Ein Aktionsmodell für die kooperative Bearbeitung von XML-Daten

Francis Gropengießer, Jens-Oliver Fischer
FG Datenbanken und Informationssysteme, TU Ilmenau
Fraunhofer-IDMT, Ilmenau
francis.gropengiesser@tu-ilmenau.de
fir@idmt.fraunhofer.de

Zusammenfassung

In vielen Anwendungsbereichen, wie z.B. im Design oder dem Medienproduktionsprozess, ist das kooperative Arbeiten mehrerer Nutzer auf einem zentralen Datenbestand unerlässlich geworden. Ein spezieller Anwendungsfall aus dem Bereich der Medienproduktion ist der Ausgangspunkt für die Betrachtungen in dieser Arbeit. Aus ihm lässt sich eine Reihe von Anforderungen ableiten. Für eine ausgewählte Menge existierender Transaktions-/Aktionsmodelle wird gezeigt, dass diese die Anforderungen nicht im vollen Maße erfüllen. In der Arbeit wird daher ein neues kooperatives Aktionsmodell vorgestellt. Die Vorteile des Modells sind die Ermöglichung eines kooperativen Arbeitsprozesses, ein voll automatisches Transaktionsrecovery, die einfache Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, durch das Reset-Repeat-Prinzip sowie das Vermeiden eines nachträglichen Mischens verschiedener Versionen eines XML-Dokumentes.

1 Einleitung

Produktionsprozesse, z.B. Design- oder Medienproduktionsprozesse, erfordern ein kooperatives Arbeiten mehrerer Nutzer auf einem gemeinsamen Datenbestand bzw. an einem gemeinsamen Projekt. Nur so können die Ideen und Vorschläge jedes einzelnen Beteiligten berücksichtigt werden und in die Erstellung eines Gesamtergebnisses oder eines Endproduktes mit einfließen.

Einen speziellen Anwendungsfall aus dem Bereich der Medienproduktion stellt das am Fraunhofer-IDMT entwickelte neuartige Audiowiedergabesystem IOSONO [IOS] dar, das auf dem Prinzip der Wellenfeldsynthese [Ber88] beruht. Es ermöglicht die stabile Positionierung virtueller Schallquellen im dreidimensionalen Raum, wodurch ein realitätsgetreues Raumklangerlebnis für den Hörer entsteht. Die bei der Positionierung der Soundobjekte entstehenden Metainformationen werden in Form eines Szenegraphs dargestellt, der in einer XML-Datei abgespeichert wird.

Aus Sicht eines Datenbankentwicklers besteht die Kernaufgabe darin, das kooperative Arbeiten mehrerer Autoren auf den gleichen XML-Daten transaktionell zu ermöglichen. Unter Kooperation ist ein verhandelndes Arbeiten mehrerer gleichberechtigter Nutzer an einer gemeinsamen Aufgabe zu verstehen. Es findet ein Informationsfluss in beliebiger Richtung zwischen den Nutzern statt.

Allerdings birgt das kooperative Arbeiten mehrerer Autoren auf einer zentralen Datenbasis einige Probleme in sich. Die Transaktionen, welche die Designtätigkeiten der Autoren beinhalten, sind sehr lang. Dies führt einerseits dazu, dass Änderungen erst sehr spät sichtbar werden, was ein kooperatives Arbeiten verhindert, und andererseits dazu, dass im Falle eines Abbruchs der gesamte Arbeitsfortschritt verloren ist. Weiterhin erfordert die Ermöglichung eines kooperativen Arbeitsprozesses den Verzicht auf Serialisierbarkeit und damit die Aufweichung der Isolation.

Benötigt werden daher ein offen geschachteltes Transaktionskonzept, was ein frühes Sichtbarmachen von Änderungen erlaubt und die Atomarität im strengen Sinne aufweicht, ein Recoverykonzept, welches die Fehleratomarität gewährleistet, ein geeignetes Synchronisationsverfahren

sowie ein Konzept zur Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können.

2 Verwandte Arbeiten

Basis für diese Arbeit sind erweiterte Transaktionskonzepte. Diese stellen eine Möglichkeit dar, den Problemen langer Transaktionen zu begegnen. Allerdings sind sie nur eingeschränkt für den speziellen Anwendungsfall aus Abschnitt 1 geeignet, wie nachfolgend gezeigt wird.

Die geschlossen geschachtelten Transaktionen [Mos81, Kru97] fordern Isolation zwischen verschiedenen Transaktionsbäumen und zwischen Geschwistern innerhalb eines Transaktionsbaumes. Dadurch wird jegliche Form der Kooperation unterbunden.

Die offen geschachtelten Transaktionen [WS92, Kru97] erfüllen die Forderung nach Kooperation im vollen Umfang. Problematisch gestaltet sich jedoch die Fehlerbehandlung. Hier sind Kompensationstransaktionen notwendig, die eine Vielzahl von Problemen [GFJK03] in sich bergen.

Das Konzept der Sagas [GMS87, Moc95, Kru97] fordert Kommutativität der Subtransaktionen verschiedener Sagas, wodurch Kooperation unterbunden wird.

Das ConTract-Modell [WR92, Kru97] ermöglicht zwar die Kooperation verschiedener Autoren, bedarf jedoch wieder der Nutzung von Kompensationstransaktionen zur Fehlerbehandlung.

Das DOM-Transaktionsmodell [BOH$^+$92, Kru97] ermöglicht die Kooperation verschiedener Autoren. Allerdings basiert die Fehlerbehandlung auch in diesem Modell auf den Kompensationstransaktionen.

Das CONCORD-Modell [RMH$^+$94, Kru97] ermöglicht ebenfalls Kooperation. Allerdings ist in diesem Modell das Verschmelzen verschiedener Versionen eines XML-Dokumentes zu einer finalen erforderlich, wodurch ein zusätzlicher Arbeitsaufwand für die Autoren entsteht.

Die geschachtelten dynamischen Aktionen [NW94, Kru97] basieren auf den dynamischen Aktionen [NS92, Moc95], die ihrerseits die Dauerhaftigkeit garantieren und somit einen Verzicht auf Kompensationsaktionen ermöglichen. Allerdings fordern die geschachtelten dynamischen Aktionen Serialisierbarkeit für ganze Aktionsbäume, wodurch keine Kooperation zwischen verschiedenen Autoren möglich ist.

Die geschachtelten dynamischen Aktionen für kooperative Anwendungen [Kru97] heben die Serialisierbarkeit gezielt innerhalb von Kooperationsgruppen auf. Allerdings ist das Prinzip der Kooperationgruppen inpraktikabel für den speziellen Anwendungsfall aus Abschnitt 1. Weiterhin erfordert die Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, ein sehr systemnahes Eingreifen der Autoren. Dies kann jedoch von einem Designer nicht erwartet werden.

Zusammenfassend kann festgehalten werden, dass keines der hier genannten Modelle als vollständig geeignet für den speziellen Anwendungsfall aus Abschnitt 1 angesehen werden kann. Somit ist die Entwicklung eines neuen Transaktions-/Aktionskonzepts erforderlich.

3 Das kooperative Aktionsmodell

3.1 Aufbau des Modells

Zunächst sollen die für die Arbeit an einem XML-Dokument notwendigen Operationen definiert werden. Read dient dem inhaltlichen, strukturellen oder holografischen Lesen eines Knotens oder Teilbaums. Inhaltliches Lesen dient dem Erfassen des Attributwerts oder des Textes eines Knotens, strukturelles Lesen dem Erfassen der Struktur eines XML-Dokumentes und holografisches Lesen dem Sehen bereits gelöschter Knoten. Mit Edit werden die Werte einzelner Attribut- oder Textknoten geändert. Delete dient dem Löschen von Knoten oder Teilbäume. Mit Insert kann eine Knotenmenge an ein XML-Element angefügt werden. Mit Hilfe der Operation Move wird

eine Knotenmenge verschoben. Dabei dürfen andere Operationen (Read, Edit, Delete, Move, Insert) auf dieser Knotenmenge ausgeführt werden. Mit Reset wird eine Folge von Operationen rückgängig gemacht. Repeat dient dem Wiederholen der durch ein Reset rückgängig gemachten Operationen. Das Reset-Repeat-Prinzip wird zur Behandlung von Konflikten genutzt, die infolge der fehlenden Serialisierbarkeit auftreten können.

Nachfolgend sollen nun die zulässigen Operationsfolgen definiert werden. Eine Operationsfolge besteht entweder aus einer Folge von Leseoperationen und höchstens einem Edit, Delete, Move, Insert, Reset oder Repeat oder zwei Folgen von Leseoperationen und genau einem Move. Mindestens eine Leseoperation ist Bestandteil jeder Operationsfolge. Jede weitere Leseoperation dient nur dem Hineinspringen in eine Verschiebung, da von außerhalb keine Änderungsoperationen (Edit, Delete, Move, Insert, Reset, Repeat) auf der sich in einer Verschiebung befindenden Knotenmenge ausgeführt werden dürfen.

Anhand eines Beispiels soll nun der Aufbau des Aktionsmodells dargestellt werden. Auf dem Beispielszenegraph aus Abbildung 1, wobei sich der Teilbaum *(Effekte, Knall)* in einer Verschiebung befindet, wird die Operationsfolge *Read(Szene, Dialog, Effekte, Knall) Read(Effekte, Knall) Delete(Knall)* ausgeführt. In der Abbildung 2 ist die Umsetzung dieser Operationsfolge auf das Aktionsmodell zu sehen. Dabei werden Operationen auf Teilbäumen in Operationen auf

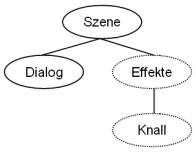

Abbildung 1: Beispielszenegraph

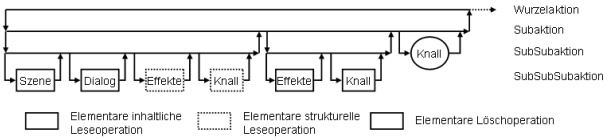

Abbildung 2: Aufbau des Aktionsmodells

einzelnen Knoten zerlegt. Dies ermöglicht das Abbrechen elementarer Operationen, ohne dass dies das Scheitern einer Operation auf einem Teilbaum oder einer ganzen Operationsfolge nach sich zieht.

3.2 Synchronisation

Die Synchronisation konkurrierender Operationen wird durch ein Sperrmodell realisiert. Hierbei wird berücksichtigt, dass es sich um Operationen auf XML-Bäumen handelt.

Zunächst sollen allen Operationen entsprechende Sperrtypen zugeordnet werden. CRL, SRL und HRL sind die Sperrtypen für das inhaltliche, strukturelle und holografische Lesen. EL wird der Operation Edit, DL dem Delete, IL der Operation Insert, ML dem Move und RRL dem Reset und Repeat zugewiesen.

Tabelle 1 zeigt die Sperren-Konfliktmatrix. Für die ML werden zwei Fälle unterschieden. In der vorletzten Zeile der Tabelle ist der Fall dargestellt, in dem sich der Autor außerhalb der Verschiebung befindet. Die letzte Zeile greift den Fall auf, in dem der Autor Teil der Verschiebung ist. ($\surd$ bedeutet vollständige Kompatibilität, $-$ keine Kompatibilität und $+$ bedeutet, dass die Sperren nur kompatibel sind, wenn nicht das Wurzelelement der Verschiebung betroffen ist.)

3.3 Transaktionsrecovery

Die Behandlung von Transaktionsabbrüchen basiert auf einem Objektversionensystem, wobei hier, im Gegensatz zur Multiversionen-Synchronisation [BG83], nur eine sequenzielle Historie von Objektversionen zugelassen wird. Jede Operation greift daher immer auf die aktuelle Version zu. Zur Protokollierung der Historie eines Knotens wird ein Log-Buch eingesetzt. Es hält in drei Dimensionen (Inhalt, Ort und Materialisierung) alle Informationen zu den verschiedenen Versionen eines Knotens fest. Wird eine Aktion abgebrochen, so müssen lediglich die entsprechenden Informationen aus den Dimensionen des Log-Buchs entfernt werden. Weiterhin ist durch die Verwendung dieses Fehlerbehandlungsmodells eine einfache Realisierung des Reset-Repeat-Prinzips

	SRL	CRL	HRL	EL	DL	IL	RRL	ML
SRL	√	√	√	√	−	√	−	
CRL	√	√	√	−	−	√	−	
HRL	√	√	√	√	√	√	√	
EL	√	−	√	−	−	−	−	
DL	−	−	√	−	−	−	−	
IL	√	√	√	−	−	√	−	
RRL	−	−	√	−	−	−	−	
ML	√	−	−	−	−	−	−	−
ML	√	√	√	√	+	√	−	+

Tabelle 1: Sperren-Konfliktmatrix

möglich. Um beispielsweise mehrere Operationen anderer Autoren rückgängig zu machen, muss ein Nutzer lediglich einen alten Zustand eines Objekts auswählen. Dieser wird dann durch den einmaligen Aufruf von Reset wiederhergestellt.

3.4 Eigenschaften

In diesem Abschnitt soll kurz auf einige Eigenschaften eingegangen werden, die durch das entwickelte Modell ermöglicht bzw. garantiert werden.

ACID Die Atomarität im strengen Sinne wird in diesem Modell stark aufgeweicht. Subaktionen können abbrechen, ohne dass dies das Scheitern der Wurzelaktion nach sich zieht. Die Fehleratomarität hingegen ist gesichert, da abgebrochene Aktionen keine Effekte auf die Datenbank haben.

Die einzige in diesem Modell definierte Integritätsbedingung ist die Wahrung der Struktur eines XML-Dokumentes. Diese wird durch die Operationen garantiert. Wird beispielsweise das Delete für einen Knoten aufgerufen, welcher Nachfolger hat, so wird auch für diese das Delete aufgerufen.

Die Isolation wird in dem Modell zwischen Wurzelaktionen aufgehoben. Wurde eine Subaktion beendet, werden die benutzten Sperren für die Allgemeinheit freigegeben.

Die Dauerhaftigkeit ist dadurch garantiert, dass das Modell auf den dynamischen Aktionen basiert, die ihrerseits die Dauerhaftigkeit gewährleisten.

Kooperation Die Kooperation wird in dem Modell dadurch ermöglicht, dass die Isolation zwischen Wurzelaktionen aufgehoben wird. Ordnet man jedem Autor eine Wurzelaktion zu, so kann ein Informationsfluss in beliebiger Richtung zwischen ihnen stattfinden.

Transaktionsrecovery Das vorgeschlagene Fehlerbehandlungsmodell ist einfach zu realisieren. Es ermöglicht eine vollständig automatische Behandlung von Aktionsfehlern und das Reset-Repeat-Prinzip zur Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können. Durch die streng sequenzielle Historie steht am Ende der Arbeiten der Autoren eine finale Version eines Dokumentes fest, die persistent gespeichert werden kann.

4 Zusammenfassung und Ausblick

Gegenstand dieser Arbeit war die Präsentation eines neu entwickelten Aktionsmodells. Aus einem speziellen Anwendungsfall, dem IOSONO-System, wurden einige Probleme und Aufgabenstellungen abgeleitet. Eine kurze Bewertung existierender Transaktions-/Aktionsmodelle hat ergeben, dass keines dieser Modelle als vollständig geeignet für den Einsatz in dem speziellen

Anwendungsfall angesehen werden kann. Daher wurde ein neues Modell auf Basis der dynamischen Aktionen für kooperative Anwendungen entwickelt und vorgestellt. Dieses zeichnet sich durch seine Unterstützung der Kooperation von Autoren, sein voll automatisches Transaktionsrecovery, die einfache Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, durch das Reset-Repeat-Prinzip sowie das Vermeiden eines nachträglichen Mischens verschiedener Versionen eines XML-Dokumentes aus.

In zukünftigen Arbeiten gilt es herauszufinden, wie sich eine Realisierung dieses Modells in der Praxis bewährt. Des Weiteren ist die Entwicklung einer Recoverykomponente zur Behandlung von Systemausfällen notwendig.

Literatur

[Ber88] A. J. Berkhout. A holographic approach to acoustic control. In *Journal Audio Eng. Soc.*, volume 36, pages 977–995. 1988.

[BG83] Philip A. Bernstein and Nathan Goodman. Multiversion concurrency control - theory and algorithms. In *ACM Trans. Database Syst.*, volume 8, pages 465–483, 1983.

[BOH+92] Alejandro P. Buchmann, M. Tamer Ozsu, Mark Hornick, Dimitrios Georgakopoulos, and Frank A. Manola. A transaction model for active distributed object systems. In *Database Transaction Models for Advanced Applications*, pages 123–158, 1992.

[GFJK03] Paul Greenfield, Alan Fekete, Julian Jang, and Dean Kuo. Compensation is Not Enough. In *EDOC '03*, page 232, 2003.

[GMS87] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD '07*, pages 249–259, 1987.

[IOS] http://www.iosono-sound.com/.

[Kru97] Armin Kruse. Ein kooperatives Sperrverfahren für geschachtelte dynamische Aktionen — Diplomarbeit an der Universität Bonn. März 1997.

[Moc95] Michael Mock. *Aktionsunterstützung für verteilte, kooperative Anwendungen — Konzept und Realisierung.* GMD-Bericht Nr. 250, R. Oldenbourg Verlag, 1995.

[Mos81] J. Eliot B. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing.* PhD thesis, 1981.

[NS92] Edgar Nett and Ralf Schumann. Supporting fault-tolerant distributed computations under real-time requirements. In *Comput. Commun.*, volume 15, pages 252–260, 1992.

[NW94] Edgar Nett and Beatrice Weiler. Nested dynamic actions - how to solve the fault containment problem in a cooperative action model. In *Symposium on Reliable Distributed Systems*, pages 106–115, 1994.

[RMH+94] Norbert Ritter, Bernhard Mitschang, Theo Härder, Michael Gesmann, and Harald Schöning. Capturing Design Dynamics the Concord Approach. In *ICDE*, pages 440–451, 1994.

[WR92] Helmut Wächter and Andreas Reuter. The contract model. In *Database transaction models for advanced applications*, pages 219–263, 1992.

[WS92] Gerhard Weikum and Hans-Jorg Schek. Concepts and applications of multilevel transactions and open nested transactions. In *Database Transaction Models for Advanced Applications*, pages 515–553. 1992.