Oberseminar Datenbanken - Aktuelle Trends

Data Stream Management Systeme und Complex Event Processing

Frank Büchel

27. Juli 2015

In unserer Umwelt sind Rechner allgegenwärtig. Ob mobil, in Form eines Smartphones oder als Laptop werden sie immer stärker zur Kommunikation über das Internet verwendet. Industrielle Anlagen steuern jeden einzelnen Prozess automatisiert und benachrichtigen Teilsysteme oder Anwender über Zustände. Sensoren sind inzwischen billig und minimal, sodass sie in viele Geräten des Alltages integriert sind und stetig Messwerte liefern. Dabei entstehen kontinuierlich fließende Informationen, welche Grundlage für weiterführende Analysen darstellen. Für die Erfassung und Verwaltung solcher Datenströme sind spezialisierte Systeme notwendig, welche Anwendern durch verschiedene Techniken eine Auswahl von Operatoren zur Auswertung der enthaltenen Daten bieten. im Folgenden werden Anforderungen an solche Systeme, deren Unterschiede und Einsatzzwecke diskutiert.

1 Data Stream Management System

Ähnlich zu einem DBMS stellt ein solches System eine zentrale Verwaltungseinheit zur Erfassung und Verarbeitung von Datenströmen dar. Durch die Annahme, dass ein Strom, gegeben durch seine Begrifflichkeit, in seinem Umfang keine Begrenzung findet, unterscheidet sich ein DSMS von einem DBMS im Wesentlichen durch die Tatsache, dass eine Anwendung von Operatoren nicht auf einen festen Datenbestand geschehen kann. Daten treffen in Echtzeit ein, d.h. es existiert keine Möglichkeit zur Manipulation des Stromes und enthaltene Elemente können nicht in jedem Fall gespeichert werden. Weiterhin erreichen sie das System in beliebiger Ordnung und sind dabei häufig nur zu einem Zeitpunkt sichtbar. Derlei Systeme benötigen zur Abarbeitung nebenläufiger, kontinuierlicher Anfragen, Strategien hinsichtlich zeitkritischer Verarbeitung und optimaler Verwendung von Ressourcen.

2 Datenrepräsentation

Die Ströme sind meist als Relationen-Tupel, u.U. horizontal partitioniert, oder als Hierarchische Objekte, z.B. Transportprotokolle interpretierbar. Jedes Stromelement ist mit einem Zeitstempel oder einer ganzzahligen Sequenznummer versehen, entweder *direkt* durch die Quelle generiert, oder *indirekt* bei Registrierung durch das System. Dadurch können Teilmengen als physische bzw. logische Zeitfenster definiert werden, welche zur Einteilung des Stromes in verwaltbare Einheiten und als Approximationsschema zur Anfrageverbesserung dienen. Die Re-evaluation der Zeitfenster ge-

schieht *eager*, sobald ein neues Element eintrifft, oder *lazy* nach einem vorgegebenen Intervall und sind fester Bestandteil der Anfragesemantik.

3 Anfragen

Aufgrund der kontinuierlich eintreffenden Datenelemente müssen Anfragen auf diese ebenso kontinuierlich und stabil sein, z.B. bezüglich Änderungen von Stromraten. Ein *Selektionsoperator* filtert Elemente aus dem Strom, Aggregationsoperatoren erzeugen neue Informationen über den Strominhalt. Weiterhin müssen Ströme zusammenzufassen sein durch Multiplexing- (*UNION*, *GROUP BY*) und *JOIN*-Operatoren. Ähnlichkeitsanlyse, Pattern Matching und Forecasting, sowie Anwendung von Operationen auf Zeitfenster sind geforderter Bestandteil der Anfragesprachen.

Sei $A(Q,t)$ das Ergebnis einer kontinuierlichen Anfrage Q zum Zeitpunkt t, τ die aktuelle Zeit, und 0 die Startzeit. Dann unterscheidet sich die Semantik in monotone (1) und nicht-monotone (2) Anfragesprachen wie folgt [6]:

$$A(Q,\tau) = \bigcup_{t=1}^{\tau} (A(Q,\tau) - A(Q,t-1)) \cup A(Q,0) \quad (1)$$

$$A(Q,\tau) = \bigcup_{t=0}^{\tau} A(Q,t) \quad (2)$$

Also eine Unterscheidung in der Auslieferung des Ergebnisses in Form einer Tupelmenge oder als Ausgabestrom. Diese Semantiken finden sich in unterschiedlichen Ansätzen wie STREAM, Aurora und StreaQel wieder. Das System Aurora [2, 6], eine gemeinschaftliche Entwicklung der Universitäten Brandeis, Brown und dem MIT hingegen bietet dem Nutzer eine Schnittstelle in Form eines Datenfluss Graphen, wobei verschiedene Operatoren in Form von Boxen miteinander verknüpft werden. Die Anfragesprache CQL (Continuous Query Language) des STREAM (Standford Stream Data Manager) ist deklarativ und an SQL angelehnt. Datenstromelemente werden als Relationen-Tupel interpretiert, wobei jede Anfrage auf diese ein Zeitfenster spezifiziert. STREAM ermöglicht dem Nutzer die Resultate explizit monotonic oder non-monotonic auszuliefern, sowie eine Sample Rate (Stichprobe aus dem Strom) zu spezifizieren. Die folgende Beispielanfrage in CQL listet, aus einem Metadatenstrom eines Telekommunikationsdienstleisters, die Durchnittsdauer der Ferngespräche aller Nutzer auf, wobei jedoch nur die letzten 10 berücksichtigt werden sollen.

```
SELECT AVG(S.minutes)
FROM Calls S
[
   PARTITIONED BY S.customer_id
   ROWS 10 PRECEDING
   WHERE S.type = 'Long Distance'
]
```

Dabei erzeugt die Deklaration in eckigen Klammern ein Zeitfenster, dass bei eintreffen neuer Elemente im Strom, also *eager* re-evaluiert wird. Die Selektion über die Ferngespräche ist dazu im deklarierten Zeitfenster integriert, wodurch das Fenster ausschließlich jene Elemente enthält, für welche diese Selektionsbedingung erfüllt ist. Zur folgenden Anfrage wird ein Zeitfenster über alle eingehenden Tupel im Datenstrom erzeugt, die Selektion aller Ferngespräche erfolgt jedoch nachträglich auf der Ergebnismenge.

```
SELECT AVG(S.minutes)
FROM Calls S
[
   PARTITION BY S.customer_id
   ROWS 10 PRECEDING
]
WHERE S.type = 'Long Distance'
```

Die Anfragesprache StreaQuel verlangt für jeden Query die Angabe eines Anfragezeitraumes in Form einer *for*-Schleife und einem *WindowIs* Statement zur Spezifikaktion der Fenstergröße [6]:

```
for( t=NOW ; t < NOW+50 ; t++ )
WindowIs(S, t-4, t);
```

Dabei ist `NOW` der aktuelle Zeitpunkt und `S` der betreffende Strom. Die Anfrage erzeugt also ein Zeitfenster der Größe 5, und iteriert über 50 Zeiteinheiten vom aktuellen Zeitpunkt.

In diversen Systemen werden die Tupel als mehrdimensionale Arrays verwaltet, was Nachfolger-Operatoren auf Tupel-Attribute wie beispielsweise *PREV* in AQuery ermöglicht.

```
SELECT price - prev(price)
FROM Trades
WHERE company = 'IBM'
```

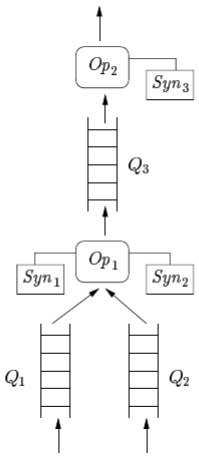

Abbildung 1: Schematische Darstellung eines Anfrageplans in *STREAM* [1]

4 Anfrageplanung

Die Planung der Anfragen bedarf einer Spezialisierung hinsichtlich der Kontinuität. Blockadefreiheit der Operatoren muss durch Techniken wie *Puctuations* gewährleistet sein, denn sie müssen Ergebnisse liefern können, bevor alle Daten gelesen wurden. *Punctuations* sind zusätzliche Information für Operatoren, indem die Aussagen über Folge-Tupel liefern. In einer Anfrage mit Sortiertierung des Ergebnis über ein Attribut `daynumber` kann der Operator beispielsweise bereits ein Ergebnis liefern, wenn die Information vorliegt, dass für alle folgenden Elemente gilt `daynumber > 10`. Aufgrund der Tatsache, dass einige Elemente im Strom mitunter nur einmal lesbar sind, sind *Single-Pass-Algorithmen* notwendig. Weiterhin ist zu berücksichtigen, dass in Abhängigkeit der Datenrate und Tupel-Schemata die Fenster-Re-Evaluierung unterschiedlich aufwändig ist, wie auch die Berechnung der Ergebnismenge verschieden aufwändig sein kann. Die Methodik *Batch-Processing* berücksichtigt schnelle re-evaluierung und langsame Berechnung des Ergebnisses, indem die Elemente gepuffert werden und erst zu einem Zeitpunkt mit geringer System-Auslastung ein Ergebnis erzeugt wird. Eine Antwort in Echtzeit ist durch diese Vorgehensweise ausgeschlossen. *Sampling* hingegen ermöglicht eine deklaration von Teilmengen aus Stromelementen, z.B. bei einer hohen Stromrate. Wenn eine Auswertung schnell erfolgt, kann eine Annäherung an das Ergebnis auch in Echtzeit erfolgen. *Synopsis* sind spezielle Datenstrukturen für Operatoren, die zur Beschleunigung der Fensteraktualisierung und Berechnung des Ergebnisses beitragen. Sie verwalten Kennzahlen, wie z.B. Element-Frequenzen, historische Minima bzw. Maxima und Erwartungswerte die zur Resultat-Approximierung verwendet werden können und für Aggregierungsfunktionen verwendet werden.

Eine Strategie zur Effizienten Unterstützung von nebenläufigen Queries sind *geteilte Anfragepläne* auf Gruppen von Strömen. Dabei entsteht eine Vereinigung der Resultate, die später über separate Selektionsbedingungen gefiltert werden. *Indexierte Anfragepläne* verlangen eine Verwaltung der Anfrage-Prädikate in Tabellen. Für jedes eintreffende Element erfolgt eine Attributextraktion und ein Lookup auf die Prädikattabelle. Bei einem Match wird die entsprechende Anfrage ausgeführt. Abbildung 4 zeigt das Schema eines Ausführungsplans einer Anfrage in *STREAM*

5 Complex Event Processing

CEP ist ein etabliertes Gebiet mit Wachstumsmarkt. In modernen Unternehmen und Einrichtungen ist, bedingt durch steigende Komplexität, die rechtlich, vertraglich oder betrieblich bedingte Überwachung von Informationssystemen verlangt. Dies Betrifft das Monitoring von industriellen Fertigungsanlagen und Geschäftsprozessen ebenso wie die Überwachung des Aktien und Optionshandels. Für letztgenannte Branche unterliegen Entwicklungen in diesem Gebiet einer strengen Geheimhaltung, weshalb Expertise bei einigen wenigen Unternehmen angesiedelt ist.

Komponenten oder Teile eines Systems erzeugen standardisierte Ereignisnachrichten, die beispielsweise eine Zustandsänderungen definieren. Aufgrund dieser Events kann eine Folgereaktion gefordert sein, wie Unterbrechung eines Prozesses bzw. Einleitung eines anschließenden oder alternativen. Dabei wird unterschieden zwischen der Erkennung von Mustern, und der Definition von bekannten Ereignissen als Muster im Ereignisstrom. Neben Datenextraktion, Komposition und Akkumulation sind Methodiken zur Identifizierung zeitlicher Zusammenhänge gefordert. *Deduktive Regeln* dienen der Generierung von neuen Ereignissen, wenn bekannte Ereignismuster im Ereignisstrom auftreten. *Reaktive Regeln* sind Anweisungen zur Reaktion auf eintreffende Ereignisnachrichten, wie Prozeduraufrufe oder Datenbankänderungen. Für die Formulierung dieser Regeln sind Operatoren zur Selektion, Sequenzierung, Negation (ein erwartetes Ereignis ist ausgefallen), Konjunktion und Komposition gefordert. Anfragesprachen müssen die Deklaration dieser Regeln Berücksichtigen. In diversen Projekten, wie beispielsweise $Xchange^{EQ}$ [5], werden Ansätze zu Anfragesprachen in CEP erforscht.

Nachrichten sind streng strukturiert, können von komplexer Gestalt sein und werden nicht ausschließlich in geschlossenen Systemen registriert und verwaltet. Aus diesem Grund werden Nachrichten in vielen Systemen als domänenspezifische Sprache in XML oder Derivaten konstruiert. Gängige Technologien wie beispielsweise der SAX-Parser (Simple API for XML [3]) ermöglichen ereignisbasierte Interpretation von XML in Programmen. Da er nicht auf dem Dokument-Objekt-Modell basiert und die Interpretation mit dem ersten Zeichen beginnt, operiert dieser Parser auf einem Zeichenstrom. Für Elemente mit möglichem Vorkommen im Strom werden Ereignis-Methoden konstruiert, deren Wirkungsweise keiner Vorgabe unterliegt. Die Software *Esper* [4] ist ein durch die GPL v2 Lizenz frei verfügbares Open Source CEP Produkt und bietet mit EPL (Event Processing Language) eine auf dem SQL-92 Standard aufsetzende Anfragesprache. Sie realisiert Operatoren für Pattern Matching, Erzeugung von Fenstern und ermöglicht JOINs auf Ereignissen.

6 Zusammenfassung

DSMS sind Systeme zur Verarbeitung von Datenströmen mit weitestgehend unstrukturiertem Inhalt. Sie bieten mit unterschiedlichen Ansätzen Anfragesprachen zur nebenläufigen, kontinuierlichen Analyse von Sensordaten sowie Netzwerkpaketen aller Art. Spezielle Operatoren selektieren, gruppieren Elemente in Datenströmen und liefern Ergebnismengen als Relationen oder in einigen Systemen auch wahlweise als Ergebnisstrom. Single-Pass-Algorithmen gewährleisten Resultate, auch wenn Datenelemente nur einmal im System erscheinen. Verschiedene Strategien wie Batch-Processing und Sampling dienen der Stabilität von Anfragen bzgl. verfügbarer Resourcen und Änderungen im Datenstrom. Für Methoden wie Deep Package Inspection, also der Analyse von Netzwerk-Protokollen, sind sie auch Schlüsseltechnologie zur Überwachung von Kommunikation in Telekommunikationsnetzen.

Complex Event Processing ist ein Fachgebiet, welches die Generierung und Analyse von Ereignisnachrichten zur Grundlage hat. Die Definition

bzw. Erkennung von Mustern in Ereignisströmen durch definierte *reaktive* und *deduktive Regeln* ist die Hauptanforderung in Ereignis getriebenen Systemen.

Literatur

[1] Brian Babcock u. a. “Models and Issues in Data Stream Systems”. In: *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS ’02. Madison, Wisconsin: ACM, 2002, S. 1–16. ISBN: 1-58113-507-6. DOI: 10.1145/543613.543615. URL: http://ilpubs.stanford.edu:8090/535/1/2002-19.pdf.

[2] *Data Stream Management System*. "Geändert: 08.11.2014, Zugriff: 30.06.2015". URL: https://de.wikipedia.org/wiki/Data_Stream_Management_System.

[3] *Esper*. "Geändert: 11.04.2015, Zugriff: 30.06.2015". URL: https://de.wikipedia.org/wiki/Simple_API_for_XML.

[4] *Esper*. "Geändert: 15.05.2015, Zugriff: 30.06.2015". URL: https://en.wikipedia.org/wiki/Esper_%28software%29.

[5] Michael Eckert und Francois Bry. *Complex Event Processing (CEP)*. Zugriff: 01.07.2015. Institut für Informatik, Ludwigs-Maximilians-Universität München, 2009. URL: http://www.en.pms.ifi.lmu.de/publications/PMS-FB/PMS-FB-2009-5/PMS-FB-2009-5-paper.pdf.

[6] Lukasz Golab und M. Tamer Özsu. *Issues in Data Stream Management*. Zugriff: 30.06.2015. 2003. URL: http://www.cs.virginia.edu/~son/cs851/stream/papers/p5-golab.pdf.