Complex Event Processing (CEP)

Michael Eckert · François Bry

Ereignisgesteuerte Informationssysteme benötigen eine systematische und automatische Verarbeitung von Ereignissen: Complex Event Processing (CEP).

CEP ist ein Sammelbegriff für Methoden, Techniken und Werkzeuge, um Ereignisse zu verarbeiten *während sie passieren*, also kontinuierlich und zeitnah. CEP leitet aus Ereignissen höheres, wertvolles Wissen in Form von sog. komplexen Ereignissen, d.h. Situationen die sich nur als Kombination mehrerer Ereignisse erkennen lassen, ab.

Anwendungsgebiete

Service Oriented Architecture (SOA), Event-Driven Architecture (EDA), kostengünstige Sensortechnik und die rechtlich, vertraglich oder betrieblich bedingte Überwachung von Informationssytemen haben die Anzahl der Ereignisse in Computersystemen in den letzten Jahren erheblich gesteigert. Einher damit geht der Bedarf, diese Ereignisse automatisch, systematisch und zeitnah zu verwalten und zu verarbeiten. Als wichtige Anwendungsgebiete von Complex Event Processing (CEP) sind die folgenden zu nennen:

Business Activity Monitoring überwacht Geschäftsprozesse und unternehmenskritische Ressourcen, um Probleme und Chancen frühzeitig zu erkennen. Dazu werden Ereignisse in so genannten Key Performance Indikatoren zusammengefasst, wie z.B. die durchschnittliche Dauer eines Prozesses.

Sensor-Netzwerke übermitteln Messdaten aus der Außenwelt, z.B. an Supervisory Control and Data Acquisition Systeme, die zur Überwachung industrieller Anlagen eingesetzt werden. Um Mess- und andere Fehler zu minimieren, sollen Daten von mehreren Sensoren kombiniert werden. Ferner muss oft aus einfachen numerischen Messungen (z.B. Temperatur, Rauch) eine höhere symbolische Situation (z.B. Brand) erschlossen werden.

Marktdaten wie z.B. Aktien- oder Rohstoffpreise können auch als Ereignisse aufgefasst werden. Sie müssen zeitnah und kontinuierlich analysiert werden um frühzeitig Trends zu erkennen und ggf. automatisch zu reagieren (Stichwort Algorithmic Trading).

In diesen Anwendungen sind zu erkennende Situationen und zugehörige Informationen über mehrere Ereignisse verteilt. Sie müssen durch CEP aus mehreren Ereignissen und deren Zusammenhängen erschlossen werden.

Arten von Complex Event Processing

Der Begriff Complex Event Processing wurde geprägt in [9]; allerdings hat CEP viele unabhängige Wurzeln in der Forschung, von ereignisorientierter Simulation über aktive Datenbanken und Netzwerkmanagement bis zu temporalem Schließen

DOI 10.1007/s00287-009-0329-6

Michael Eckert · François Bry
Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstr. 67
80538 München, Deutschland
E-Mail: michael.eckert@pms.ifi.lmu.de

*Vorschläge an Prof. Dr. Frank Puppe <puppe@informatik.uni-wuerzburg.de> oder Prof. Dr. Dieter Steinbauer <dieter.steinbauer@schufa.de>

Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter: www.ai-wuerzburg.de/as

in der KI. Erst in den letzten Jahren tritt CEP als eigenständiges Gebiet und wichtiger Trend in der Industrie auf, wie auch die Gründung der Event Processing Technical Society [6] Anfang 2008 zeigt.

Zu unterscheiden ist beim CEP, ob komplexe Ereignisse als a-priori bekannte Muster über Ströme von Ereignissen spezifiziert werden oder bisher unbekannte Muster in den Strömen als komplexe Ereignisse erkannt werden sollen. Im ersten Fall bieten spezielle Ereignisanfragesprachen eine komfortable Möglichkeit, komplexe Ereignisse zu spezifizieren und effizient zu erkennen. Im zweiten Fall werden maschinelles Lernen und Data Mining auf Ereignisse angewandt. Dieser Artikel konzentriert sich auf Ereignisanfragen, da sie das ausgereiftere Teilgebiet sind.

Abgrenzung zu anderen Gebieten

Die Erkennung von komplexen Ereignissen ist natürlich kein Selbstzweck; ein ereignisgesteuertes Informationssystem soll auf erkannte Ereignisse automatisch und sinnvoll reagieren. Zu typischen Reaktionen zählen Benachrichtigungen (z.B. an ein anderes System oder einen menschlichen Benutzer), einfache Aktionen (z.B. Kauf von Aktien, Aktivierung der Selbstlöschanlage), oder die Interaktion mit Geschäftsprozessen (z.B. Initiierung eines neuen Prozesses, Unterbrechung oder Abänderung eines laufenden Prozesses).

CEP steht daher in enger Verbindung mit anderen Themen, wie z.B. der Visualisierung von ereignisbezogenen Daten für menschliche Benutzer, message-based Middleware zur Weiterleitung von Nachrichten, Regelsystemen zur Spezifikation von reaktivem Verhalten (z.B. ECA Regeln und reaktive Logikprogrammierung) und Business Process Management (BPM). In diesem Artikel soll jedoch nur auf CEP im engeren Sinn der Erkennung von komplexen Ereignissen eingegangen werden.

Ereignisanfragen

Im Gegensatz zu Datenbankanfragen werden Ereignisanfragen kontinuierlich ausgewertet, während die Ereignisse passieren. Zwar arbeiten Datenbanken auch oft mit ereignisbezogenen Daten (z.B. historische Auflistung von Bestellungen). Doch sind die Anfragen dort einmalig und „spontan“ gegen eine endliche Datenmenge, nicht dauerhaft und „stehend“ gegen einen (konzeptuell) unendlichen Ereignisstrom wie beim CEP (vgl. Abb. 1).

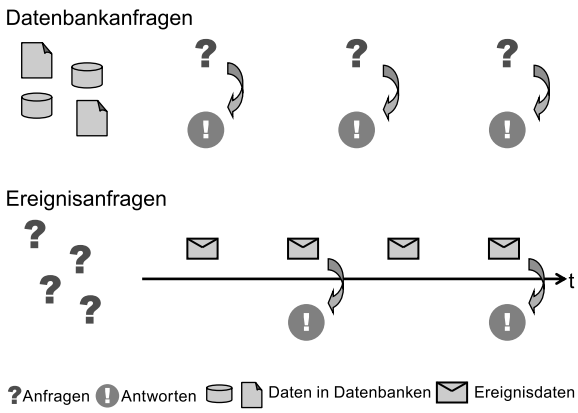

Abb. 1 Unterschied zwischen Datenbank- und Ereignisanfragen

Die Anforderungen an eine Ereignisanfragesprache lassen sich durch die folgenden vier Aspekte beschreiben:

Extraktion von Daten: Ereignisse enthalten Daten, die relevant dafür sind, ob und wie reagiert wird. Die Daten müssen für Bedingungen in Anfragen, für mögliche Reaktionen, zur Anreicherungen mit anderen Daten (z.B. Datenbanktabellen) oder zur Konstruktion neuer Ereignisse zur Verfügung stehen. Zunehmend werden Ereignisse als XML-Nachrichten dargestellt; die Daten können dann durchaus komplex strukturiert sein.

Komposition: Mehrere einzelne Ereignisse müssen verbunden werden können, so dass ihr gemeinsames, über die Zeit verteiltes Auftreten ein komplexes Ereignis ergibt. Diese Komposition muss oft datenbezogen sein (z.B. nur solche Ereignisse zusammensetzen, die den*selben* Kunden betreffen).

Zeitliche Zusammenhänge: Ereignisanfragen enthalten oft zeitliche Bedingungen, die ausdrücken, dass Ereignisse in einer gewissen Zeitspanne oder Reihenfolge passieren müssen. Ferner können auch andere Zusammenhänge, z.B. ursächlicher Art, eine Rolle spielen.

Akkumulation: Anfragen mit Negation (Fehlen eines Ereignisses) oder Aggregation von Ereignisdaten ergeben auf unendlichen Strömen wenig Sinn, da sie erst am Ende korrekt beantwortet werden könnten. Solche Anfragen können also nur gegen gewisse endliche Ausschnitte (oder „Fenster“) des Stroms gestellt werden, wo ihr Ergebnis wohldefiniert ist.

Ferner sind zwei Arten von Regeln wichtig: **Deduktive Regeln** definieren neue Ereignisse auf der Basis von Ereignisanfragen; sie sind vergleichbar mit Sichten in Datenbanken und haben keine Seiten-

effekte. Zu betonen ist, dass die deduktiven Regeln hier auf Ereignissen operieren, nicht auf Fakten (wie „traditionelle“ deduktive Regeln aus Logikprogrammierung und deduktiven Datenbanken). **Reaktive Regeln** [3] spezifizieren, wie auf (komplexe) Ereignisse reagiert wird, z.B. mit Datenbankänderungen oder Prozeduraufrufen.

Verbreitete Ereignisanfragesprachen

Im Wesentlichen sind derzeit drei Arten von Sprachen im Gebrauch, um Ereignisanfragen auszudrücken. Im Folgenden stellen wir die Grundideen der drei Sprachstile vor. Ein Ausblick auf ein aktuelles Forschungsprojekt zum Design einer Ereignisanfragesprache geben wir außerdem am Ende dieses Artikels. Die Ausführungen hier sind bewusst kurz und verallgemeinernd gehalten; wir verweisen auf Kapitel 3 von [5] für eine tiefergehende Diskussion und ausführlichere Literaturverweise.

Kompositionsoperatoren

Kompositionsoperatoren entstammen aktiven Datenbanksystemen [10]; neuere Systeme, wie z.B. Amit [1], sind jedoch unabhängig von einer Datenbank. Anfragen gegen einzelne Ereignisse werden hier mit verschiedenen Operatoren zu komplexen Ereignisanfragen zusammengesetzt. Typische Operatoren sind Konjunktion von Ereignissen (alle Ereignisse passieren, möglicherweise an unterschiedlichen Zeitpunkten), Sequenz (die Ereignisse passieren nacheinander), Negation in einer Sequenz (ein Ereignis passiert nicht in der Zeit zwischen zwei anderen Ereignissen). Durch Schachtelung lassen sich auch komplexere Anfragen ausdrücken.

Viele Sprachen erlauben Einschränkungen, welche Ereignisse für die Zusammensetzung eines komplexen Ereignisses betrachtet werden sollen. Mit Selektion kann so z.B. nur das erste oder letzte Ereignis eines bestimmten Typs ausgewählt werden. Konsumieren von Ereignissen verhindert deren Wiederverwendung in neuen komplexen Ereignissen, wenn sie bereits für eine komplexes Ereignis verwendet wurden.

Kompositionsoperatoren sind eine kompakte und intuitive Möglichkeit, komplexe Ereignisse zu spezifizieren und unterstützen temporale Zusammenhänge und Negation gut. Selektieren und Konsumieren von Ereignissen ist eine besondere Fähigkeit, die in anderen Ansätzen nicht zu finden ist. Allerdings gibt es manchmal versteckte Probleme mit dem intuitiven Verständnis von Operatoren, z.B. mehrere Varianten der Interpretation einer Sequenz (u.a. durchbrochen durch andere Ereignisse oder nicht). Ferner werden Daten in Ereignissen oft vernachlässigt, besonders im Hinblick auf Komposition und Aggregation.

Bisher verfolgen nur wenige CEP-Produkte den auf Kompositionsoperatoren basierenden Ansatz, darunter IBM Active Middleware Technology (Amit) und ruleCore.

Datenstrom-Anfragesprachen

Datenstrom-Anfragesprachen wie CQL [2] basieren auf der Datenbankanfragesprache SQL und haben folgendes Grundprinzip: Datenströme, die Ereignisse als Tupel enthalten, werden in Relationen umgewandelt. Auf diesen Relationen wird dann eine reguläre SQL-Anfrage ausgewertet und das Ergebnis (eine Relation) zurück in einen Datenstrom gewandelt. Konzeptionell wird dieser Vorgang an jedem Zeitpunkt einer vorgegeben diskreten Zeitachse durchgeführt. (Siehe jedoch [8] für Varianten.)

Die Umwandlung von Datenströmen in Relationen geschieht durch verschiedene Fensteroperationen wie z.B. „alle Ereignisse der letzten Stunde“ oder „die letzten 10 Ereignisse“. Bei der Umwandlung der Ergebnisrelation zurück in einen Datenstrom gibt es drei Möglichkeiten: nur Tupel, die im Vergleich zum vorherigen Ergebnis hinzugekommen sind, geben ein neues Ereignis, nur Tupel, die wegfallen sind, oder einfach jedes Tupel im (jetzigen) Ergebnis.

Datenstrom-Anfragesprachen sind besonders geeignet zur Aggregation von Ereignisdaten, wie v.a. für Marktdaten nötig, und bieten eine gute Integration mit Datenbanken. Negation oder temporale Zusammenhänge sind dagegen oft nur umständlich ausdrückbar. Die Umwandlung von Strömen in Relationen und zurück kann als etwas unnatürlich angesehen werden, wie auch die Voraussetzung einer diskreten Zeitachse.

Auf SQL basierende Datenstrom-Anfragesprachen sind derzeit wohl der erfolgreichste Ansatz und werden von mehreren effizienten und skalierbaren Industrieprodukten unterstützt. Zu den bekannteren zählen Oracle CEP, Coral8, StreamBase, Aleri und das Open-Source-Projekt Esper. Es gibt jedoch große Unterschiede zwischen den jeweiligen Sprachvarianten und wichtige Erweiterungen, die über die hier diskutierte Grundidee hinausgehen.

Produktionsregeln

Produktionsregeln, die heute v.a. in Business Rule Management Systemen wie Drools oder ILOG JRules eingesetzt werden, sind keine Ereignisanfragesprache im engeren Sinn. Die Regeln arbeiten eng mit einer Wirts-Programmiersprache (z.B. Java) zusammen und spezifizieren Aktionen, die ausgeführt werden sollen, wenn bestimmte Zustände eintreten [3]. Die Zustände werden ausgedrückt als Bedingungen über Objekte im sog. Working Memory, auch Fakten genannt.

Wegen ihrer inkrementellen Auswertung (z.B. mit Rete) eignen sich Produktionsregeln auch für CEP. Wenn ein Ereignis passiert, muss ein entsprechendes Faktum erzeugt werden. Ereignisanfragen werden dann als Bedingungen über diese Fakten ausgedrückt, wobei der Programmier frei und ohne „Hilfestellung" ist.

CEP mit Produktionsregeln ist sehr flexibel und gut an existierende Programmiersprachen angebunden. Jedoch arbeitet man auf einer niedrigen und – da zustands- und nicht ereignisorientiert – etwas unnatürlichen Abstraktionsebene. Besonders Aggregation und Negation sind nicht leicht auszudrücken. Garbage Collection, also das Entfernen von Ereignissen aus dem Working Memory, muss manuell programmiert werden. (Siehe jedoch [11] zu einem Ansatz für automatische Garbage Collection.) Produktionsregeln haben den Ruf, weniger effizient als Datenstrom-Anfragesprachen zu sein.

Neben ihrer Verwendung in Business Rule Management Systemen, die nicht auf Ereignisse fokussiert sind, sind Produktionsregeln auch ein wichtiger Bestandteil des CEP-Produkts TIBCO Business Events.

CEP in Praxis und Forschung

CEP ist sowohl ein industrieller Wachstumsmarkt als auch ein wichtiges, durch das Zusammenwachsen von Teilgebieten anderer Forschungsrichtungen entstandenes Forschungsgebiet. Trotz erster erfolgreicher Projekte in den am Anfang genannten Anwendungsgebieten [7, 12], besteht weithin großer Bedarf nach Erfahrungen mit und Vergleichen von Ereignisanfragesprachen in konkreten Projekten. (Dies liegt nicht zuletzt auch daran, dass im Algorithmic Trading, dem derzeit größten Markt für CEP, große Geheimhaltung herrscht.) Ferner gibt es noch wenige Benchmarks, um die Performanz von CEP-Systemen zu vergleichen und vorherzusagen. Über Ereignisanfragesprachen hinaus sind für den praktischen Einsatz von CEP auch Referenzarchitekturen und Design Patterns von großer Bedeutung.

Aktivitäten zur Standardisierung und Harmonisierung

Auch wenn sich die derzeit verbreiteten Ereignisanfragesprachen, wie hier getan, grob in drei Familien einteilen lassen, gibt es wichtige Unterschiede zwischen den einzelnen Sprachen innerhalb einer Familie. Ob eine Einigung auf eine einzige, dominante Anfragesprache für CEP möglich und sinnvoll ist, ist derzeit auch durchaus umstritten.

Bemühungen im Hinblick auf einen Standard für eine auf SQL basierende Datenstromanfragesprache sind derzeit erkennbar [8], jedoch noch nicht innerhalb eines offiziellen Organs. Für Produktionsregeln wird eine standardisierte XML-Syntax im Rahmen des Rule Interchange Format (RIF) des W3C entwickelt, jedoch werden die besonderen Erfordernisse von CEP dort derzeit nicht betrachtet. Gleiches gilt für die Production Rule Representation (PRR) der Object Management Group (OMG).

Mit dem Ziel die Modellierung von Ereignissen in UML zu unterstützen, hat die OMG kürzlich einen Aufruf für Vorschläge zu einen Event Metamodel and Profile (EMP) herausgegeben. Das EMP soll auch explizit die Modellierung von CEP-Funktionalität erlauben.

Aktivitäten innerhalb der Event Processing Technical Society (EPTS) zielen auf eine Koordinierung und Harmonisierung ab, unter anderem durch die Arbeit an einem Glossar zu CEP-Begriffen und gerade beginnender Arbeit zur Analyse von Ereignisanfragesprachen. Die EPTS will ferner Standardisierungsbemühungen innerhalb anderer Organe unterstützen.

Aktuelle Forschung: XChange^{EQ}

Manche Probleme der oben diskutierten Sprachen lassen sich darauf zurückführen, dass dort die vier verschiedenen Aspekte von Ereignisanfragen gemischt und dabei bestimmte Aspekte vernachlässigt werden. Das Forschungsprojekt XChange^{EQ} [4] entwickelt eine gleichnamige Ereignisanfragesprache, die einen Ansatz verfolgt, in dem Anfragen im Stil von logischen Formeln ausgedrückt und dabei die vier Aspekte klar getrennt werden. XChange^{EQ} unterstützt ferner deduktive Regeln über Ereig-

nisse und direkte, muster-basierte Anfragen gegen Ereignisse in XML-Formaten.

Die Ereignisanfrage auf der rechten Seite der folgenden deduktiven Regel drückt z.B. aus, dass eine Bestellung mit weniger als zehn Artikeln (q) als verspätet gilt, falls sie innerhalb von zwei Tagen nicht ausgeliefert wurde:

$$\begin{aligned}\text{late}(id, t) \leftarrow\ & o : \text{order}(id, q),\ s : \text{shipping}(id, t),\\ & w : \text{extend}(s, 2\ \text{days}),\\ & \text{while}\ w : \text{not delivery}(t),\ o\ \text{before}\ s,\ q < 10\end{aligned}$$

Die Forschung an XChange^{EQ} illustriert die Bedeutung von gutem Sprachdesign und formalen Grundlagen in CEP und versucht Probleme mit den verbreiteten Ansätzen zu beheben. Der prinzipielle Ansatz, Ereignisanfragen als logische Formeln mit einer Trennung der Aspekte zu schreiben ist z.B. auch übertragbar auf Produktionsregeln und kann dort eine Hilfestellung bei der Formulierung von Anfragen bieten.

Weitere Forschungsthemen

Forschungsbedarf im Complex Event Processing besteht insbesondere noch an den formalen Grundlagen, besonders im Hinblick auf Ausdrucksstärke und Optimierung. Wichtig für die Anfrageoptimierung sind dabei Ausnutzung von gemeinsamen Teilanfragen (Multi-Query-Optimierung) sowie verteilte und parallele Auswertung. Weitere Forschungsthemen sind der Umgang mit ungenauen Ereignissen (z.B. probabilistische Methoden) und die Erkennung von unbekannten komplexen Ereignissen (z.B. Data Mining auf Ereignisströmen).

Literatur

1. Adi A, Etzion O (2004) Amit – the situation manager. VLDB Journal 13(2):177–203
2. Arasu A, Babu S, Widom J (2006) The CQL continuous query language: Semantic foundations and query execution. VLDB Journal 15(2):121–142
3. Berstel B, Bonnard P, Bry F, Eckert M, Pătrânjan P-L (2007) Reactive rules on the Web. In Reasoning Web, Int. Summer School, Number 4636 in LNCS. Springer, Berlin Heidelberg, pp. 183–239
4. Bry F, Eckert M (2007) Rule-Based Composite Event Queries: The Language XChange^{EQ} and its Semantics. In Proc. Int. Conf. on Web Reasoning and Rule Systems, Number 4524 in LNCS. Springer, Berlin Heidelberg, pp. 16–30
5. Eckert M (2008) Complex Event Processing with XChange^{EQ}: Language Design, Formal Semantics, and Incremental Evaluation for Querying Events. PhD thesis, Institute for Informatics, University of Munich, http://edoc.ub.uni-muenchen.de/9405/ (Zugriff: 16.02.2009)
6. Event Processing Technical Society (EPTS), http://www.ep-ts.com (Zugriff: 16.02.2009)
7. Greiner T, Düster W, Pouatcha F, von Ammon R, Brandl H-M, Guschakowski D (2006) Business activity monitoring of norisbank taking the example of the application easyCredit and the future adoption of complex event processing (CEP). In Proc. Int. Symp. on Principles and Practice of Programming in Java. ACM, New York, pp. 237–242
8. Jain N, Mishra S, Srinivasan A, Gehrke J, Widom J, Balakrishnan H, Çetintemel U, Cherniack M, Tibbetts R, Zdonik SB (2008) Towards a streaming SQL standard. In Proc. Int. Conf. on Very Large Databases, VLDB Endowment, Auckland, NZ, pp. 1379–1390
9. Luckham DC (2002) The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Reading
10. Paton NW (ed) (1998) Active Rules in Database Systems. Springer, Berlin Heidelberg
11. Walzer K, Breddin T, Groch M (2008) Relative temporal constraints in the Rete algorithm for complex event detection. In Proc. Int. Conf. on Distributed Event-Based Systems. ACM, New York, pp. 147–155
12. Wittenburg G, Terfloth K, Villafuerte FL, Naumowicz T, Ritter H, Schiller JH (2007) Fence monitoring — experimental evaluation of a use case for wireless sensor networks. In Proc. Eur. Conf. on Wireless Sensor Networks, volume 4373 of LNCS, Springer, Berlin Heidelberg, pp. 163–178