Continuous Queries over Data Streams – Semantics and Implementation

Jürgen Krämer

Philipps-Universität Marburg
kraemerj@informatik.uni-marburg.de

Abstract: Der rapide technologische Fortschritt der vergangenen Jahre hat die Entwicklung einer neuen Klasse von Anwendungen begünstigt, die sich dadurch auszeichnen, dass enorme Datenmengen in Form von Datenströmen bereitgestellt und kontinuierlich verarbeitet werden müssen, um zeitnah wichtige Informationen und Kennzahlen zu ermitteln. Beispiele für Anwendungen finden sich in den unterschiedlichsten Bereichen, die sich vom Business Activity Monitoring über die zeitkritische Analyse von Sensordaten in der Fabrikautomation bis hin zur Trenderkennung in Börsenkursen erstrecken. Konventionelle Datenbanksysteme sind für die erforderliche kontinuierliche Anfrageverarbeitung, bei der die eintreffenden Daten möglichst direkt und ohne vollständige Zwischenspeicherung verarbeitet werden müssen, nicht ausgelegt. Das wesentliche Ziel dieser Dissertation besteht darin, eine solide theoretische und gleichzeitig praktisch umsetzbare Grundlage zur adäquaten Verarbeitung kontinuierlicher Anfragen auf Datenströmen bereitzustellen.

1 Einleitung

Die Menge an Daten, mit der wir im Alltag konfrontiert werden, ist in den vergangenen Jahren durch die rasante technologische Entwicklung, vor allem in den Bereichen der Sensorik, der mobilen Endgeräte, der lokalen und globalen Vernetzung sowie der stetig wachsenden Beliebtheit des Internets mit seinen Web-Applikationen, drastisch gestiegen. Heutige Unternehmen sehen sich daher in ihrer heterogenen Informationslandschaft zunehmend mit einer wahren Flut an Daten über ihre Geschäftsprozesse, ihre Kunden und ihr Marktumfeld gegenüber. Sei es etwa ein Finanzunternehmen, das pro Sekunde tausende von Marktdaten zu analysieren hat, eine Bank, die schnellstmöglich einen Kreditkartenbetrug anhand der Transaktionsdaten erkennen muss, oder ein produzierender Betrieb, der mittels einer zeitnahen Auswertung der Produktionsdaten anstrebt, seine Produktionsprozesse zu optimieren.

Die umrissene Klasse von Anwendungen stellt an die verarbeitenden Systeme hohe Anforderungen, denn die in Form von *Datenströmen* fortlaufend eintreffenden Informationen müssen kontinuierlich und möglichst zeitnah ausgewertet werden. Hierbei stellt die Verfügbarkeit der Daten nicht das eigentliche Problem dar. Vielmehr gilt es, aus der Datenflut (i) relevante Informationen zu identifizieren, (ii) Zusammenhänge, Muster und Trends zu erkennen, (iii) den Informationsgehalt durch die geeignete Verknüpfung von Daten an-

zureichern, (iv) Aggregate zu berechnen, und schließlich (v) die Analyseergebnisse flexibel und aktuell, z. B. in Form graphischer Dashboards oder Web-Services, bereitzustellen. Die Bestimmung solcher Leistungsindikatoren ermöglicht Entscheidungsträgern in Unternehmen, einerseits schnell und sachlich fundiert auf aktuelle Marktgegebenheiten zu reagieren und anderseits neue Potentiale zur Verbesserung der eigenen Geschäftsprozesse ausfindig zu machen.

Obwohl es bis zu einem gewissen Grad möglich ist, die eintreffenden Daten zunächst abzuspeichern, gestaltet sich die anschließende Suche in den über die Zeit angehäuften Datenbergen sowie deren Analyse äußerst schwierig und aufwändig. Traditionelle Datenbankmanagementsysteme (DBMS) wurden dafür konzipiert, große Mengen an Daten zu archivieren und komplexe Anfragen über diesen persistenten Datenbeständen zu berechnen. Obige Datenstromanwendungen erfordern jedoch eine *kontinuierliche Auswertung* von Anfragen über Datenströmen zur zeitnahen Umsetzung der Anwendungslogik. Da DBMS diese Art der Anfrageverarbeitung nicht adäquat unterstützen, haben sich in den letzten Jahren zahlreiche Forschergruppen mit dem Entwurf und der Implementierung einer neuen Generation von Datenverarbeitungssystemen, so genannten *Datenstrommanagementsystemen* (DSMS), befasst.

2 Datenströme und kontinuierliche Anfragen

Bevor auf die Unterschiede zwischen der Datenverarbeitung in konventionellen DBMS und der in neuartigen DSMS näher eingegangen wird, um relevante Forschungsfragestellungen zu identifizieren, soll zunächst das Datenstrommodell genauer dargelegt werden.

2.1 Datenstrommodell

Aus den oben genannten Beispielanwendungen lassen sich folgende Charakteristika von Datenströmen und spezifische Anforderungen an die Datenstromverarbeitung ableiten:

- Ein Datenstrom ist eine potentiell unbegrenzte Folge zeitbehafteter Datensätze, die einer aktiven Datenquelle entspringen. Ein einzelnes Element eines Datenstroms kann als *Ereignis* (engl: *Event*) aufgefasst werden, da es über einen Zeitstempel verfügt.
- Getrieben durch die Aktivität der Datenquelle treffen fortlaufend Elemente eines Datenstroms beim verarbeitenden System ein. Das verarbeitende System kann weder deren Reihenfolge noch die Übertragungsgeschwindigkeit beeinflussen. In vielen Applikationen variiert die Übertragungsgeschwindigkeit mit der Zeit.
- Jedes Element eines Datenstroms wird nur einmal von der zugrunde liegenden Datenquelle geliefert. Da die Elemente eines Datenstroms sequentiell abgegriffen werden müssen, kann auf Datenstromelemente aus der Vergangenheit nur zugegriffen

werden, falls diese zuvor explizit gespeichert wurden. Allerdings verbietet die unbeschränkte Größe eines Datenstroms eine vollständige Speicherung.

- Anfragen über Datenströmen sollten kontinuierlich ausgewertet werden und beim Eintreffen eines neuen Datenstromelements sofort die zugehörigen Ergebnisse liefern.

An dieser Stelle sei angemerkt, dass DSMS zusätzlich die Verknüpfung von Datenströmen mit archivierten Datenbeständen, wie etwa Datenbanken, unterstützen sollten.

2.2 Unterschiede zu Datenbanksystemen

Im Folgenden werden kurz die wesentlichen Unterschiede zwischen der Anfrageverarbeitung in traditionellen DBMS und den sich aus dem Datenstrommodell ergebenden Verarbeitungsanforderungen für DSMS dargestellt (siehe Abbildung 1).

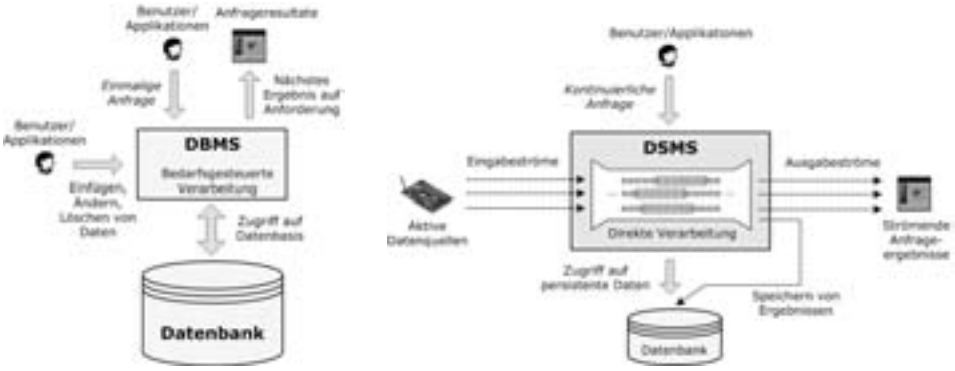

Abbildung 1: Vergleich der Datenverarbeitung in DBMS und DSMS

Datenquellen DBMS verarbeiten *passive*, *persistente* Datenquellen, nämlich auf dem Externspeicher abgelegte Relationen endlicher Größe. Im Gegensatz dazu, arbeiten DSMS auf *aktiven* Datenquellen, die ihre Daten in Form größenunbeschränkter, flüchtiger und zeitvariierender Datenströme bereitstellen. Da viele Datenstromanwendungen kurze Reaktionszeiten erfordern, findet die Datenverarbeitung primär im Hauptspeicher statt. Demgegenüber nutzen DBMS vorwiegend Externspeicherverfahren.

Anfragetypen Während DBMS *einmalige* Anfragen über persistenten Daten berechnen, werten DSMS *kontinuierliche* Anfragen über flüchtigen Daten aus. DBMS geben die Ergebnisse einer Anfrage bezüglich des gegenwärtigen Zustands der zugrunde liegenden

Relationen aus. Danach ist die Anfrage abgearbeitet und somit beendet. In DSMS hingegen laufen Anfragen über lange Zeit ab und produzieren kontinuierlich Ergebnisse, indem sie neu eintreffende Datenstromelemente verarbeiten. Hierbei bestimmt die jeweilige Applikation, wie lange eine Anfrage aktiv im System verbleibt.

Anfrageresultate Die Resultate für Anfragen in DBMS sind stets *exakt* bezüglich der zugrunde liegenden Daten. Anfragen in DSMS produzieren aus folgenden Gründen oftmals *approximative* Antworten: (i) Für eine Vielzahl kontinuierlicher Anfragen gilt, dass diese nicht mit endlichen Speicherressourcen berechnet werden können, wie zum Beispiel das kartesische Produkt zweier unbegrenzter Eingabeströme. (ii) Einige relationale Operatoren wie etwa die Gruppierung mit Aggregation sind *blockierend*, da sie zuerst die gesamte Eingabe konsumieren müssen, um ein Resultat herausgeben zu können. (iii) Die Elemente der Eingabeströme können schneller beim System eintreffen als das System diese verarbeiten kann. Im Allgemeinen gilt jedoch, dass approximative Antworten hoher Güte den meisten Anwendungen genügen.

Verarbeitungsprinzipien Die Datenverarbeitung in DBMS findet *bedarfsgetrieben* statt, d. h., die Berechnung wird durch das Stellen einer neuen Anfrage angestoßen. Hierbei werden Tupel *pull-basiert* – also nach und nach – aus Relationen mittels sequentieller oder indexbasierter Zugriffsmethoden eingelesen. DSMS setzen die *datengetriebene* Verarbeitung um. Die vollständige Antwort steht somit nicht unmittelbar nach dem Stellen einer Anfrage zur Verfügung, sondern wird inkrementell berechnet. Die Verarbeitung wird durch das Eintreffen neuer Elemente angestoßen und ist somit *push-basiert*. DBMS und DSMS unterscheiden sich zusätzlich in der Zugriffsart. Ohne explizite Zwischenspeicherung können DSMS die Daten lediglich sequentiell in deren Ankunftsreihenfolge zugreifen, während DBMS wahlfreien Zugriff auf die Daten haben.

Anfrageoptimierung Klassische DBMS optimieren Anfragen vor deren Ausführung auf Basis eines Kostenmodells und unter Einbeziehung gesammelter Metadaten über die beteiligten Relationen. Diese *statische* Optimierung genügt in DSMS im Allgemeinen nicht, da Anfragen für lange Zeit im System ablaufen, sich aber Stromcharakteristika, wie etwa die Datenrate oder Werteverteilungen, und die Systemlast zur Laufzeit ändern können. Um signifikanten Verschlechterungen der Systemperformanz entgegenzuwirken, sollten kontinuierliche Anfragen *dynamisch* optimiert werden. Dies bedeutet, dass je nach Notwendigkeit neben der initialen Optimierung zusätzliche Reoptimierungen zur Laufzeit stattfinden.

3 Forschungsfragestellungen im Datenstrommanagement

Trotz der im letzten Abschnitt genannten, mannigfaltigen Unterschiede zwischen DBMS und DSMS wurde in den vergangenen Jahren versucht, traditionelle DBMS mit spezieller Funktionalität zur Unterstützung kontinuierlicher Anfragen über Datenströmen anzu-

reichern. Prinzipiell ist dies zwar möglich, allerdings birgt dieser Ansatz zwei deutliche Nachteile in sich. Zum einen skalieren heutige DBMS nicht ausreichend bei einer sehr großen Anzahl von Triggern, zum anderen schränkt man die Anfrageoptimierung stark ein, da das System keine Kontrolle über die neu hinzugefügte Funktionalität besitzt.

Die geschickte Umsetzung des Paradigmenwechsel von einmaligen Anfragen über persistenten Daten hin zu kontinuierlichen Anfragen über flüchtigen Daten wirft verschiedenste Forschungsfragestellungen auf. Im Anschluss werden die wesentlichen Problemstellungen zur Konstruktion eines Datenstrommangementsystems kurz umrissen. Ausführlichere Informationen findet der interessierte Leser in den exzellenten Übersichtsartikeln: [BBD+02, GÖ03, SCZ05].

Spezifikation der Anwendungslogik Ein Großteil der heute im Einsatz befindlichen Software zur Umsetzung kontinuierlicher Anfragen über Datenströmen besteht aus Eigenentwicklungen. Neben hohen Entwicklungs- und Wartungskosten sind solche Lösungen zudem schlecht erweiterbar. Folglich stellt sich das Ändern der Anwendungslogik oft als äußerst aufwändig heraus. Eine Modifikation von Anfragen zur Laufzeit ist nahezu unmöglich. Anstatt Anfragen umständlich über eigens entwickelten Programmcode umzusetzen, wäre zu evaluieren, ob die Verwendung und Erweiterung einer etablierten Anfragesprache, wie z. B. SQL, die Entwicklung und Wartung der Anwendungslogik nicht maßgeblich vereinfachen würde.

Temporale Semantik Jegliche Formulierung von Anfragen macht nur dann Sinn, wenn die Anfragesemantik wohldefiniert ist. Um verlässliche und reproduzierbare Anfrageergebnisse zu garantieren, sollte die Semantik einer kontinuierlichen Anfrage nicht von Systeminterna abhängen. Darüber hinaus spielt der Aspekt der *Zeit* eine wesentliche Rolle für die Semantik einer kontinuierlichen Anfrage, da die ankommenden Daten zeitbehaftet sind. Aufgrund der systeminternen Übersetzung der textuellen Repräsentation einer Anfrage in einen Operatorplan, ergibt sich die Semantik der Anfrage implizit durch die der beteiligten Operatoren. Ein wichtiger Schritt besteht deshalb in der Bestimmung und Definition einer geeigneten *Operatoralgebra*. Dies wirft die Frage auf, inwiefern eine solche Algebra von der relationalen Algebra und deren wohlbekannter Semantik profitieren kann.

Online-Algorithmen Die Anforderungen des Datenstrommodells erfordern ein Umdenken bei der Implementierung von Anfragen in Hinblick auf folgende Aspekte: (i) Die für die Operatoren eines Anfrageplans verwendeten Algorithmen müssen eintreffende Elemente *on-the-fly*, d. h. direkt bei deren Ankunft, verarbeiten und erzeugen auf diese Weise über die Zeit fortlaufend Resultate. Hierbei sollte die amortisierte Verarbeitungsdauer pro Stromelement möglichst gering sein. (ii) Algorithmen, die aus Stromelementen abgeleitete Statusinformationen benötigen, wie zum Beispiel Join und Aggregation, müssen trotz unbeschränkter Eingabeströme mit *begrenztem Hauptspeicher* auskommen. Dieser Aspekt bedingt den Einsatz von Approximationsverfahren, um die Antwortgüte den verfügbaren Speicherressourcen anzupassen. Zusätzlich stellt sich die Frage, welche aussagekräftigen Garantien für die Ergebnisse eines Algorithmus gelten. (iii) Da manche Implementierun-

gen relationaler Operatoren wie etwa die Aggregation blockierend sind, aber häufig in der Praxis vorkommen, müssen entsprechende *nicht-blockierende* Verfahren entwickelt werden.

Adaptive Laufzeitumgebung Da DSMS Anfragen über lange Zeit im System verwalten und auswerten, müssen solche Systeme skalierbar ausgelegt sein. Außerdem können die hochgradig dynamischen und nicht vorhersehbaren Rahmenbedingungen vieler Datenstromanwendungen zusätzliche Anpassungen des System zur Laufzeit erforderlich machen. So sollten Mechanismen zur Lastreduktion bereitstehen, um einer Saturierung des Systems entgegenzuwirken und auf diese Weise die Stabilität und Verfügbarkeit des Systems zu gewährleisten. Techniken zur Umverteilung von Ressourcen zur Laufzeit sowie zur Reoptimierung von Anfragen sollten herangezogen werden, um basierend auf kontinuierlich aktualisierten Laufzeitstatistiken die Systemleistung dynamisch zu verbessern. Generell sollte ein System stets bestrebt sein, die *Qualität* der Anfrageresultate zu maximieren.

4 Ergebnisse

Dieser Abschnitt präsentiert in Kürze die wesentlichen Beiträge der Dissertation im Kontext obiger Forschungsfragestellungen [Krä07]. Im Rahmen der Dissertation beziehen sich kontinuierliche Anfragen auf *gleitende Fenster* über den potentiell unbeschränkt großen Datenströmen, um die Ressourcenanforderungen bei der Anfrageverarbeitung zu begrenzen. Diese Approximationstechnik bietet zwei entscheidende Vorteile. Zum einen beziehen sich die Ergebnisse einer Anfrage stets auf die aktuellen Daten, die üblicherweise für die Anwendungen von höherer Relevanz sind, zum anderen lässt sich die Semantik einer Anfrage exakt spezifizieren und intuitiv begreifen.

4.1 Formulierung kontinuierlicher Anfragen

Um von den Vorteilen eines etablierten Standards zu profitieren und Entwicklern mit Datenbankkenntnissen den Einstieg in die Datenstromverarbeitung zu erleichtern, schlägt die Dissertation vor, SQL als Anfragesprache zu verwenden. Minimale Erweiterungen des SQL:2003 Standards reichen bereits zur Integration von Datenströmen mitsamt entsprechender Operatoren aus. Die Änderungen an der Grammatik beschränken sich im Wesentlichen auf die `FROM`-Klausel, die nun neben Relationen auch Datenströme als Eingabe erlaubt. Zusätzlich kann ein Fensterkonstrukt dem Bezeichner eines Datenstroms folgen. Die Fensterkonstrukte entsprechen denen der SQL:2003 OLAP-Funktionen. Genau wie in SQL können Anfragen auch ineinander geschachtelt werden und komplexe Subanfragen mit Aggregaten und Quantoren enthalten.

Beispiel: Als Eingabeströme liegen die Aktienkurse zweier Unternehmen AG1 und AG2 vor. Eine vereinfachte Pair-Trading-Anfrage eines Brokers könnte wie folgt lauten:

„Benachrichtige mich, wenn dass durchschnittliche Verhältnis der Kurse von AG1 und AG2 *bezüglich der letzten 10 Minuten* vom üblichen Verhältnis von 5 zu 3 um mehr als 10% abwich."

In unserer Anfragesprache ließe sich diese Anfrage folgendermaßen ausdrücken:

```
SELECT *
FROM (SELECT AVG(AG1.price/AG2.price) AS ratio
      FROM   AG1 WINDOW(RANGE 10 MINUTES),
             AG2 WINDOW(RANGE 10 MINUTES)) AS SUBQUERY
WHERE ABS(SUBQUERY.ratio - 5/3) > 0.1*5/3;
```

Das Beispiel verdeutlicht die Verwendung gleitender Zeitfenster und zeigt zudem, wie einfach und elegant sich mittels dieses deklarativen Ansatzes Anwendungslogik umsetzen lässt.

4.2 Semantik kontinuierlicher Anfragen

In Analogie zu klassischen Datenbanksystemen unterscheidet die Dissertation zwischen einer logischen und einer physischen Operatoralgebra. Die *logische Algebra* definiert die Semantik kontinuierlicher Anfragen, indem sie die Datenströme als temporale Multimengen repräsentiert und für jeden einzelnen Operator dessen Ausgabestrom eindeutig spezifiziert [KS05]. Die vorgestellte Datenstromalgebra enthält für jeden Operator der erweiterten relationalen Algebra, bis auf die Sortierung als inhärent blockierende Operation, ein entsprechendes Gegenstück. Darüber hinaus umfasst die Algebra verschiedene Fensteroperatoren, mit deren Hilfe sich beispielsweise zeitbasierte oder größenbasierte gleitende Fenster über den Datenströmen ausdrücken lassen. Die daraus resultierende Einschränkung des Auswertungsbereichs von Anfragen auf endliche Teilmengen über den Eingabeströmen bewirkt, dass alle Operatoren trotz der generell unbeschränkten Eingaben mit endlichen Systemressourcen auskommen. Diese Funktionalität wurde in die anderen Operatoren nicht fest integriert, um einerseits Redundanz zu vermeiden und andererseits den einfachen Austausch von Fenstertypen zu ermöglichen. Die Kombination aus Fensteroperatoren mit den Datenstromvarianten der relationalen Operatoren ergibt die gewünschte Fenstersemantik. Die logische Algebra legt die temporalen Eigenschaften der Datenstromoperatoren offen und fungiert deswegen als wertvolles Werkzeug zum Erkennen und Validieren algebraischer Äquivalenzen.

4.3 Ausführung kontinuierlicher Anfragen

Die *physische Operatoralgebra* beschreibt die Implementierung einer kontinuierlichen Anfrage und setzt somit die durch die logische Algebra vorgegebene Semantik der Anfrage um. Physische Operatoren sind nicht-blockierende, datengetriebene Online-Algorithmen, die ein oder mehrere Datenströme als Eingabe haben und einen Datenstrom als Ausga-

be erzeugen. Die physische Algebra modelliert Datenströme als unendliche Folgen von Tupel-Zeitintervall-Paaren. Die Zeitintervalle repräsentieren die Gültigkeitsdauer des assoziierten Tupels gemäß den spezifizierten Fenstern. Die Online-Algorithmen für statusbehaftete Operatoren verwenden intern spezielle *Sweepline*-Datenstrukturen, die das effiziente Suchen und Löschen im Status sicherstellen. Mittels einer Performanzanalyse gestützt durch experimentelle Studien belegt die Dissertation, dass dieser neuartige Zeitintervall-Ansatz für einen Großteil der Anfragen konkurrierenden Ansätzen deutlich überlegen ist.

Neben der Erstellung von Anfrageplänen auf Basis der physischen Operatoralgebra widmet sich die Dissertation gleichermaßen architekturellen Gesichtspunkten einer adaptiven und skalierbaren *Laufzeitumgebung*, die Komponenten wie die Ablaufsteuerung, den Speichermanager und den Anfrageoptimierer umfasst (siehe Abbildung 2). Die in der Dissertation entwickelten Konzepte und Verfahren bilden den Kern der Softwareinfrastruktur *PIPES* (*Public Infrastructure for Processing and Exploring Streams*) [KS04, CHK+06]. Da es nahezu unmöglich ist, ein allgemeines und gleichzeitig performantes System für beliebige Datenstromanwendungen zu entwickeln, verfolgt PIPES ein Baukastenprinzip, das durch flexible Rahmenwerke für die einzelnen Komponenten den maßgeschneiderten Entwurf eines hocheffizienten Systems, zugeschnitten auf die jeweiligen Anforderungen einer Datenstromanwendung, ermöglicht.

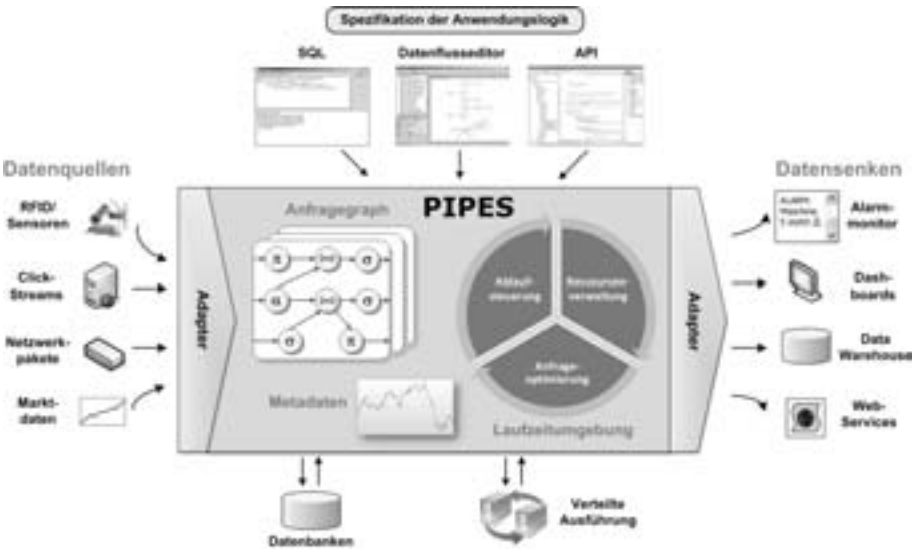

Abbildung 2: Architektureller Überblick der Softwareinfrastruktur PIPES

Für das *adaptive Ressourcenmanagement* werden Techniken vorgestellt, die auf Anpassungen der Fenstergrößen und Zeitgranularitäten der Anfragen basieren [CKSV08]. Gegenüber vergleichbaren, sampling-basierten *Load-Shedding*-Verfahren bieten unsere Techniken den Vorteil aussagekräftiger Garantien über die Ergebnisqualität und Semantik, wodurch sie nicht in Konflikt mit Reoptimierungen von Anfragen zur Laufzeit stehen.

4.4 Optimierung kontinuierlicher Anfragen

Die Dissertation erklärt die einzelnen Schritte von der Formulierung einer Anfrage über die Erzeugung von logischen und physischen Anfrageplänen bis hin zu deren Ausführung. Insbesondere gelang es, die aus Datenbanken bekannten *algebraischen Äquivalenzen* auf kontinuierliche Anfragen über Datenströmen zu übertragen, wodurch eine hervorragende Grundlage zur Anfrageoptimierung auf Basis der Multimengen-Schnappschuss-Äquivalenz [SJS01] geschaffen wurde. In Ergänzung hierzu stellt die Arbeit neue algebraische Äquivalenzen für die Fensteroperatoren sowie spezielle *physische Optimierungen* vor.

Ein Schwerpunkt der Dissertation liegt in der Entwicklung und Evaluation eines umfangreichen *Kostenmodells*, mit dem die Ressourcenallokation auf Operator-, Anfrage- und Systemebene unter Einbeziehung der Datenstromcharakteristika abgeschätzt werden kann [CKSV08]. Für adaptive Systemkomponenten ist ein solches Kostenmodell unerlässlich, da es erlaubt, die Effekte von Änderungen an Anfrageplänen im Voraus einzuschätzen. Der Ressourcenmanager nutzt das Kostenmodell, um festzulegen, in welchem Maße Fenstergrößen bzw. Zeitgranularitäten modifiziert werden müssen, um den vorgegebenen Systemressourcen zu genügen. Der Anfrageoptimierer wählt auf Basis des Kostenmodells von einer Menge semantisch äquivalenter Anfragepläne den optimalen Ausführungsplan aus.

Im letzten Teil der Dissertation werden Verfahren zur dynamischen Anfrageoptimierung präsentiert [YKPS07]. Diese sogenannten *Planmigrationsverfahren* transformieren einen gegenwärtig ausgeführten Anfrageplan, der über die Zeit durch Änderungen in den Datenstromcharakteristika ineffizient wurde, sukzessive in einen neuen effizienteren Anfrageplan. Die vorgestellten Migrationstechniken erlauben somit das Ersetzen physischer Anfragepläne zur Laufzeit, ohne dabei die Auswertung der zugrunde liegenden Anfrage stoppen zu müssen.

5 Schlussbemerkungen

In einem gemeinsamen Projekt mit dem Hamburger Unternehmen Langner Communications AG wurde PIPES erfolgreich an eine industrielle Integrationsplattform für die Serienfertigung gekoppelt [CHK+06]. Diese Kopplung ermöglicht die kontinuierliche Analyse zeitkritischer Daten aus Produktionsprozessen über alle Ebenen der Automatisierungspyramide hinweg. Der kommerzielle Nutzen besteht in der verbesserten Transparenz der Produktionsprozesse, da wichtige Leistungsindikatoren flexibel und zeitnah aus der Flut an heterogenen Produktionsdaten bereitgestellt werden können. In ähnlicher Weise lassen sich die im Rahmen der Dissertation entwickelten Verfahren zur komplexen Analyse von Geschäftsprozessen einsetzen. Aufgrund der hohen praktischen Relevanz und der positiven Rückmeldungen auf internationalen Konferenzen sowie der CeBIT 2007 wurde die RTM Realtime Monitoring GmbH als Spinoff-Unternehmen der Philipps-Universität Marburg gegründet. Das Unternehmen ist der erste deutsche Anbieter maßgeschneiderter DSMS-Lösungen.

Literatur

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani und Jennifer Widom. Models and Issues in Data Stream Systems. In *Symp. on Principles of Database Systems (PODS)*, Seiten 1–16, 2002.

[CHK+06] Michael Cammert, Christoph Heinz, Jürgen Krämer, Tobias Riemenschneider, Maxim Schwarzkopf, Bernhard Seeger und Alexander Zeiss. Stream Processing in Production-to-Business Software. In *Proc. of the IEEE Int. Conf. on Data Engineering (ICDE)*, Seiten 168–169, 2006.

[CKSV08] Michael Cammert, Jürgen Krämer, Bernhard Seeger und Sonny Vaupel. A Cost-Based Approach to Adaptive Resource Management in Data Stream Systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(2):230–245, 2008.

[GÖ03] Lukasz Golab und M. Tamer Özsu. Issues in Data Stream Management. *SIGMOD Record*, 32(2):5–14, 2003.

[Krä07] Jürgen Krämer. *Continuous Queries over Data Streams – Semantics and Implementation*. Dissertation, Philipps-Universität Marburg, 2007.

[KS04] Jürgen Krämer und Bernhard Seeger. PIPES - A Public Infrastructure for Processing and Exploring Streams. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Seiten 925–926, 2004.

[KS05] Jürgen Krämer und Bernhard Seeger. A Temporal Foundation for Continuous Queries over Data Streams. In *Proc. of the Int. Conf. on Management of Data (COMAD)*, Seiten 70–82, 2005.

[SCZ05] Michael Stonebraker, Ugur Cetintemel und Stan Zdonik. The 8 Requirements of Real-time Stream Processing. *SIGMOD Record*, 34(4):42–47, 2005.

[SJS01] Giedrius Slivinskas, Christian S. Jensen und Richard T. Snodgrass. A Foundation for Conventional and Temporal Query Optimization Addressing Duplicates and Ordering. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(1):21–49, 2001.

[YKPS07] Yin Yang, Jürgen Krämer, Dimitris Papadias und Bernhard Seeger. HybMig: A Hybrid Approach to Dynamic Plan Migration for Continuous Queries. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(3):398–411, 2007.

Jürgen Krämer wurde am 28. Juni 1977 in Alsfeld geboren. Ab 1997 studierte er an der Philipps-Universität Marburg Informatik mit Nebenfach Physik und erhielt 2003 sein Diplom. Während seines Studiums war er nebenberuflich als Softwareentwickler tätig. Danach arbeitete er in der Arbeitsgruppe Datenbanksysteme als wissenschaftlicher Mitarbeiter im Rahmen eines Projekts der Deutschen Forschungsgemeinschaft (DFG) zum Thema „Anfrageverarbeitung aktiver Datenströme“ unter Leitung von Prof. Dr. Bernhard Seeger. Seine Dissertation wurde mit dem Wissenschaftspreis der Industrie- und Handelskammer Kassel ausgezeichnet. Nach seiner Promotion 2007 gründete er gemeinsam mit Kollegen die RTM Realtime Monitoring GmbH, in der er die Position des Geschäftsführers übernahm.