy), które zarejestrowane zostały w strumieniu `Strumien_pojazdow` w czasie ostatnich 5 minut.

Szczególnym przypadkiem operatora jest `S[Now]`, zawierający wszystkie krotki, które pojawiły się w tej chwili oraz `S[Range Unbounded]`, zawierająca wszystkie krotki, które pojawiły się kiedykolwiek w strumieniu.

**Ruchome okno krotek** (ang. *tuple-base sliding window*). Wynikiem działania na strumieniu $S$ operatora ruchomego okna krotek `S[Rows N]` w chwili $t$ jest relacja $R(t)$ zawierająca $N$ krotek strumienia $S$, które pojawiły się do chwili $t$.

**Przykład:**

```
Strumien_pojazdow[Rows 10]
```

Operator `Rows` tworzy relację zawierającą 10 najnowszych krotek (10 ostatnio zarejestrowanych pojazdów) ze strumienia.

**Ruchome okno podzielone** (ang. *Partitioned sliding window*). Wynikowa relacja `S[Partition By A1, . . . , Ak Rows N]` jest tworzona w następujący sposób: strumień $S$ dzielony jest na podstrumienie zawierające elementy o identycznych wartościach atrybutów $A_1, \ldots, A_k$, następnie z tych podstrumieni wybierane jest $N$ najnowszych elementów i tworzona jest relacja wynikowa.

**Przykład:**

```
Strumien_pojazdow [Partition By rodzaj_pojazdu Rows 5]
```

Relacja zawiera dane dotyczące pięciu ostatnio zarejestrowanych pojazdów każdego rodzaju (`rodzaj_pojazdu = {SOD, SC, A, AP}` ).

### 3.3. Operatory relacja-relacja

Operatory relacja-relacja generują relację $R$ na podstawie relacji $R_1, \ldots, R_n$. W każdej chwili $t$ relacja $R(t)$ obliczana jest z relacji $R_1(t), \ldots, R_n(t)$. Operatory relacja-relacja w języku CQL wywodzą się z języka SQL. Ich działanie jest analogiczne, przy czym operują one na relacjach tylko w tych samych chwilach $t$.

### 3.4. Operatory relacja-strumień

Operatory relacja-strumień generują na podstawie relacji $R$ strumień $S$ o takim samym schemacie. W każdej chwili $t$ strumień $S$ powinien być obliczany z elementów relacji $R$ aż do chwili $t$.

**Strumień elementów dodanych** `IStream` tworzy, na podstawie relacji $R$, strumień, który w chwili $t$ zawiera elementy postaci $(s,t)$, gdzie $s$ należy do zbioru elementów dodanych do relacji $R$ w chwili $t$.

**Przykład:**

Zapytanie CQL:

```
SELECT IStream(*)
FROM Strumien_pojazdow [Rows Unbounded]
WHERE rodzaj_pojazdu = "SOD"
```

przekształca strumień `Strumien_pojazdow` w relację operatorem `[Rows Unbounded]`, a następnie nakłada filtr, który przepuszcza tylko krotki dotyczące samochodów osobowych `rodzaj_pojazdu = "SOD"` i tworzy z nich operatorem `IStream` nowy strumień.
Jest to przykład filtrowania strumienia danych działający w czasie rzeczywistym, ponieważ dane napływają ciągłym strumieniem `Strumien_pojazdow`.

**Strumień elementów usuniętych** `DStream` tworzy, na podstawie relacji $R$, strumień, który w chwili $t$ zawiera elementy postaci $(s,t)$, gdzie $s$ należy do zbioru elementów usuniętych z relacji $R$ w chwili $t$.

**Przykład:**

Zapytanie CQL:

```
SELECT DStream(*)
FROM Strumien_pojazdow [Range 10 Minutes]
```

jest filtrem, który generuje na podstawie strumienia `Strumien_pojazdow` relację zawierającą pojazdy zarejestrowane w ciągu ostatnich 10 minut, a następnie przekształca ją w strumień operatorem `DStream`.

**Strumień elementów relacji** `RStream` tworzy na podstawie relacji $R$ strumień, który w chwili $t$ zawiera elementy postaci $(s,t)$, gdzie $s$ należy do relacji $R(t)$.

**Przykład:**

Zapytanie CQL:

```
SELECT RStream(*)
FROM Strumien_pojazdow [Now]
WHERE V_pojazdu > 50
```

generuje na podstawie strumienia `Strumien_pojazdow` relację zawierającą pojazdy rejestrowane na bieżąco `[Now]`. Następnie nakłada filtr, który przepuszcza tylko krotki dotyczące samochodów poruszających się z prędkością większą od 50 km/h `V_pojazdu>50` i przekształca ją w strumień operatorem `RStream`.

## 4. PODSUMOWANIE

Przestawione w artykule strumieniowe bazy danych są wykorzystywane między innymi do przetwarzania informacji w systemach, do których dane napływają w sposób ciągły. Systemy zarządzania bazami danych DSMS wywodzą się z relacyjnych systemów RDBMS i wykorzystują nowo opracowany język zapytań CQL, będący rozwinięciem standardowego, szeroko znanego i względnie łatwego do opanowania języka SQL.

Szybki rozwój tego modelu danych pozwala przypuszczać, że pojawią się wkrótce dobrze opracowane rozwiązania komercyjne systemów DSMS. Umożliwią one projektowanie specjalizowanych strumieniowych baz danych, które będzie można wykorzystać do monitorowania ruchu w sieciach transportowych.

Przykładem prototypowych systemów DSMS są: system STREAM opracowany na Uniwersytecie Stanford, system AURORA opracowany wspólnie przez Brandeis University, Brown University oraz Massachusetts Institute of Technology oraz system Linear Road Benchmark opracowany przez wymienione ośrodki naukowo-badawcze.

## Literatura

1. Arasu A., Babu S., Widom J.: The CQL Continous Query Language: Semantic foundations and query execution. Technical report, Stanford University Database Group, October 2003.
2. Arasu A., Widom J.: A Denotational Semantics for Continuous Queries over Streams and Relations. Technical Report, Mar. 2004.
3. Babcock B., Babu S., Datar M., Motwani R., Widom J.: Models and Issues in Data Stream Systems. Invited paper in Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS 2002), June 2002.
4. Babcock B., Datar M., Motwani R.: Load Shedding for Aggregation Queries over Data Streams. In Proc. of Intl. Conference on Data Engineering (ICDE 2004), March 2004.
5. Babu S., Widom J.: Continuous Queries over Data Streams. In SIGMOD Record, September 2001.
6. Babu S., Subramanian L., Widom J.: A Data Stream Management System for Network Traffic Management.
7. Beynon-Davies P.: Systemy baz danych. WNT, Warszawa 2000.
8. Chomicz P.: Hurtownie danych - współczesność, X Konferencja PLOUG Kościelisko, Październik 2004.
9. Date C.: Wprowadzenie do baz danych. WNT, Warszawa 1981.

10. Date C. J., Darwen H.: SQL. Omówienie standardu języka. WNT, Warszawa 2000.
11. Garcia-Molina H. Ullman J. D., Widom J.: Implementacja systemów baz danych. WNT, Warszawa 2003.
12. Karoń G., Żochowska R.: Zastosowanie strumieniowych baz danych w systemach transportowych. III Konferencja naukowo-techniczna nt.: Systemy Transportowe – Teoria i Praktyka, Katowice 7 września 2005.
13. Kline K., Kline D.: SQL. Almanach. Opis poleceń języka. „Helion", Gliwice 2002.
14. Lausen G., Vossen G.: Obiektowe bazy danych. Modele danych i języki. WNT, Warszawa 2000.
15. Motwani R., Widom J., i inni: Query Processing, Resource Management, and Approximation in a Data Stream Management System. In Proc. of the 2003 Conf. on Innovative Data Systems Research (CIDR), January 2003.
16. Petersen J.: Wprowadzenie do baz danych. „Helion", Gliwice 2003.
17. Srivastava U., Munagala K., Widom J.: Operator Placement for In-Network Query Processing In Proc. of the 2004 ACM Symp. on Principles of Database Systems December 2004.
18. Srivastava U., Widom J.: Flexible time management in data stream systems. In Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, pages 263-274, June 2004.
19. Thomas D., Motwani R.: Caching Queues in Memory Buffers In In Proc. of the Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2004), January 2004.
20. Todman Ch.: Projektowanie hurtowni danych. Zarządzanie kontaktami z klientami (CRM). WNT, Warszawa 2003.
21. Tucker P.A., Maier D., Sheard T., Fegaras L.: Exploiting punctuation semantics in continous data stream. IEEE Trans. on Knowledge and Data Engg. 2003.
22. Ullman J. D., Widom J.: Podstawowy wykład z systemów baz danych. WNT, Warszawa 2001.