Datenströme und Complex Event Processing

Timo Michelsen

11. Mai 2010

Gliederung

Motivation

- Sensoren oder Sensornetzwerke liefern kontinuierlich Daten
- Einige Beispiele:
  - Sensoren in Windkraftanlagen
  - Analyse von Computernetzwerke
  - Verkehrsüberwachung
- Theoretisch unendliche Folge von Datenelementen
- Damit auch theoretisch unendlich großes Datenvolumen

Datenströme

- Eine unendliche Folge von Datenelementen wird als *Datenstrom* bezeichnet
- Eigenschaften:
  - Zeitbehaftet und zeitlich geordnet
  - Von einer aktiven Datenquelle selbständig gesendet
  - Jedes Datenelement wird nur einmal gesendet
  - Neue Datenelemente werden an das Ende des Datenstroms angehängt
  - Über die Reihenfolge und Ankunftsrate entscheidet die Datenquelle

Datenströme

Konventionelle Datenbankmanagementsysteme (DBMS) sind für die Verarbeitung der Datenströme ungeeignet

- Unendliche Datenvolumen nicht persistent speicherbar
- Daher spezielle Datenbanksysteme notwendig

Datenströme - Verarbeitung

- Datenelemente müssen direkt bei der Ankunft verarbeitet werden (*data-driven*)
- Das System hat keinen Einfluss auf Ankunftsrate und Reihenfolge
- Nur sequenzieller Zugriff möglich
- *One-Pass-Paradigma*: Für jedes Element muss explizit entschieden werden, ob es nach der Verarbeitung sofort verworfen oder für weitere Schritte gespeichert wird
- Ergebnisse auf Anfragen müssen möglichst in Echtzeit geliefert werden

Datenstrommanagementsysteme (DSMS)

Systeme, welche sich explizit mit der Verarbeitung von Datenströmen beschäftigen, werden als *Datenstrommanagementsysteme* (DSMS) bezeichnet.

- Beispiele: Odysseus, Aurora, Pipes

Unterschiede zwischen DBMS und DSMS (1/2)

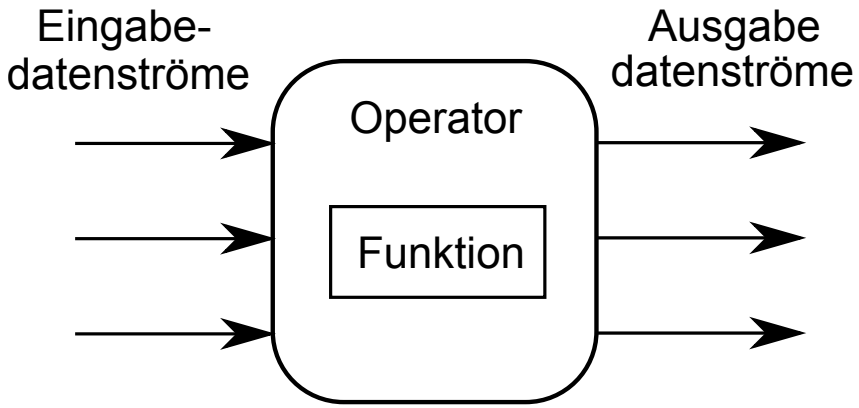
Aspekt	**DBMS**	**DSMS**
Datenquellen	passiv, persistent gespeichert	aktiv, liefert Datenströme
Datenzugriff	wahlfrei	sequenziell
Anfragetypen	Einmalanfragen, die über den aktuellen Zustand der Datenbank ausgewertet werden	kontinuierlich, bei Eintreffen neuer Datenelemente ausgewertet
Antworten	exakt	approximativ

Unterschiede zwischen DBMS und DSMS (2/2)

Aspekt	**DBMS**	**DSMS**
Verarbeitungsmethode	demand-driven: Verarbeitung beginnt mit Anfrage	data-driven: Verarbeitung beginnt mit Eintreffen neuer Datenelemente
Verarbeitungsstruktur	Operatorbaum	Operatorgraph
Anfrageoptimierung	einmalig, statisch, vor der Ausführung	statisch, zur Laufzeit Reoptimierungen

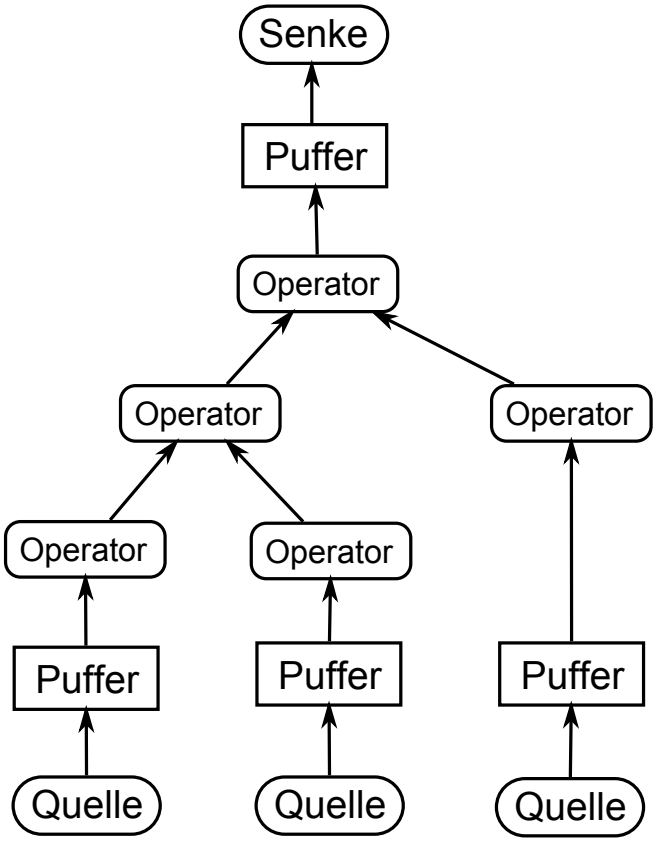
Operatoren

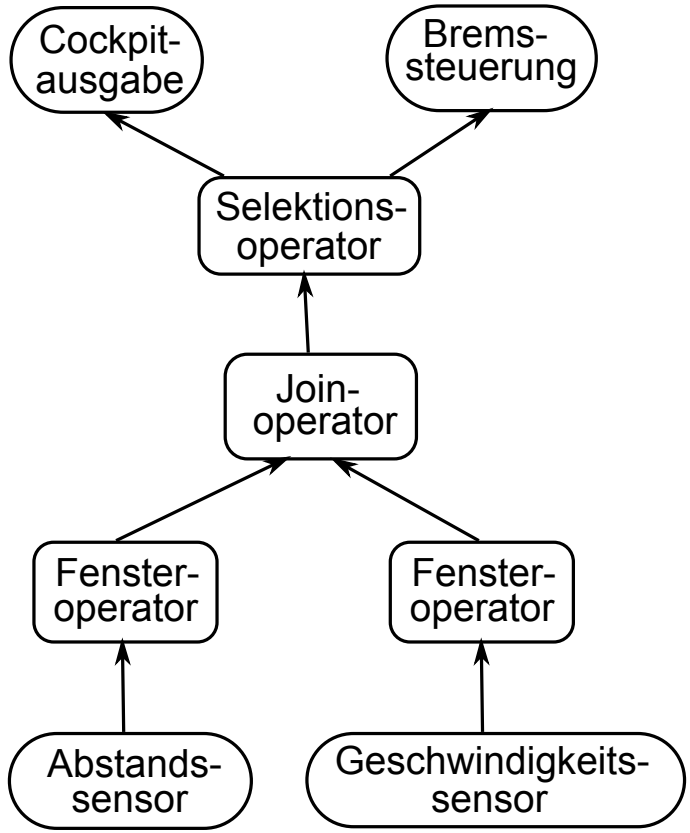
- Repräsentieren einzelne Verarbeitungsschritte
- Nehmen Eingabedatenströme entgegen
- Realisieren eine Funktion auf die Datenelemente
- Die Ausgaben ergeben Ausgabedatenströme



Operatorgraph (1/2)

Mehrere Operatoren können zu einen Operatorgraph verbunden werden:





Operatorgraph (2/2)

Typen von Operatoren

- *Quellen* besitzen nur Ausgabedatenströme (bspw. Sensoren)
- *Senken* besitzen nur Eingabedatenströme (bspw. Ausgaben)
- *Operatoren* sind gleichzeitig Quelle und Senke

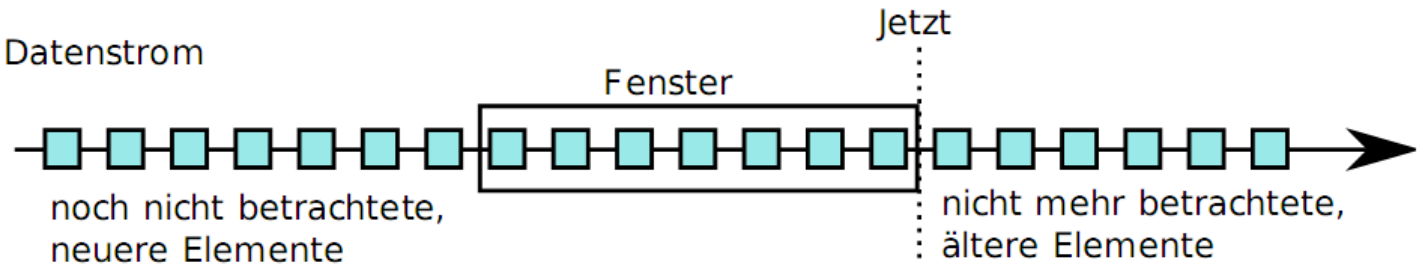
Problem mit Operatoren

Problem: Einige Operatoren können nur dann Ergebnisse liefern, wenn sie alle Datenelemente gelesen haben

- Beispiel: Aggregationsoperator MAX (Bestimmung eines Maximalwertes)
- In unendlichen Datenströmen unmöglich, es würde nie ein Ergebnis produziert werden
- Sogenannte *blockierende Operatoren*
- Blockierungen müssen in DSMS aufgelöst werden

Das Fenster-Prinzip

- Ein Fenster-Operator markiert auf einem unendlichen Datenstrom einen endlichen Bereich als gültig
- Andere Operatoren führen ihre Operationen nur in dieser Menge durch
- Blockierungen werden aufgeöst
- Jedoch werden Ergebnisse für einige Operatoren approximativ



Klassifikation der Fenster

Fenster können unterschiedlich definiert werden

- Fensterbreite: zeitenbasiert, elementbasiert oder prädikatenbasiert
- Beweglichkeit der zwei Fensterendpunkte:
  - beide fest: Landmark Window
  - beide beweglich: Sliding Window
  - eins fest, eins beweglich: Landmark Window
- Update-Intervall: Wann werden die Fensterendpunkte aktualisiert?

Umsetzungen (1/2)

Positiv-Negativ-Ansatz:

- Jedes neue Datenelement erhält einen (+)-Marker, das Element ist nun gültig
- Läuft die Gültigkeit ab, dann wird das gleiche Datenelement mit (-)-Marker erneut gesendet
- Alle Operatoren müssen die Marker betrachten und dementsprechend die Datenelemente verarbeiten oder ignorieren/löschen

Umsetzungen (2/2)

Intervall-Ansatz:

- Jedes Datenelement erhält ein Gültigkeitsintervall
- Besteht aus Start- und Endzeitstempel
- Trifft nun bei einem Operator ein Datenelement mit einem Startzeitstempel ein, welcher größer ist als der Endzeitstempel eines vorangegangenen Elements, so ist das vorangegangene Element ungültig

Complex Event Processing (1/3)

Es existieren Anwendungsgebiete, in denen Sensornetzwerke *Ereignisse* als Datenelemente liefern.

- Ereignisströme
- Ereignis = Objekt, welche eine bereits geschehene Aktivität in einem System beschreibt
- Beispiel: In einem Supermarkt wurde ein Produkt aus einem Regal entnommen

Complex Event Processing (2/3)

- Sensoren liefern „nur“ primitive Ereignisse
- Häufig werden jedoch semantisch höherwertige Ereignisse benötigt
  - Nicht direkt durch Sensoren erfassbar
  - Lassen sich aber durch Kombination von primitiven Ereignissen generieren
  - Dabei wird in Ereignisströmen nach Mustern gesucht
  - Gefundene komplexe Ereignisse werden dem Datenstrom zur Verarbeitung hinzugefügt
- In DSMS häufig als spezielle Operatoren umgesetzt

Complex Event Processing (3/3)

Komplexes Ereignis

- Ereignis, welches sich aus primitiven Ereignissen zusammensetzt
- Semantisch höherwertiger

Complex Event Processing (CEP)

- Der Prozess der Suche und Erstellung komplexer Ereignisse

Complex Event Processing – Beispiel

Ein Ladendiebstahl liegt genau dann vor, wenn ein Produkt aus einem Regal genommen, nicht bei der Kasse erkannt und es durch die Ausgangstür transportiert wurde.

- Komplexes Ereignis: Ladendiebstahl
- Primitive Ereignisse:
  - Produkt einem Regal entnommen
  - Nicht an der Kasse erkannt
  - Durch die Ausgangstür transportiert
- Die Reihenfolge der primitiven Ereignisse ist wichtig!

Zusammenfassung

- Datenströme sind theoretisch unendliche Sequenzen von Datenelementen
  - Können nicht von konventionellen DBMS verarbeitet werden
- DSMS sind spezielle Systeme zur Verarbeitung von Datenströmen
  - Unterscheiden sich erheblich von konventionellen DBMS
  - Operatorgraphen strukturieren die Datenverarbeitung
  - Fenster-Operatoren markieren einen Bereich im Datenstrom als gültig und lösen Blockierungen auf
- Im Complex Event Processing werden Datenelemente als Ereignisse verarbeitet
  - Erkennen und Generieren von komplexen, semantisch höherwertigen Ereignissen

Danke für ihre Aufmerksamkeit!