Datenströme im Kontext des Verkehrsmanagements*

Michael Cammert Christoph Heinz Jürgen Krämer
Bernhard Seeger
Fachbereich Mathematik und Informatik
Universität Marburg
35032 Marburg, Deutschland
{cammert, heinzch, kraemerj, seeger}@informatik.uni-marburg.de

Abstract: Anhand mobiler Objekte im Verkehrsmanagement diskutieren wir die Verarbeitung der dabei entstehenden Datenströme. Wir skizzieren typische Anforderungen an Datenverarbeitungssysteme und begründen, warum traditionelle Datenbankmanagementsysteme im Kontext von Datenströmen weniger geeignet sind. Vielmehr motivieren wir den Einsatz von Datenstrommanagementsystemen, wobei wir einen Überblick über allgemeine Problemstellungen und Ansätze bei der Verarbeitung von Datenströmen geben. Neben prototpyischen Systemen präsentieren wir unser bibliotheksorientiertes Rahmenwerk, mit dem sich Anfragen über Datenströmen adäquat formulieren und ausführen lassen. Im Ausblick erläutern wir die Konstruktion eines allgemeinen Datenstrommanagementsystems aus den Komponenten unseres Rahmenwerks, welches sich in einem weiteren Schritt auf die Anforderungen des Verkehrsmanagements anpassen ließe.

1 Einleitung

Die stetig wachsende Belastung des Straßennetzes erfordert eine adäquate Steuerung des Verkehrs. Dazu müssen Verkehrsdaten erfasst, übertragen, analysiert und verwaltet werden, was unter dem Begriff Verkehrsmanagement zusammengefasst wird. Die Daten können sowohl von stationären als auch von mobilen Sensoren geliefert werden. Beispielsweise registrieren stationäre Sensoren an Autobahnbrücken den aktuellen Verkehrsfluss, während mobile Sensoren fahrzeugspezifische Informationen, wie etwa die exakte Position eines Fahrzeugs, übermitteln.

Die initialen Ziele des Verkehrsmanagements liegen in einer effizienteren Ausnutzung der bestehenden Kapazität des Straßennetzes und einer Verlagerung der Verkehrsbelastung, um dem wachsenden Verkehrsaufkommen entgegenzukommen. Geeignete Verkehrsmanagementsysteme könnten dazu eine Vielzahl von Daten berücksichtigen, z. B. Routenpläne, Baustellendaten, Floating-Car-Daten oder Wetterprognosen, und so etwa einen Autofahrer bei Staus intelligent umleiten. Eine Studie zu den Wirkungspotentialen des Verkehrsma-

*Diese Arbeit wurde von der Deutschen Forschungsgemeinschaft (DFG) unter Projektnummer SE 553/4-1 gefördert.

nagements [Pr99] prognostiziert für das Jahr 2010 eine Steigerung der Streckenkapazität auf den Autobahnen um ca. 10% unter Verwendung geeigneter Steuerungsanlagen.

Im Kontext des Verkehrsmanagements fassen wir die Verkehrsteilnehmer als bewegliche Objekte auf. Die zu deren Erfassung verwendeten Sensoren senden ihre Daten autonom, wobei die Updaterate allerdings variieren kann. Hierbei entsteht ein Tradeoff zwischen den Verarbeitungskosten, die durch hohe Updateraten verursacht werden, und der Genauigkeit der aktuellen Verkehrserfassung. Idealerweise ließe sich die Updaterate der beteiligten Fahrzeuge bei Staugefahr erhöhen und im Gegenzug bei wenig befahrenen Streckenabschnitten senken.
Die erfassten Sensordaten werden im Allgemeinen in einem Datenbanksystem verwaltet, wodurch sämtliche Updates persistent gespeichert werden. Hieraus resultieren enorme Datenbestände, von denen meist nur ein Bruchteil für aktuelle Analysen herangezogen wird, da eine vollständige Auswertung der archivierten Daten zu aufwändig wäre. Das stellt den Einsatz von Datenbanksystemen zur Verwaltung von Sensordaten generell in Frage. Seit zirka zwei Jahren beschäftigen sich daher einige Forschergruppen mit dem Entwurf von Datenstrommanagementsystemen, die sich speziell für die Analyse und Verarbeitung von Daten eignen, die autonomen Datenquellen entspringen. In diesem Artikel geben wir zunächst einen Überblick über die in diesem Themengebiet grundsätzlich auftretenden Problemstellungen und Ansätze. Danach präsentieren wir PIPES (Public Infrastructure for Processing and Exploring Streams), unser Rahmenwerk zur Verarbeitung von Datenströmen, das auf einer Publish-/Subscribe-Architektur basiert.

Der Rest dieses Artikels ist wie folgt strukturiert: In Abschnitt 2 betrachten wir im Kontext des Verkehrsmanagements die Unterschiede zwischen der Datenstromverarbeitung und der traditionellen Vorgehensweise basierend auf Datenbanksystemen. Im Anschluss stellen wir in Abschnitt 3 unser Rahmenwerk PIPES zur Verarbeitung von Datenströmen vor. Abschließend fassen wir die vorgestellten Konzepte zusammen und geben einen Ausblick auf mögliche Anwendungsszenarien.

2 Datenströme und Verkehrsmanagement

Im Folgenden skizzieren wir Probleme, die bei einer Datenverarbeitung im Rahmen des Verkehrsmanagements mit einem traditionellen Datenbankmanagementsystem (DBMS) entstehen [WXCJ98] und zeigen, dass ein Datenstrommanagementsystem (DSMS) angemessener wäre. In Abbildung 1 werden die beiden Konzepte gegenübergestellt.

2.1 Anforderungen

Zunächst möchten wir zentrale Anforderungen bei der Datenverarbeitung im Kontext des Verkehrsmanagements herausarbeiten. Da permanent verschiedenste Sensordaten erfasst werden, ergeben sich potentiell unendliche Datenströme. Für eine dynamische Steuerung des Verkehrsflusses müssen diese Daten kontinuierlich ausgewertet werden. Daraus resul-

tieren einerseits permanente Anfragen, wie z. B. die Berechnung der aktuellen Verkehrsdichte eines Streckenabschnitts, und andererseits zeitlich begrenzte Anfragen, die etwa bei der Betreuung eines Verkehrsteilnehmers auf seiner Route erforderlich sind. Weiterhin muss ein System in der Lage sein, neue Anfragen entgegenzunehmen, adäquat zu integrieren und nach Ablauf wieder zu entfernen. Bei hoher Verkehrsdichte trifft folglich eine größere Zahl von zum Teil ähnlichen Anfragen auf eine stark erhöhte Menge von Sensordaten. Allerdings existieren auch Phasen geringerer Verkehrslast. Ein geeignetes System muss hinsichtlich dieser Laständerungen skalierbar und adaptiv ausgelegt sein. Insbesondere sollte das Zusammenspiel mehrerer Anfragen optimiert werden (Multi-Anfragen-Optimierung), weil diese typischerweise lange im System verweilen. Eine generelle Möglichkeit zur Reduktion der Updaterate besteht in der dezentralen Ausführung von Anfragen, falls die Sensoren über genügend Rechenkapazität verfügen [BGS01, CBB+03]. Neben der Quantität der Daten spielen ebenso deren Qualität und Verfügbarkeit eine Rolle. So muss ein System auch bei Ausfall oder Fehlfunktion von Sensoren sowie einer Unterbrechung bzw. Überlastung des Kommunikationsnetzwerks weiterhin funktionieren. Eine zusätzliche Anforderung besteht in der effektiven Unterstützung von orts- und zeitabhängigen Anfragen. Darüber hinaus müssen sämtliche Anfragen nahezu in Echtzeit bedient werden, was eine komplette Zwischenspeicherung der Daten für eine spätere Verarbeitung ausschließt. Vielmehr sollten diese direkt bei ihrer Ankunft vom System verarbeitet werden. Wenn die exakte Auswertung der Anfrage aus Komplexitäts- oder Performanzgründen nicht möglich ist, sollen approximative Antworten zur Laufzeit bereitgestellt werden. Weiterhin wird eine einfache und flexible Integration heterogener Datenquellen sowie eine Anbindung an bestehende Systeme angestrebt. Das Speichern historischer Daten in anderen Datenbanken, wie etwa das durchschnittliche Verkehrsaufkommen an einem Freitag Nachmittag, könnte die Auswertung ähnlicher Anfragen unterstützen.

Diese vielseitigen Anforderungen an ein System zur Datenstromverarbeitung im Kontext des Verkehrsmanagements verdeutlichen, dass die Faktoren Adaptivität, Flexibilität, Skalierbarkeit sowie die Unterstützung kontinuierlicher Anfragen eine fundamentale Rolle spielen.

2.2 Datenbankmanagementsysteme

Wir diskutieren im Folgenden die Eignung von DBMS im Kontext des Verkehrsmanagements. Ein DBMS gestattet ausschließlich Anfragen über den in seiner Datenbank persistent abgespeicherten Daten. Die Elemente eines zu verwaltenden Datenstroms müssen daher zunächst in diese Datenbank eingefügt werden. Anfragen werden somit bezüglich des Datenbestands zu einem festen Zeitpunkt einmalig beantwortet. Dadurch basieren aktuelle Resultate auf fortwährenden Wiederholungen derselben Anfrage. Dies kann sich, ebenso wie das explizite Einfügen eines neu eintreffenden Datenstromelements, als sehr aufwändig erweisen. [TGNO92]
Manche Komponenten eines DBMS sind nur sehr bedingt nützlich, wie beispielsweise das Transaktionsmanagement. Fehlende Funktionalität, die in Fremdsystemen bereitgestellt wird, ist dagegen schwierig an ein bestehendes DBMS anzubinden. Darüber hinaus

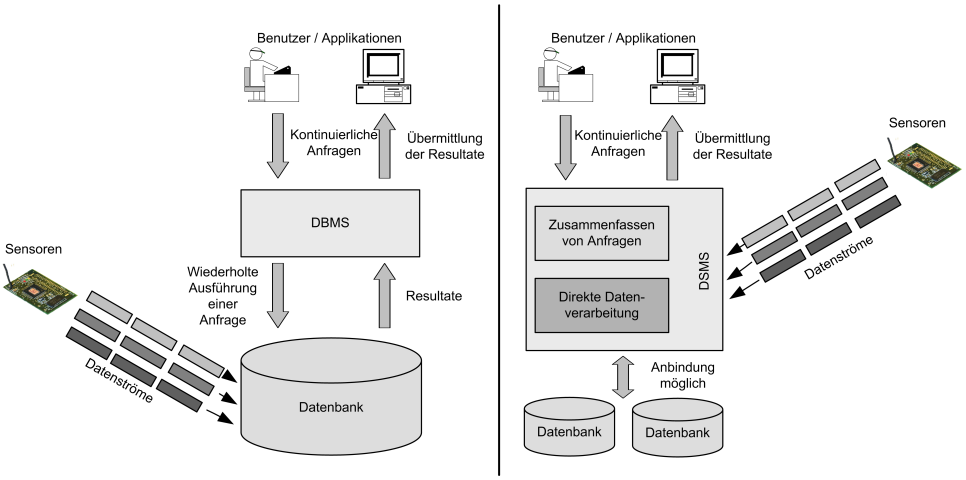

Abbildung 1: Datenverarbeitung in DBMS und DSMS

liefern DBMS stets exakte Antworten, d. h. eine Adaption im Falle einer zu hohen Systemlast ist nicht vorgesehen. Die wiederholte Auswertung der kontinuierlichen Anfragen erschwert zusätzlich die Multi-Anfragen-Optimierung, da sich die zu optimierende Anfragemenge fortlaufend ändert.
In Kombination mit obigen Anforderungen erweisen sich DBMS insbesondere hinsichtlich der Adaptivität als weniger geeignet für die Datenverarbeitung im Rahmen des Verkehrsmanagements.

2.3 Datenstrommanagementsysteme

Bezüglich der Charakteristika von Datenströmen treffen wir in Einklang mit [BBD+02] folgende Annahmen:

1. Die Verarbeitung der Anfragen erfolgt direkt beim Empfang der Daten, ohne zuvor den gesamten Datenstrom explizit zu speichern.
2. Die Ordnung auf den Datenströmen sowie deren Übertragungsgeschwindigkeit wird nicht durch die verarbeitenden Algorithmen, sondern primär durch die aktiven Datenquellen selbst bestimmt.
3. Die Datenströme sind in ihrer Größe potentiell unbeschränkt.
4. Jedes Element eines Datenstroms wird nur einmal vom Strom geliefert.
5. Änderungsoperationen auf Elementen eines Datenstroms sind nicht möglich. Neue Elemente können nur am Ende eines Datenstroms angehängt werden *(append-only)*.

Die im Rahmen des Verkehrsmanagements verarbeiteten Daten weisen diese Eigenschaften auf. Stationäre und mobile Sensoren senden kontinuierlich Daten in Form von Updates, welche wir nach obiger Definition als Elemente eines Datenstroms auffassen. Da ein Sensor stets aktuelle Daten übermittelt, tritt jedes Datum nur ein einziges Mal auf. Die Datenverarbeitung wiederum muss sich nach den Datenraten der aktiven Datenquellen richten. Ein wahlfreier Zugriff auf die Daten ist nur möglich, wenn diese zuvor archiviert wurden. Eine komplette Zwischenspeicherung und anschließende Auswertung der Datenströme ist ausgeschlossen, da diese einerseits in ihrer Größe nicht begrenzt und andererseits beim Verkehrsmanagement regelmäßig und möglichst frühzeitig Resultate erforderlich sind. Die Exploration der daraus resultierenden riesigen Datenarchive würde eine teure Aufgabe für ein Datenbanksystem darstellen. Daher sollte eine Online-Analyse der Datenströme direkt bei deren Ankunft erfolgen und, wenn überhaupt, nur ein Bruchteil der Daten zwischengespeichert werden, falls diese für künftige Anfragen erforderlich sind.
Neben historischen Anfragen spielen jedoch kontinuierliche Anfragen eine weitaus bedeutendere Rolle in DSMS, die hinsichtlich beider Anfragetypen skalierbar ausgelegt sein sollten. Durch die zu Grunde liegenden Datenströme und deren fluktuierende Datenraten müssen solche Systeme zusätzlich in einem hohen Maße adaptiv sein. Diese Adaptivität in Verbindung mit limitierten Ressourcen ermöglicht teilweise nur approximative Antworten zu einer Anfrage. Liefert eine Induktionsschleife beispielsweise einen Datensatz pro Fahrzeug, so steigt die Datenrate mit zunehmender Verkehrsdichte. Lassen die Werte eine erhöhte Staugefahr vermuten, so ist eine schnelle Benachrichtigung der betroffenen Verkehrsteilnehmer sicherzustellen, während die Bestimmung der exakten Zahl der erfassten Fahrzeuge weniger von Belang ist.
Während Anfragen in DBMS vor ihrer Ausführung statisch optimiert werden, ergeben sich im Kontext von DSMS neue Herausforderungen. Zur Steigerung der Performanz werden mehrere Anfragen gebündelt, so dass gemeinsame Teilanfragen nur einmal ausgewertet werden. Dabei muss auch das dynamische Einfügen und Löschen von Anfragen unterstützt werden. Zudem ist eine dynamische Laufzeitoptimierung dieser lange laufenden Anfragen notwendig.
Die für ein Verkehrsmanagement relevanten Daten liegen sowohl persistent als auch in Form eines Datenstroms vor. Straßenkarten sind beispielsweise in Datenbanken abgelegt, während die von mobilen oder stationären Sensoren gelieferten Informationen kontinuierlich gesendet werden. Tabelle 1 zeigt zusammenfassend die in diesem Abschnitt dargelegten Unterschiede zwischen herkömmlichen DBMS und DSMS.

2.4 Anfrageverarbeitung auf Datenströmen

Im Weiteren diskutieren wir gegenwärtige Konzepte und allgemeine Problemstellungen bei der Verarbeitung von Datenströmen.

Datenbankmanagementsystem (DBMS)	Datenstrommanagementsystem (DSMS)
• Passive Datenquellen	• Aktive Datenquellen
• Persistente Daten (Relationen)	• Flüchtige Daten (Datenströme)
• Externspeichernutzung	• Hauptspeichernutzung
• Wahlfreier Zugriff	• Sequentieller Zugriff
• historische Anfragen	• Kontinuierliche Anfragen
• Exakte Antworten	• Approximative Antworten
• Statische Optimierung	• Dynamische Optimierung

Tabelle 1: Unterschiedliche Anforderungen der Datenverarbeitungssysteme

2.4.1 Algebra

Anfragen in Datenbanksystemen werden typischerweise in einer deklarativen Sprache wie SQL formuliert und dann in einen Anfrageplan übersetzt, der im Wesentlichen aus Operatoren einer geeigneten Algebra besteht. Man könnte SQL um Funktionalität zur Verarbeitung von Datenströmen erweitern, wobei jedoch bestehende Funktionalität eventuell nicht mehr benötigt wird. Einfüge- und Änderungsoperationen auf Datenströmen werden obsolet, wohingegen sich approximative Anfragen bislang nicht spezifizieren lassen. Sinnvoll wäre folglich die Konzeption neuer Anfragesprachen oder die direkte Formulierung von Anfragen mittels einer GUI. Zentrales Problem [BBD+02] beider Ansätze ist die Definition einer geeigneten Algebra, die als Grundlage für die Verarbeitung von Datenströmen dient.
Neben der logischen Algebra, in der primär die Semantik der Operatoren festgelegt wird, gibt es eine physische Algebra, die i. A. verschiedene Implementierungen der Operatoren enthält. Durch die direkte Verarbeitung der Elemente eines Datenstroms ist die klassische bedarfsgesteuerte Verarbeitung mittels Iteratoren (ONC) nach [Gr93] nicht möglich. Aufgrund potentiell unbeschränkter Datenströme dürfen keine blockierenden physischen Operatoren eingesetzt werden. Darunter versteht man Operatoren, die ihre gesamte Eingabe konsumieren müssen, bevor ein Ergebnis geliefert werden kann. Es existieren jedoch logische Operatoren, wie beispielsweise die Minus- oder die Sortieroperation, die keine nicht-blockierende Implementierung erlauben. Dieses Problem lässt sich allerdings durch die Verwendung von über einen Datenstrom gleitenden Fenstern, inkrementeller Auswertung oder a priori bekannten Eigenschaften eines Datenstroms lösen. Zum Beispiel wäre es für eine aktuelle Stauprognose ausreichend, die Sensordaten der letzten Stunden zu betrachten.

2.4.2 Indexierung

Mit Indizes lassen sich Anfragen effektiv unterstützen. Typische Anfragen beim Verkehrsmanagement sind sowohl orts- als auch zeitgebunden. Eine auf Indexstrukturen mit Zeitfenstern basierende Anfrage wäre etwa: ”Wie hoch war die mittlere Verkehrsdichte auf den

Autobahnen im Großraum München bezogen auf die vergangenen vier Stunden?" Deswegen bieten sich mehrdimensionale Indexstrukturen wie der MVB-Baum [BGO+96] an, die effiziente Suchoperationen unterstützen. Neben persistenten Daten, wie etwa Karten- und Baustelleninformationen, lassen sich auch mobile Sensoren mit Indexstrukturen verwalten. Für die Sensordaten benötigt man performante Einfüge-, Lösch- und Änderungsoperationen, wobei insbesondere mengenbasierte Operatoren von Vorteil für die Performanz wären. Ebenso müssen gleitende Fenster adäquat unterstützt werden, was bei den bislang existierenden Ansätzen jedoch nicht der Fall ist.

2.4.3 Optimierung

Eine zentrale Komponente eines DBMS ist der Anfrageoptimierer, der bei der Anfrageübersetzung deskriptiv formulierte Anfragen in einen möglichst optimalen Ausführungsplan transformiert. In DSMS werden bei der statischen Optimierung Anfragen in einem Anfragegraphen unter Berücksichtigung übereinstimmender Teilanfragen zusammengefasst. Bei Anfragen auf Datenströmen benötigt man des Weiteren eine adaptive und dynamische Optimierung zur Laufzeit. Die Laufzeit einer kontinuierlichen Anfrage wiederum wird primär durch den Benutzer bestimmt, der diese bei einem Anfragekoordinator an- bzw. abmeldet. Bei Datenströmen geht man von einer Vielzahl von Anfragen mit oftmals überlappenden Teilanfragen aus, wodurch eine Multi-Anfragen-Optimierung erforderlich ist. So sind sicherlich mehrere Verkehrsteilnehmer an den aktuellen Stauinformationen für die Autobahn A8 auf dem Streckenabschnitt von München nach Stuttgart interessiert, während andere lediglich die Staus zwischen Augsburg und Ulm erfahren möchten.
Folglich müssen bei der Optimierung bereits ablaufende Anfragen ebenfalls berücksichtigt werden. Die traditionellen, auf der Schätzung von Zwischenergebnissen basierenden Kostenmodelle eignen sich nicht für potentiell unbegrenzte Datenströme. Stattdessen sollte eine Optimierung [VN02] die Ein- bzw. Ausgaberaten der Operatoren in Betracht ziehen. Es stellt sich zudem die Frage, wie eine dynamische Optimierung zur Laufzeit zu erreichen ist, da eine Re-Optimierung des Anfragegraphen aus Performanzgründen schwierig zu realisieren ist. Deswegen erscheinen partielle Umstrukturierungen sinnvoller. Bei einer dynamischen Optimierung sind darüber hinaus weitere Faktoren zu berücksichtigen wie etwa benutzerdefinierte Quality of Service (QoS) oder der Tradeoff zwischen Effizienz, Genauigkeit und Speicherverbrauch bei der Ausführung einer Anfrage.

2.4.4 Ressourcenmanagement

Eine weitere Problemstellung bei der Verarbeitung von Datenströmen ergibt sich bei der Zuteilung der begrenzten Systemressourcen CPU-Zeit, Haupt- und Externspeicher sowie Bandbreite. Zur Laufzeit müssen diese unter den gleichzeitig ablaufenden Anfragen verteilt werden. Der Ressourcenbedarf einer Anfrage kann dabei aufgrund der potentiell unendlich großen Datenströme unbeschränkt sein, wie z.B. bei der Berechnung des kartesischen Produktes. Da in einem DSMS immer limitierte Ressourcen potentiell unbeschränkten Ressourcenanforderungen gegenüberstehen, gestaltet sich die optimale Zuteilung der Ressourcen als äußerst schwierig. So lassen sich aufgrund des begrenzten Spei-

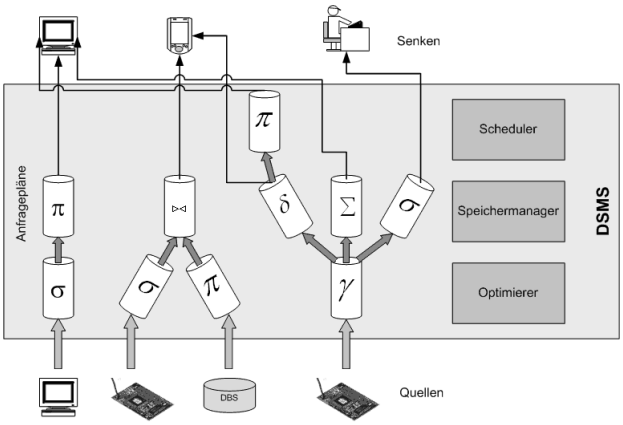

Abbildung 2: Anfrageverarbeitung im Rahmenwerk PIPES

chers häufig nur approximative Antworten zu einer gegebenen Anfrage bestimmen, über die idealerweise Qualitätsaussagen getroffen werden können.
Die Zuteilung der Rechenzeit geschieht mittels eines Schedulers, der hinsichtlich vieler gleichzeitig ablaufender Anfragen skalierbar ausgelegt sein muss. Weiterhin sollten dabei Abhängigkeiten zwischen der Ausführung der Operatoren sowie benutzerdefinierte QoS, wie z. B. Deadlines, berücksichtigt werden.
Die Mehr-Anfragen-Optimierung führt zur Einsparung von Ressourcen (*resource sharing*), weil gemeinsame Teilanfragen nur einmal ausgewertet werden. Ebenso könnte der gemeinsame Zugriff auf Datenstrukturen, wie z. B. Puffer oder Indexstrukturen, die Speicherausnutzung verbessern. Bei der Zuteilung der verfügbaren Ressourcen versucht man primär, die Genauigkeit und die Effizienz hinsichtlich der Anfragen zu optimieren. Hierbei spielen die Faktoren Adaptivität und Skalierbarkeit eine zentrale Rolle, da sowohl die Anfragemenge als auch die Datenraten zur Laufzeit variieren. Darüber hinaus sollte die Ressourcenverwaltung zusätzlich mit dem Optimierer gekoppelt sein.

3 Unser konkretes Rahmenwerk PIPES

In diesem Abschnitt stellen wir die Grundzüge unseres Rahmenwerks PIPES vor, mit dem sich Anfragen auf Datenströmen formulieren und ausführen lassen. Eine ausführliche Darstellung von PIPES findet sich in [CHK+03].

3.1 Anfragegraph

PIPES erlaubt die Integration beliebiger heterogener Datenquellen, was Anfragen auf Datenströmen und persistenten Daten ermöglicht. Diese Anfragen werden in Anfragepläne übersetzt, die in einem gerichteten Anfragegraphen zusammengefasst werden. Ein inhärenter Publish-Subscribe-Mechanismus erlaubt es, Anfragen zur Laufzeit dynamisch an- bzw. abzumelden.

3.1.1 Logische Operatoren

In Anlehnung an Datenflussgraphen unterscheiden wir innerhalb eines Anfragegraphen zwischen Quellen, Operatoren und Senken (siehe Abb. 2). Die Daten fließen dabei stets von den Quellen zu den Senken. Während eine Quelle Daten emittiert, werden diese durch eine Senke konsumiert und verarbeitet. In unserer Modellierung entspricht ein Operator sowohl einer Quelle als auch einer Senke, da er Daten empfängt, diese verarbeitet und anschließend überträgt. PIPES verfügt daher über zwei zentrale Schnittstellen `Source` und `Sink`. Diese werden von der Schnittstelle `Pipe`, die einen Operator modelliert, erweitert.

Sobald eine Quelle geöffnet ist, transferiert sie ihre Elemente an solche Senken, die über spezielle Methoden namens `subscribe` bzw. `unsubscribe` bei dieser an- bzw. abgemeldet werden können. Werden die Daten einer Quelle nicht mehr benötigt, so kann man sie schließen und verwendete Ressourcen freigeben. Eine Senke konsumiert und verarbeitet die Elemente der Quellen, auf denen sie abonniert ist. Sie wird informiert, falls eine ihrer Quellen versiegt. Für Optimierungszwecke lassen sich zusätzlich die Datenraten eines Knotens, d. h. die Anzahl der Elemente, die pro Sekunde in einem Datenstrom fließen, abfragen. Außerdem ist möglich, Anfragen temporär zu deaktivieren und später ohne Neuübersetzung zu reaktivieren.
Initiale Quellen eines Anfragegraphen können in unserem Rahmenwerk zum Beispiel mobile Sensoren oder auch Datenbanksysteme sein, während terminale Senken konkrete Applikationen oder PDAs eines Endbenutzers repräsentieren können. Für deren leichte und erweiterbare Anbindung verfügt PIPES über vorgefertigte Klassen, die gemeinsame Basisfunktionalität transparent bereitstellen.

3.1.2 Physische Operatoren

Bei der Konzeption und Entwicklung des Rahmenwerks PIPES wurde die Java-Bibliothek XXL [CHKS03, BBD+01] nahtlos erweitert. Neben den oben vorgestellten abstrakten Modellierungen und Schnittstellen enthält PIPES eine Vielzahl generischer Operatoren, die auf beliebigen Objekten arbeiten und über Funktionen und Prädikate parametrisierbar sind. Das Blockieren von Operatoren kann bislang mittels gleitender Fenster umgangen werden. Zum Funktionsumfang von PIPES gehören diverse Implementierungen folgender Operatoren: Filter, Map, Join, Gruppierung, Vereinigung, Differenz, Duplikateliminierung, Online-Aggregation et cetera.

3.1.3 Komponenten

Aufbauend auf dem Anfragegraphen, der mittels eines Anfragekoordinators verwaltet wird, existieren in PIPES noch drei weitere Komponenten, die in Abb. 2 dargestellt sind. Der *Scheduler* übernimmt die Ablaufsteuerung im Anfragegraphen, indem er den einzelnen Knoten CPU-Zeit zuweist. Dabei werden QoS-Aspekte nach benutzerdefinierten Vorgaben berücksichtigt. Ein prioritätsbasierter, adaptiver *Speichermanager* verwaltet die zur Verfügung stehenden Haupt- und Externspeicherressourcen und erlaubt deren Umverteilung zur Laufzeit. Der *Optimierer* in PIPES beinhaltet eine statische und eine dynamische Komponente. Während bei der statischen Optimierung neue Anfragepläne in den laufenden Anfragegraphen geeignet integriert werden, initiiert die dynamische Optimierung eine partielle Re-Optimierung des bestehenden Anfragegraphen.

Basierend auf den vorgestellten Schnittstellen, physischen Operatoren und Komponenten kann man auf einfache Weise ein DSMS entwickeln, das den speziellen Anforderungen eines Anwendungsszenarios gerecht wird. Wegen der generischen Konzeption muss dazu lediglich die anwendungsspezifische Zusatzfunktionalität implementiert werden.

4 Vergleichbare Projekte

Dieser Abschnitt gibt zunächst einen Überblick über verwandte Ansätze und stellt dann aktuelle Projekte vor, die Prototypen von DSMS entwickeln.

Das *Alert* System [SPAM91] benutzt ein konventionelles DBMS mit event-basierten Triggern zur Auswertung kontinuierlicher Anfragen. *Tukwila* [IFF+99] dient in erster Linie der Datenintegration verteilter, autonomer Datenquellen. Es realisiert eine ereignisbasierte Kommunikation der einzelnen Operatoren, vergleichbar mit der Funktionalität von Triggern, und enthält erste Ansätze zur adaptiven Re-Optimierung.
NiagaraCQ [CDTW00] überwacht und filtert kontinuierlich über ein Wide-Area-Netzwerk (WAN) verteilte, persistente XML-Datenbestände. Das System verfügt über eine Gruppierungsstrategie für Anfragen, die das inkrementelle Aufsetzen neuer Anfragen auf bestehende Materialisierungen unterstützt.
Das *Aurora* Projekt [CCC+02] befasst sich mit der Entwicklung eines Echtzeitsystems zur Überwachung von Datenströmen. Zentraler Bestandteil sind QoS, für die Techniken wie das Löschen von Elementen (load shedding) und speicheradaptives Operatorscheduling eingesetzt werden. Aurora unterstützt *resource sharing* durch die Verwendung von speziellen Queues. Das Aufsetzen mehrerer Anfragen wird an speziellen Verbindungspunkten, sog. *connection points*, ermöglicht. Diese erlauben es neu angemeldeten Operatoren, initial auf gepufferte historische Daten zurückzugreifen, bevor sie mit den aktuellen Daten des Datenstroms versorgt werden.
STREAM [MWA+03] ist ein zentrales DSMS, das auf dem relationalen Datenmodell basiert. Anfragen, die unter anderem kontinuierlich sein können, werden in einer von SQL abgeleiteten deklarativen Anfragesprache formuliert. Um die Ressourcenanforderungen zu reduzieren, fasst STREAM die Daten in Synopsen zusammen und stellt einen globalen Scheduler bereit, der den Speicherverbrauch der Queues minimiert. Letztere werden zur

Kommunikation der Operatoren verwendet. Des Weiteren wird der Join zwischen Datenströmen und bestehenden Relationen betrachtet.
Das *TelegraphCQ* System [CCD+03] vereinigt bisherige Arbeiten aus Fjords, Eddies und PSoup. Das *Fjords* [MF02] API definiert eine Menge von Operatoren zur Auswertung von Sensordaten, die sowohl bedarfs- als auch datengesteuert eingesetzt werden können. Allerdings sind diese nicht adaptiv. *Eddies* [AH00] sind spezielle, zentrale Operatoren, die mit sämtlichen Operatoren eines Anfrageplans verbunden sind. Ein Eddy bestimmt dabei adaptiv, in welcher Reihenfolge ein Element die Operatoren durchläuft. *PSoup* [CF02] erweitert diesen Ansatz, indem Daten und Anfragen symmetrisch behandelt werden, wodurch zusätzlich historische Anfragen unterstützt werden.

5 Zusammenfassung und Ausblick

Die stetig wachsende Verkehrsbelastung erfordert leistungsfähige Verkehrsmanagementsysteme. Die fortlaufend anfallenden Daten werden durch stationäre und mobile Sensoren geliefert und üblicherweise mittels traditioneller DBMS verarbeitet. Da diese Daten allerdings in Form von Datenströmen vorliegen, empfiehlt sich stattdessen der Einsatz eines DSMS, dessen Vorzüge wir insbesondere im Hinblick auf Adaptivität herausgestellt haben. Neben generellen Problemstellungen und Lösungsansätzen der Datenstromverarbeitung haben wir unser Rahmenwerk PIPES präsentiert, das Anfragen sowohl über Datenströmen als auch persistenten Daten erlaubt. Dieses komponentenbasierte Rahmenwerk ermöglicht einem Entwickler einerseits, komplexere Bausteine zu formieren, und andererseits, eigene Funktionalität nahtlos zu integrieren. Ausgehend von unserem generischen, erweiterbaren und flexiblen Bibliotheksansatz lassen sich auf diese Weise anwendungsspezifische DSMS konstruieren. Dies bietet sich im Kontext des Verkehrsmanagements hervorragend an. So könnte man ein intelligentes Verkehrsmanagementsystem realisieren, das, ausgehend von Datenströmen stationärer und mobiler Sensoren, die aktuelle Verkehrsbelastung überwacht und steuernd eingreift. Dieser offene Ansatz ließe sich analog auf Datenströme übertragen, die von beliebigen mobilen Objekten, wie etwa Flugzeugen oder Schiffen, stammen.

Literatur

[AH00] Avnur, R. und Hellerstein, J. M.: Eddies: Continuously Adaptive Query Processing. In: *Proc. of the ACM SIGMOD*. S. 261–272. ACM Press. 2000.

[BBD+01] Bercken, J., Blohsfeld, B., Dittrich, J.-P., Krämer, J., Schäfer, T., Schneider, M., und Seeger, B.: XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries. In: *Proc. of the Conf. on Very Large Databases (VLDB)*. S. 39–48. 2001.

[BBD+02] Babcock, B., Babu, S., Datar, M., Motwani, R., und Widom, J.: Models and Issues in Data Stream Systems. In: *Symp. on Principles of Database Systems*. S. 1–16. 2002.

[BGO+96] Becker, B., Gschwind, S., Ohler, T., Seeger, B., und Widmayer, P.: An Asymptotically Optimal Multiversion B-Tree. *VLDB Journal*. 5(4):264–275. 1996.

[BGS01] Bonnet, P., Gehrke, J., und Seshadri, P.: Towards Sensor Database Systems. In: *Proc. of 2nd Int. Conf. on Mobile Data Management*. S. 3–14. 2001.

[CBB+03] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Çetintemel, U., Xing, Y., und Zdonik, S. B.: Scalable Distributed Stream Processing. In: *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*. 2003.

[CCC+02] Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., und Zdonik, S. B.: Monitoring Streams: A New Class of Data Management Applications. In: *Proc. of the Conf. on Very Large Databases (VLDB)*. S. 215–226. 2002.

[CCD+03] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., und Shah, M.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In: *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*. 2003.

[CDTW00] Chen, J., DeWitt, D., Tian, F., und Wang, Y.: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In: *Proc. of the ACM SIGMOD*. S. 379–390. 2000.

[CF02] Chandrasekaran, S. und Franklin, M. J.: Streaming Queries over Streaming Data. In: *Proc. of the Conf. on Very Large Databases (VLDB)*. 2002.

[CHK+03] Cammert, M., Heinz, C., Krämer, J., Markowetz, A., und Seeger, B.: PIPES: A Multi-Threaded Publish-Subscribe Architecture for Continuous Queries over Streaming Data Sources. Technical report. University of Marburg. 2003. CS-32 (submitted for publication).

[CHKS03] Cammert, M., Heinz, C., Krämer, J., und Seeger, B.: A Status Report on XXL - A Software Infrastructure for Efficient Query Processing. 26(2):12–18. 2003.

[Gr93] Graefe, G.: Query Evaluation Techniques for Large Databases. *ACM Computing Surveys*. 25(2):73–170. 1993.

[IFF+99] Ives, Z. G., Florescu, D., Friedman, M., Levy, A., und Weld, D. S.: An Adaptive Query Execution System for Data Integration. In: *Proc. of the ACM SIGMOD*. S. 299–310. 1999.

[MF02] Madden, S. und Franklin, M. J.: Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data. In: *Proc. of the IEEE Conference on Data Engineering*. 2002.

[MWA+03] Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olsten, C., Rosenstein, J., und Varma, R.: Query Processing, Resource Management, and Approximation in a Data Stream Management System. In: *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*. 2003.

[Pr99] Prognos AG. Wirkungspotentiale der Verkehrstelematik zur Verbesserung der Verkehrsinfrastruktur- und Verkehrsmittelnutzung. 1999. *Bundesministerium für Verkehr-, Bau- und Wohnungswesen* FE-Nr. 96.584/1999.

[SPAM91] Schreier, U., Pirahesh, H., Agrawal, R., und Mohan, C.: Alert: An Architecture for Transforming a Passive DBMS into an Active DBMS. In: *Proc. of the Conf. on Very Large Databases (VLDB)*. S. 469–478. 1991.

[TGNO92] Terry, D., Goldberg, D., Nichols, D., und Oki, B.: Continuous Queries over append-only Databases. In: *Proc. of the IEEE Conference on Data Engineering*. S. 321–330. 1992.

[VN02] Viglas, S. D. und Naughton, J. F.: Rate-Based Query Optimization for Streaming Information Sources. In: *Proc. of the ACM SIGMOD*. S. 37–48. 2002.

[WXCJ98] Wolfson, O., Xu, B., Chamberlain, S., und Jiang, L.: Moving Objects Databases: Issues and Solutions. In: *Statistical and Scientific Database Management*. S. 111–122. 1998.