# Konsistenzwahrung durch aktive Vermittlersysteme

Thomas Kudraß, Jürgen Zimmermann
Technische Hochschule Darmstadt, Fachbereich Informatik, FG Datenverwaltungssysteme
D-64293 Darmstadt, Julius-Reiber-Str. 17
E-Mail: {kudrass, zim}@dvs1.informatik.th-darmstadt.de

## Zusammenfassung

*In diesem Papier wird diskutiert, wie durch den Einsatz von Vermittlersystemen in Multidatenbanken globale Integritätsbedingungen kontrolliert werden können. Ausgehend von einer allgemeinen Architektur von Vermittlersystemen werden die Anforderungen an einzelne Komponenten und deren Zusammenwirken beschrieben. Am Beispiel einer objektorientierten Integration von relationalen Datenbanken wird die prinzipielle Arbeitsweise von aktiven Vermittlern dargestellt. Zusammenfassend wird ein Ansatz skizziert bei dem aus einer deklarativen Beschreibung von Integritätsbedingungen die Komponenten eines regelbasierten Vermittlersystems generiert werden können.*

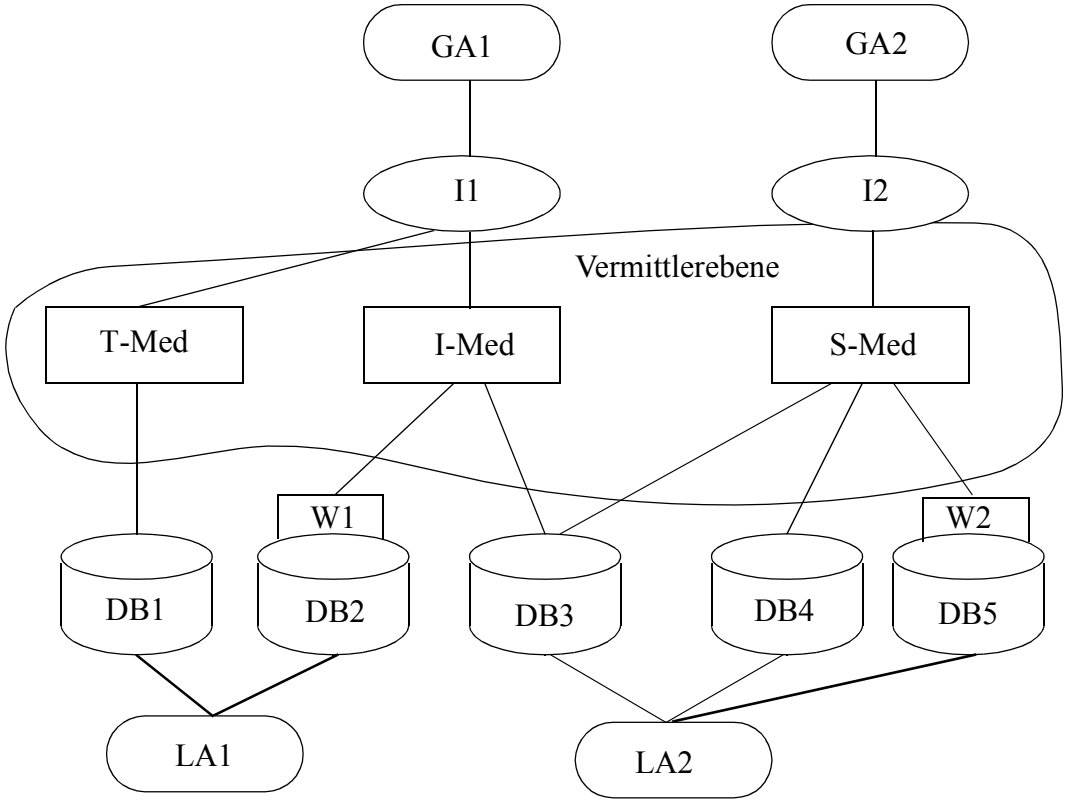
## 1 Einleitung

Die Erfordernisse wachsender Kooperation existierender Informationssysteme verursachen Probleme bei der Integration der Daten, die zwar semantisch verwandt sind (z.B. Replikate), aber möglicherweise in heterogenen Subsystemen repräsentiert werden. Wir untersuchen hierbei die Fragestellung, inwieweit durch aktive Vermittlersysteme (*Mediators*) globale Konsistenzbedingungen ausgedrückt und überwacht werden können. Nach [Wied92] kann ein Mediator als eine Softwareschicht zwischen Datenhaltung und Applikation betrachtet werden. Eine Reihe von wissensbasierten Vermittlerdiensten sind denkbar, so z.B. die Kontrolle globaler Integrität oder Security-Dienste. Neben den Integritätsregeln sind auch zusätzliche semantische Informationen über die lokalen Applikationen erforderlich (in [Madn95] als *Kontext* bezeichnet) Wir verfolgen den Mediator-Ansatz in einer Multidatenbankumgebung, in der sowohl globale als auch lokale Queries gestellt werden können. Die globale Integrität kann damit durch globale Benutzer, aber auch durch lokale Benutzer verletzt werden, die ihre Applikationen wie vor der Integration verwenden. Um die Erfordernisse lokaler Autonomie und globaler Konsistenz möglichst gut in Übereinstimmung zu bringen, wurde eine flexible Lösung entworfen, bei der die lokalen Systeme durch einen aktiven Ansatz integriert werden können.

## 2 Eine Vermittler-Architektur

In [Wied95] wird eine Mediator-Architektur als Erweiterung des Client-Server-Modells beschrieben. Zwischen globalen Applikationen und einer Anzahl von lokalen (heterogenen) Datenbanken befindet sich die Vermittlungs-Ebene, innerhalb der verschiedene Dienste der Konsistenzkontrolle angesiedelt sind. Zu diesen Diensten gehören die Überwachung globaler Integritätsbedingungen, die durch Datenbank-Ereignisse verletzt werden können, die Steuerung zeitabhängiger Aufgaben als auch die Kontrolle lokaler Schemaveränderungen. Die

Vermittler sind einerseits für bestimmte Aufgaben, andererseits für bestimmte Weltausschnitte spezialisiert (vgl. Abbildung 1). Sie haben jeweils eine Wissensbasis, die entsprechend unserem Ansatz als ECA-Regeln (*E*vent *C*ondition *A*ction) repräsentiert ist. Die Aufgabe des Integrators ist die Integration heterogener Datenbanken, um dem globalen Benutzer eine einheitliche Sicht zu gewähren. Um die Kontrollfunktion des Mediators zu unterstützen, ist oftmals eine zusätzliche Schicht auf den lokalen Datenbanksystemen notwendig, die als Wrapper bezeichnet wird.




| | | | |
|---|---|---|---|
| GA | Globale Applikation | I-Med | Integritäts-Vermittler |
| I | Integrator | S-Med | Schema-Vermittler |
| W | Wrapper | T-Med | Temporaler Vermittler |
| DB | Datenbank | | |
| LA | Lokale Applikation | | |

Abbildung 1: Architektur eines Systems mit mehreren Vermittlern zur Konsistenzkontrolle

## 3 Datenbank-Wrapping

Voraussetzung für die Integration eines lokalen Systems ist die Anwendung eines Wrapping-Mechanismus, dem im Kontext von Integritätskontrolle die Aufgabe der lokalen Event-Detektion zukommt. Dieses sollte unter größtmöglicher Wahrung lokaler Autonomie geschehen, d.h. möglichst keine Änderungen an Applikations- und Datenbanksystemen bzw. Datenbanken erfordern. Unter mehreren möglichen Ansätzen [Wins95] wurde für Client-Server-Systeme das Database-Wrapping favorisiert, das durch ein zusätzliches Interface zwischen Applikationen und Datenbankserver realisierbar ist. Das Prinzip besteht darin, jedes am Server ankommende Statement zu analysieren und gegebenfalls eine Weiterleitung an den Vermittler vorzunehmen. Einzelheiten einer solchen Gateway-Lösung am Beispiel relationaler Datenbanken können

[Kray95] entnommen werden. Darüber hinaus ist die Protokollierung lokaler Ereignisse möglich. Das Gateway kann entsprechend der global benötigten Dienste flexibel konfiguriert werden. Voraussetzung für den gewählten Ansatz ist die Existenz einer dynamisch interpretierbaren Query-Language und die Möglichkeit eines Eingriffs in die Kommunikation zwischen Clients und Datenbank-Server.

## 4 Eine objekt-relationale Testumgebung

Lokale Datenbanken lassen sich im Verbund als ein Raum aktiver Objekte modellieren [Buch90], deren Verhalten durch ECA-Regeln beschrieben ist. Die von uns untersuchte Testumgebung umfaßt lokale relationale Datenbanken, die aufgrund ihres hohen Verbreitungsgrades untersucht wurden. Ein objektorientiertes Modell ist durch seine höhere Ausdrucksmächtigkeit als globales Datenmodell geeignet. Hierzu betrachten wir ODMG-93 als einen Quasi-Standard für objektorientierte Datenbanken, dessen Einbindung in C++ definiert ist [Catt93]. Dem globalen Benutzer soll eine objektorientierte Sicht auf die lokalen relationalen Daten ermöglicht werden, wofür der Einsatz eines objekt-relationalen Systems in Frage kommt. Dabei ist eine strukturelle und operationelle Abbildung C++/SQL in beiden Richtungen wünschenswert.

Das *forward mapping* umfaßt die Erzeugung eines relationalen Schemas aus einer objektorientierten Darstellung (z.B. C++ Headerdateien). Methodenaufrufe sind in äquivalente SQL-Befehle zu transformieren. Ausgewählt wurde dafür das System Persistence[1] mit seinen beiden Komponenten *Relational Interface Generator* (RIG) und *Relational Object Manager* (ROM) [KeJA94]. Es lassen sich damit Multidatenbank-Applikationen in C++ entwickeln, wobei die Daten in relationalen Datenbanken gehalten werden. Wichtig für die Integration existierender Datenbanken ist der Einsatz sogenannter Dictionary-Reader, die ein relationales Schema in ein objektorientiertes Schema transformieren (strukturelles *reverse mapping*). Auf dem Markt verfügbare Werkzeuge beschränken sich zumeist auf eine 1:1-Abbil-dung von Tabellen auf Klassen ohne semantische Anreicherung des generierten Schemas. Sollen lokale SQL-Kommandos auch durch den objektorientierten Vermittler verarbeitet werden, besteht das Hauptproblem der operationellen Abbildung in Methodenaufrufe auf Objekten darin, daß die Zielsprache nicht dynamisch ist (d.h. keine Interpretation zur Laufzeit erfolgt) und das Klassenschema bekannt sein muß.

Für diesen Zweck wurde ein Mediator-Generator entwickelt, der aus zwei Bestandteilen besteht: einem Parser, der das lokale DB-Schema verarbeitet, indem er alle Informationen über Klassen, Attribute und Beziehungen extrahiert, und einem Präprozessor, der mit Hilfe von Template-Files den Code des Vermittlers generiert. Mehr Details darüber können [KuLB96, Loew95] entnommen werden. Gateway und Vermittler kommunizieren miteinander auf der Basis von Remote Procedure Calls, am Gateway ankommende Befehle (insbesondere DML-Kommandos) können somit zum Vermittlerprozeß umgeleitet werden, unter dessen Kontrolle diese dann ausgeführt werden.

[1] Persistence ist ein Produkt von Persistence Software, Inc.

## 5 Aktive Eigenschaften des Vermittlers

Ein Ziel der Arbeit bestand in der Untersuchung, inwieweit Konzepte aktiver Datenbanksysteme [Daya88, BBKZ93] sich auf Multidatenbanken übertragen lassen. Im Rahmen des Projektes REACH wurde bereits ein homogenes aktives DBMS mit C++-Schnittstelle basierend auf dem objektorientierten DBMS Open OODB entwickelt [BZBW95]. Events können in REACH Methodenevents, Transaktionsevents, Zeitevents oder auch Kompositionen aus diesen sein. Zur Detektion ist ein Methoden-Wrapping erforderlich, zu dessen Umsetzung ein Präcompiler genutzt wird, mit dem die Applikationen übersetzt werden müssen.

Im Unterschied dazu beschränkt sich die Regelkomponente des Vermittlers auf Datenbank-Events, die über vom Relational Interface Generator *(*RIG*)* vordefinierte Methodenschnittstellen ausgelöst werden können [Bitt96]. Dazu gehören: Objekt-Events (Erzeugen, Löschen, Lesen, Abspeichern), Attribut-Events (Wertänderung), Beziehungs-Events (Ändern, Lesen). Benutzerdefinierte Methoden müssen zum Zugriff auf die Daten immer die vom RIG bereitgestellten Methoden aufrufen, so daß der Aufwand für die Detektion sich auf die Definition von sogenannten Hook-Methoden beschränkt, die vor oder nach dem Datenbank-Event aufgerufen werden und die ggf. Regeln triggern. Die Repräsentation der Regeln erfolgte in Form von Single-Instance-Regelklassen als Subklassen einer allgemeinen Klasse *Rule*.

Bei der Implementierung der Regelverwaltung in einem heterogenen System waren einige Besonderheiten zu beachten, die in einem homogenen aktiven DBS nicht auftreten. Kopplungsmodi zwischen Event- und Condition bzw. Condition und Action beruhen auf der Existenz *eines* Transaktionsmanagers. Wenn aber in einer globalen Applikation gleichzeitig Transaktionen in verschiedenen Datenbanksystemen eröffnet werden können, muß das Event-Management entsprechend erweitert werden, um die Lokalität des Events zu erfassen.

## 6 Ein Werkzeugkasten zur Generierung eines Konsistenz-Mediators

Ausgehend von einer Analyse und Beschreibung von globalen Integritätsbedingungen besteht das Ziel darin, einen Vermittler automatisch zu generieren. Dazu sind folgende Bestandteile notwendig: Die beteiligten Datenbankschemata müssen identifiziert werden, dazu erforderlich sind Angaben über bestehende datenbankübergreifende Abhängigkeiten, sogenannte *Interdependencies* [ShRu90]. Zusätzliche Informationen enthalten Konsistenzprädikate, denen eine zustands- oder zeitbasierte Abschwächung des Konsistenzbegriffs zugrundeliegt. Weiterhin können Dienste benannt werden, die von den lokalen Systemen (bzw. vom lokalen Datenbankgateway) zusätzlich zu erbringen sind, z.B. eine Protokollierung der lokal ausgeführten Kommandos.

Aus den Angaben über die Objekte, die Bestandteil globaler Konsistenzprädikate sind, sowie den geforderten Diensten läßt sich die Konfiguration des Gateways ableiten, das somit von überflüssigen Analyse- und Protokollierungsaufgaben entlastet werden kann. Die Schemadaten bilden den Input für den Mediator-Generator, aus dem ein schemaabhängiger Vermittler generiert wird. Aus der Definition der Abhängigkeiten werden ECA-Regeln produziert (vgl. hierzu auch [CeWi93]), die vom Vermittler verarbeitet werden, so daß als Ergebnis ein regelbasierter Vermittlerprozeß erzeugt wird, der mit dem Gateway interagieren kann.

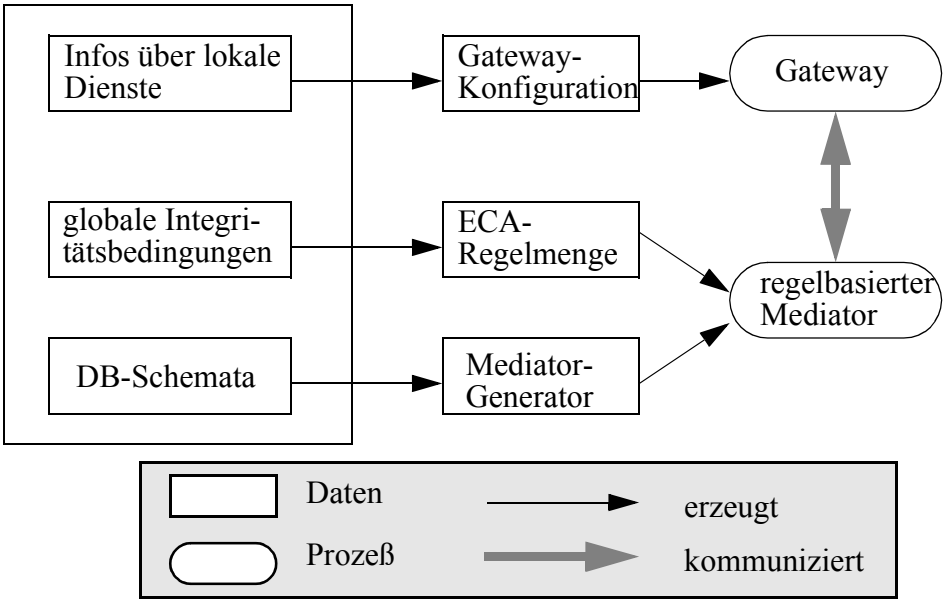



Abbildung 2: Erzeugung eines Vermittlers zur Konsistenzwahrung

## Literatur


[Bitt96] T. Bittmann: *Entwicklung einer Regelverwaltungskomponente in einem Vermittlersystem für heterogene Datenbanken*, Diplomarbeit, TH Darmstadt, FB Informatik, 1996.

[Buch90] A. Buchmann: *Modelling Heterogeneous Systems as a Space of Active Objects*, in: Proc. 4th Intl. Workshop on Persistent Objects, Martha's Vineyard, Sept. 1990.

[BBKZ93] H. Branding, A. Buchmann, T. Kudrass, J. Zimmermann: *Rules in an open system: The REACH rule system*, in: Proc. of the 1st Intl. Workshop on Rules in Database Systems, Edinburgh, 1993.

[BZBW95] A. Buchmann, J. Zimmermann, J.A. Blakeley, D.L. Wells: *Building an Integrated Active OODBMS: Requirements, Architecture and Design Decisions*, in: Proc. 11th Internat. Conference on Data Engineering, Taipeh, 1995.

[Catt93] R. Cattell (ed.): *The Object Database Standard: ODMG-93*, Morgan Kaufmann, 1993.

[CeWi93] S. Ceri, J. Widom: *Managing Semantic Heterogeneity with Production Rules and Persistent Queues*, in: Proceedings of the 19th International VLDB Conference, 1993.

[Daya88] U. Dayal: *Active Database Management Systems*, in: Proc. of the 3rd Internat. Conference on Data and Knowledge Bases, Jerusalem, 1988.

[KeJA94] A. Keller, R. Jensen, S. Agarwal: *Enabling the Integration of Object Applications with Relational Databases*, Persistence Technical Overview, Persistence Software, Inc., 1994.

[Kray95] R. Krayer: *Entwicklung eines relationalen Datenbank-Gateways zur Unterstützung globaler Konsistenzkontrolle,* Diplomarbeit, TH Darmstadt, FB Informatik, 1995.

[KuLB96] T. Kudrass, A. Loew, A. Buchamnn: *Active Object-Relational Mediators*, in: Proc. of the 1st International Conference on Cooperative Informations Systems (CoopIS), Brüssel, 1996.

[Loew 95] A. Loew: *Evaluierung des Datenbank-Integrationstools Persistence und Erprobung als aktives Vermittlersystem*, Diplomarbeit, TH Darmstadt, FB Informatik, 1995.

[Madn95] S.E. Madnick: From VLDB to VMLDB (Very Many Large Databases): Dealing With Large Scale Semantic Heterogeneity, Proc. of the 21st Internat. VLDB Conf., Zürich, 1995.

[ShRu90] A. Sheth, M. Rusinkiewicz: *Management of Interdependent Data: Specifying Dependency and Consistency Requirements,* Proc. Workshop on Management of Replicated Data, Houston, 1990.

[Wied92] G. Wiederhold: *Mediators in the architecture of future information systems*, IEEE Computer '92.

[Wied95] G. Wiederhold: *Value-added Mediation in Large-Scale Information Systems*, Proc. of the IFIP-DS6 Conference, Atlanta, 1995.

[Wins95] P. Winsberg: *Legacy Code: Don't Bag It, Wrap It*, in: Datamation, May, 1995.