Feingranulare Verarbeitung von XML-Strömen

Sven Schmidt, Dirk Habich, Wolfgang Lehner

Technische Universität Dresden
dbgroup@inf.tu-dresden.de

Abstract:
Ausgehend von einer Vielzahl von Quellen haben Daten oft einen transienten Charakter und werden in Form von Datenströmen disseminiert. Zur adäquaten Verarbeitung existieren sogenannte Datenstrom-Managementsysteme (DSMS), die in der Lage sind, in strombasierter Art und Weise vom Benutzer spezifizierte Anfragen bzgl. der Datenströme auszuwerten und die Ergebnisse kontinuierlich auszugeben. In diesem Beitrag wird gezeigt, dass anspruchsvolle Verarbeitungsoperationen auf hierarchisch strukturierten Datenströmen realisiert werden können. Dabei wird die Verwendung von XML im Kontext der Datenstrom-Managementsysteme motiviert und ein angepasstes Verarbeitungsmodell basierend auf XML-Creeks skizziert.

1 Motivation

Von vielen Anwendungen werden Daten in Form von Strömen erzeugt. Dabei wird die Reihenfolge der Elemente im Strom durch den Zeitpunkt ihrer Erzeugung bzw. Dissemination implizit festgelegt. Die Verarbeitung übernehmen sogenannte Datenstrom - Managementsysteme, wobei derzeit hauptsächlich Tupel als Elemente der Ströme betrachtet werden (homogene Datenströme). Die eigentliche Datenverarbeitung wird durch eine Reihe von Operatoren realisiert, welche nach Vorgabe der Anfragen eines oder mehrerer Nutzer zu teilweise komplexen Operatorgraphen oder Operatornetzwerken komponiert werden.

Oft liegen die Daten bereits in strukturierter Form (z.B. XML) vor oder werden zur Vermeidung von Redundanzen und zur adäquaten Repräsentation der Inhalte strukturiert erzeugt (heterogene Datenströme). Sollen diese Daten nun in Form eines Datenstromes kontinuierlich veröffentlicht werden und zur Auswertung von Benutzeranfragen dienen, dann ist eine Zerlegung der strukturierten Daten in Tupel und damit einhergehend ein Verzicht auf die Datenstruktur nicht zu rechtfertigen, da die Zerlegung in Redundanz und in Vorverarbeitungsaufwand resultieren würde. Tupelbasierte DSMS sind an dieser Stelle somit ungeeignet.

Da sich XML als *das* Format für den Datenaustausch etablieren konnte und eine große Anzahl von Datenquellen mit transienten Daten in Form von Strömen hantiert, erscheint eine Betrachtung von *strukturierten Datenströmen* und den daraus abgeleiteten Anforderungen an ein Datenstrom - Managementsystem notwendig und sinnvoll.

Dazu werden XML-Datenströme als ein Vertreter hierarchischer Ströme betrachtet. Es wird die Idee der aus tupelbasierten DSMS stammenden elementaren Operatoren, komponiert zu Operatorgraphen im Kontext der XML-Stromverarbeitung motiviert. Das Beispiel der Anwendung eines Gruppierungsoperators auf die Daten eines XML-Datenstromes wird die erarbeiteten Ideen und Konzepte bestätigen.

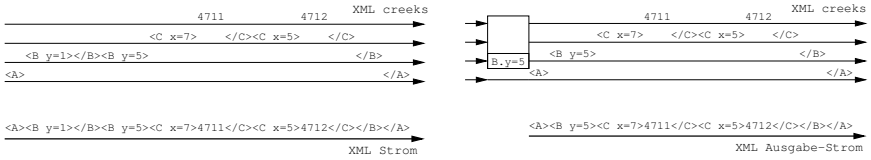
Verwandte Arbeiten: Im Bereich der Datenstrom-Managementsysteme werden ausgehend von einer tupelbasierten Stromverarbeitung verschiedene Richtungen verfolgt. Dies reicht von der Idee einer möglichst ressourcenoptimalen Anordnung der Operatoren ([ABB$^+$]) über die Spezifikation von geeigneten Strom-Anfragesprachen ([CFPR]) bis hin zum kontrollierten Verwerfen von Stromelementen in Überlastsituationen ([TÇZ$^+$]). Innovative, aber eingeschränkte Verarbeitung von XML-Daten im Strombetrieb erfolgt in [AF] durch einen automatenbasierten Ansatz. [ACGG$^+$] zeigt einen deterministischen Auswertemechanismus und [SGL] stellt darauf aufbauend einen Ansatz zur Realisierung einer einplanbaren XML-Dokumentenstrom-Verarbeitung im Kontext eines SDI-Systemes vor. In [OFB] wird ein auf Komponenten und Nachrichtenaustausch basierterender Filtermechanismus beschrieben, der sich auf die Auswertung von XPath-Ausdrücken beschränkt.

2 XML-Datenströme

Die Vielzahl von verschiedenen Datenquellen und die daraus resultierende Heterogenität bezüglich deren Eigenschaften erfordern innerhalb dieser Arbeit eine Abgrenzung der XML-Ströme von anderen Daten-Repräsentationen. Dabei sollen einem XML-Strom folgende Eigenschaften zugeordnet werden: XML-Datenströme sind von potentiell unbegrenzter Länge. Es gibt keinen expliziten Anfang, kein Ende und damit kein Root-Element. Ein XML-Strom besteht aus einer Menge von Knoten mit Name und Attributen. Verschachtelungen, wie sie dem Charakter der XML-Dokumente entsprechen sind integraler Bestandteil des Konzeptes der strukturierten Datenströme. Die Struktur des Datenstromes wird durch die Datenquelle bestimmt, welche zusätzlich zum Datenstrom ein Schema desselben veröffentlicht. Bezogen auf dieses Schema und bezogen auf die Struktur der stehenden Anfragen wird die Stromverarbeitung realisiert.

Ausgehend davon wird unter der XML-Stromverarbeitung das Abfragen, Filtern und Weiterleiten von XML-Strömen sowie die Ausführung komplexer statistischer Analysen verstanden. Die Anfrage an einen Datenstrom resultiert dabei wieder in einem Datenstrom, d.h. die jeweiligen Ergebnisse werden fortlaufend erzeugt und disseminiert. Mit der komponentenbasierten Stromverarbeitung wird außerdem eine flexible Abwägung des Ressourcenverbrauch möglich: Dieser korreliert mit der Möglichkeit, auf historische Daten (im Rahmen von Gruppierungs- und Aggregationsoperationen) zuzugreifen.

Beispiel: Die genannten Punkte sollen anhand eines aus Verkaufs-Einzeltransaktionen bestehenden XML-Stromes veranschaulicht werden. In einem Einzelhandesunternehmen sind eine Vielzahl von Kassen- und Hintergrundarbeitsplätzen durch IT-Infrastruktur na-



(a) Aufspaltung eines XML-Stromes (b) Anfragerepräsentation Filter

Abbildung 1: XML Creeks

hezu vollständig vernetzt. Transaktionsdaten, die von einem Kassenarbeitsplatz produziert werden, besitzen in ihrer Gesamtheit den Charakter eines Datenstromes, welcher kontinuierlich die Informationen wie Transaktions-ID, Transaktionsart, Warengruppe, verkaufte Artikel uvm. überträgt. Eine Reihe von „Just-in-Time“-Auswertungen wie die Bestimmung der Verkaufszeitpunkte eines Produktes, der Häufigkeit von Produktverkäufen oder der Umsätze pro Warengruppe sind auf einem derartig strukturierten Datenstrom denkbar.

3 Verarbeitungsmodell

Der wesentliche Unterschied des in diesem Kurzbeitrag beschriebenen Verarbeitungsmodells zu bisherigen Arbeiten besteht zum einen darin, dass bekannte und auf Stromsemantik übertragene Operatoren wie Selektion, Projektion, Join, Gruppierungs-/ Aggregationsoperator nur als Schablone für eine individuelle Konfiguration von speziellen Operatoren fungieren. Auf der anderen Seite zeichnet sich das Verarbeitungsmodell dadurch aus, dass eine mikrosopische Betrachtung der Verarbeitungsschritte im Kontext der hierarchischen Ströme vorgenommen wird. Benutzeranfragen, die im Stromsystem registriert sind, werden auf eine lineare Kette bzw. bei der gleichzeitigen Betrachtung mehrerer registrierter Anfragen, auf komplexe Graphstrukturen, mit den Knoten als Operatoren und Kanten zur Repräsentation des Datenflusses abgebildet. Um dem heterogenen und hierarchischen Charakter der XML-Ströme Rechnung zu tragen, erfolgt eine Erweiterung auf struktureller und operationaler Ebene:

Datenströme werden aus struktureller Sichtweise vor der eigentlichen Analyse in die hierarchischen Bestandteile nach den Ebenen aufgetrennt, so dass eine Reihe von individuellen Teilströmen entstehen, die im Folgenden als „Creeks“ („Bäche“) bezeichnet werden. Jeder Teilstrom beinhaltet dann die Elemente (Knoten) der jeweiligen XML-Ebene. Auf operationaler Ebene wird diese Aufspaltung dahingehend ausgenutzt, dass feingranulare Operatoren im Datenfluss einiger weniger Teilströme plaziert werden können, so dass flexibel eine individuelle Filterung bzw. Transformation des XML-Eingangsstromes (der XML-Eingangsströme) in einen XML-Ausgangsstrom realisiert werden kann. Der Vorteil der Aufteilung in Creeks impliziert jedoch auf Ebene der Operatoren einen zusätzliche Synchroniserungsaufwand. Abbildung 1(a) zeigt das Prinzip der Creeks als Ergebnis einer Aufspaltung eines hierarchisch strukturierten XML-Datenstromes.

In Streaming-Ansätzen, die sich mit XML-Datenverarbeitung beschäftigen, wird XPath

als die Anfragesprache positioniert, wobei im Strombetrieb die Navigationsrichtungen Parent/Ancestor (Zugriff auf Elemente in höheren Ebenen des XML-Dokumentes) und Preceeding/Following (Links-/Rechtsnavigation) ohne explizites Zwischenspeichern nicht sinnvoll sind. Als „Anfragesprache“ auf mikroskopischer Ebene (**nicht** dem Benutzer gegenüber) sind somit aus Blick der Synchronisation mit anderen Creeks nur Äquivalente für die Child- bzw. die Descendant-Achse zu suchen. Mit Blick auf die Funktionalität ist eine Spezifikation der Semantik der mikroskopischen Operatoren notwendig. Folgender Ausdruck repräsentiert, wie in Abbildung 1(b) dargestellt, eine Selektion auf Attributwerten des Elementes B, wobei „|“-Symbole zur Abtrennung der einzelnen Creeks verwendet werden: $| * |B.y = 5| + | + |$

Die „$*$“-Symbole bedeuten, dass keine Einschränkung (im Sinne einer Projektion!) vorliegt und durch einen Elementbezeichner anstelle des „$*$“ wird symbolisiert, dass nur die Elemente mit dem entsprechenden Namen „passieren“ dürfen. Durch ein „$+$“-Symbol in Creeks über einem bestimmten Operator wird ausgedrückt, dass die entsprechenden Creeks von dem Operator mit umspannt und beeinflusst werden. Entsprechend zeigt ein „$-$“-Symbol, dass ein Operator von Kontextinformationen tieferliegender Creeks abhängig ist.

Als weiteres Beispiel wird in Abbildung 2 die Funktionsweise eines Aggregationsoperators dargestellt: $| * | - |sum(C.x)| * |$
Der Aggregationsoperator kann bestimmte Aggregationsfunktionen - zum Beipsiel Summierung *sum* - auf Attributen von XML-Elementen durchführen. Die Größe der Aggregationsgruppe wird durch die Eltern-Elemente des tiefergelegenen Creeks bestimmt: Wird das Eltern-Element geschlossen, kann das Aggregat berechnet und als neues Element in den XML-Creek direkt nach den aggregierten Elementen eingefügt werden. Der Aggregationsoperator umspannt also zwei Creeks: Die Aggregationsfunktion selbst wird auf den höheren Creek angewandt; der darunterliegende Creek bestimmt den Beginn und das Ende der Aggregation. Die Elemente des ersten und des letzten Creeks werden unverändert weitergeleitet.

Für die Synchronisation zwischen den Strömen werden eingestreute Element-Open / -Close Nachrichten sowie ein Nummerierungsschema verwendet. Dadurch wird es möglich, zwischen dem Overhead durch zusätzliche Nachrichten und der Verzögerung der Creek-Elemente am Ausgang abzuwägen. Der Vorteil der Verarbeitung des XML-Stromes durch elementare Creek-Operatoren besteht in einer einfach zu realisierenden Einplanbarkeit der Operatorressourcen im Hinblick auf eine deterministische Verarbeitung, in der Möglichkeit, einzelne Teile des Operatorgraphen (einzelne Zwischenergebnisse) für mehrere stehende Anfragen zu nutzen und natürlich in der Flexibilität der Anfragegestaltung selbst.

4 Zusammenfassung

In diesem Beitrag wurde die Anwendung von DSMS-Techniken kombiniert mit geeigneten Betrachtungsweisen bezüglich der strukturierten Datenströme zur adäquaten Verarbeitung vom XML-Datenströmen angeregt. Dabei wurde exemplarisch eine Möglichkeit der

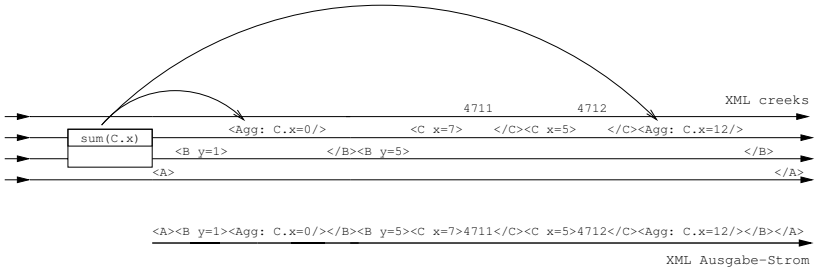


Abbildung 2: Anfragerepräsentation Gruppierung

Anfragespezifikation auf mikroskopischer Ebene sowie das Vorgehen bei der Anfrageverarbeitung skizziert.

Literaturverzeichnis

[ABB+] Arasu, A., Babcock, B., Babu, S., McAlister, J., und Widom, J.: Characterizing memory requirements for queries over continuous data streams. In: *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium, June 2002, Madison, Wisconsin, USA*.

[ACGG+] Avila-Campillo, I., Green, T., Gupta, A., Onizuka, M., Raven, D., und Suciu, D.: Xmltk: An xml toolkit for scalable xml stream processing. In: *Proc. of PLAN-X workshop, October 2002, Pittsburgh, PA, USA*.

[AF] Altinel, M. und Franklin, M. J.: Efficient filtering of xml documents for selective dissemination of information. In: *Proc of the 26th VLDB, September 2000, Cairo, Egypt*.

[CFPR] Cortes, C., Fisher, K., Pregibon, D., und Rogers, A.: Hancock: a language for extracting signatures from data streams. In: *Proc. of the 6th ACM SIGKDD, August 2000, Boston, MA, USA*.

[OFB] Olteanu, D., Furche, T., und Bry, F.: Evaluating complex queries against xml streams with polynomial combined complexity. In: *Proc. of 21st BNCOD, July 2004, Edinburgh, UK*.

[SGL] Schmidt, S., Gemulla, R., und Lehner, W.: Xml stream processing quality. In: *Proc. of the 1st XSym, September 2003, Berlin, Germany*.

[TÇZ+] Tatbul, N., Çetintemel, U., Zdonik, S. B., Cherniack, M., und Stonebraker, M.: Load shedding in a data stream manager. In: *Proc. of 29th VLDB, September 2003, Berlin, Germany*.