Einsatz von aktiven Datenbanken für CIM

Herbert Groiss, Johann Eder

Institut für Informatik
Universität Klagenfurt
Universitätsstr. 65
9020 Klagenfurt
e-mail: {herb,eder}@ifi.uni-klu.ac.at

Abstract

Die Entwicklung von CIM Systemen ist wegen der hohen Komplexität eine große Herausforderung sowohl für die organisatorische Konzeption und Implementierung als auch für die informationstechnische Realisierung. Die Vorteile der Verwendung von Datenbanken für die integrierte Verwaltung aller in einem CIM-System benötigten Daten wurde bereits vielfach hervorgehoben. Dieser Beitrag möchte auf neue Entwicklungen im Datenbankbereich, nämlich aktive Datenbanken, aufmerksam machen und zeigen wie diese vorteilhaft für die Realisierung von CIM-Systemen eingesetzt werden können.

1 Einleitung

Computer Integrated Manufactoring (CIM) zielt auf die umfassende Nutzung moderner Informationstechnologie zur Integration aller betriebswirtschaftlichen Abläufe und produktionstechnischen Prozesse, um die Nachteile durch die Brüche im Informationsfluß zu überwinden. Diese Nachteile sind vor allem die Redundanz der gespeicherten Daten, der Verlust der Daten-Integrität, Effizienzprobleme, Fehler durch manuelle Intervention und die fehlende Integrierbarkeit von Daten für Steuerungs- und Planungszwecke. Die betriebswirtschaftliche Notwendigkeit von CIM-Systemen soll hier nicht erörtert werden. Wir wollen lediglich darauf hinweisen, daß z.B. Scheer [Sch90] die Einführung von CIM für viele Unternehmen als Überlebensfrage sieht.

Die informationstechnische Umsetzung von CIM erfordert die Integration von traditionellerweise in Form von Insellösungen für Teilbereiche entwickelte Systeme wie CAD/CAM oder PPS. Eine besondere Rolle kommt dabei der integrierten Datenhaltung zu. Für diese Zwecke bietet sich natürlich in erster Linie der Einsatz von Datenbanken an, die ja bereits in der Vergangenheit sehr erfolgreich für die Integration der Daten aus den betriebswirtschaftlichen Abläufen herangezogen wurden. Datenbanken verwalten zentral alle relevanten Unternehmensdaten, gewährleisten so eine möglichst redundanzfreie und konsistente Datenhaltung und sorgen durch ihr inhärentes Transaktions- und Recoverykonzept für konsistenzerhaltenden Mehrbenutzerbetrieb sowie für hohe Datensicherheit. Zentrale Verwaltung der Daten heißt dabei nicht notwendigerweise, daß alle Daten in einer zentralen Datenbank gespeichert sein müssen, vielmehr können natürlich verteilte Datenbanken oder föderative Datenbanken, die in enger Verbindung stehen, eingesetzt werden. Für eine eingehendere Diskussion des Einsatzes von Datenbanken für CIM verweisen wir auf [Sch90, Spu92, GRW91].

Allerdings sind herkömmliche Datenbanken passiv, d.h. aufgrund von Anforderungen von Anwendungsprogrammen speichern oder retournieren sie Daten, können jedoch nicht selbst die Initiative ergreifen. Aktive Datenbanken überwinden diese Begrenzung und erlauben es, in der Datenbank selbst datenändernde Operationsfolgen zu definieren. Damit können Funktionen, die die Integrität der Daten in der Datenbank gewährleisten bzw. einen integeren Datenbankzustand bei Verletzung der Integrität wiederherstellen, aus den einzelnen Applikationsprogrammen herausgelöst werden und einmal zentral definiert werden. Damit ist es im logischen Schema der Datenbank nunmehr möglich nicht nur (deklarativ) zu definieren, was ein korrekter Datenbankzustand ist, sondern auch anzugeben, wie ein solcher wiederhergestellt wird. Darüberhinaus kann der aktive Teil einer Datenbank für datengetriebene Steuerungsaufgaben verwendet werden, was wiederum die Applikationsprogramme entlastet und die Organisation der Steuerung des Informationsflusses strafft. Durch die Entlastung der Applikationsprogramme von diesen Aufgaben steigt natürlich die Produktivität der Programmerstellung und sinkt der Wartungsbedarf von Anwendungsprogrammen. Gleichzeitig wird durch die Trennung von Steuerung und Verarbeitung die flexiblere Kombination der Programme erhöht.

Ziel dieses Beitrages ist es, das Konzept der aktiven Datenbanken kurz zu erläutern und aufzuzeigen, welche Aspekte von CIM-Systemen mit aktiven Datenbanken realisiert werden können, und zu diskutieren, welche Vorteile die Verwendung von aktiven Datenbanken für die Entwicklung von CIM-Systemen verspricht.

2 Aktive Datenbanken

Konventionelle Datenbanken sind passiv, das heißt, Anfragen und Veränderungen werden nur durch explizite Aktionen eines Benutzers oder Anwendungsprogramms durchgeführt. Aktive Datenbanken hingegen erlauben die Spezifikation von Aktionen, die automatisch ausgeführt werden, wenn bestimmte Ereignisse eintreten. Die grundsätzlichen Elemente der Architektur von aktiven Datenbanken sind in Abbildung 1 dargestellt.

Das Datenbank-Managementsystem einer aktiven Datenbank enthält einen Mechanismus zum Erkennen von Ereignissen, zum Prüfen von Bedingungen und zum Durchführen bzw. Anstoßen von Aktionen. Bei allen Abfragen und Veränderungen, die in der Datenbank eintreffen, wird überprüft, ob eine Aktion durchgeführt werden soll; diese Aktion kann wieder Veränderungen in der Datenbank auslösen oder eine Aktion außerhalb der Datenbasis anstoßen.

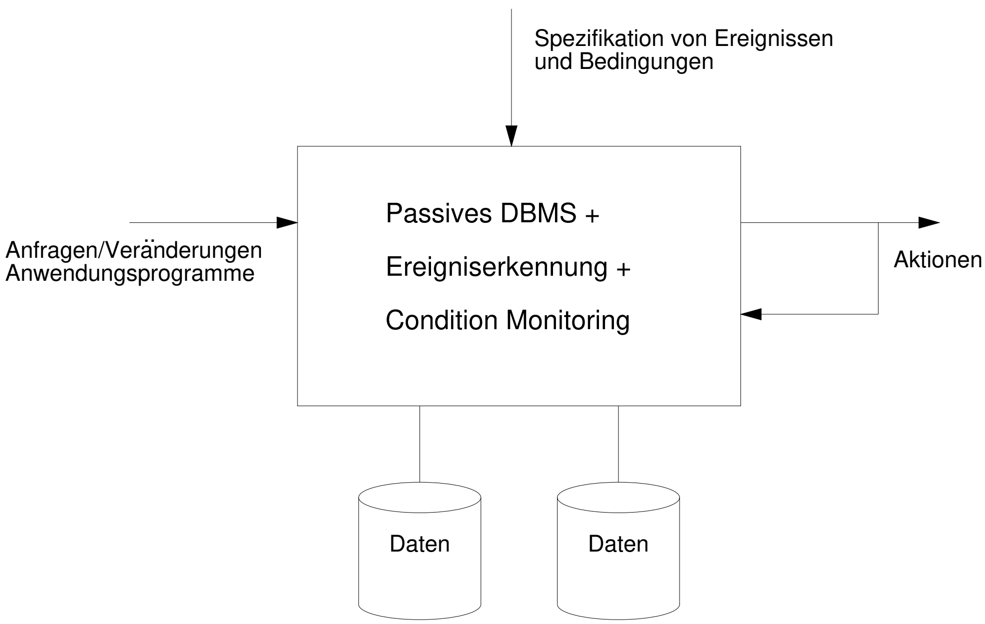

Abb. 1 Prinzip einer aktiven Datenbank

Die Spezifikation von Ereignis, Bedingungen und Aktionen erfolgt deklarativ in Form von Event-Condition-Action (ECA) Regeln.

Jeder Zugriff auf die Datenbank durch einen Benutzer oder ein Anwendungsprogramm wird als Ereignis aufgefaßt, das eine Regelanwendung anstoßen kann. Bei Triggerung einer Regel durch ein Ereignis, werden die Bedingungen dieser Regel überprüft. Sind sie erfüllt, werden die Aktionen der Regel ausgeführt.

Bedingungen sind dabei Beschreibungen von Zuständen in der Datenbank. Aktionen können wiederum Veränderungen in der Datenbank sein, aber auch Aufrufe von externen Prozeduren. Im folgenden wird für Beispiele die Starburst Rule Language [WCL91] verwendet, die Syntax einer Regel sieht folgendermaßen aus:

`create rule` *name* `on` *table*
`when` *event*
`if` *condition*
`then` *action*

Mit `create rule` wird eine Regel *name* definiert, die auf Veränderungen der Tabelle *table* reagiert. Das die Regel triggernde Ereignis wird nach dem Schlüsselwort `when` spezifiziert, der Bedingungsteil nach dem Schlüsselwort `if` enthält eine quantifizierte SQL-Abfrage, die einen bool'schen Wert retouniert. Als Aktionen (nach `then`) sind in SQL formulierte Datenbankaktionen zulässig.

Beispiel 1. Bei Veränderung des Lagerbestandes unter einen festgesetzten Wert soll automatisch eine Bestellung generiert werden. Dazu wird eine Regel formuliert, die bei Veränderungen der Bestände getriggert wird und beim Unterschreiten des Schwellwertes den Teil in eine Bestelliste einträgt.

```
create rule store_control on store
when updated(quantity),
if 'exists (select * from new_updated()
           where quantity < 5)',
then 'insert into order_list values (art_no, ...)'
```

Diese Regel reagiert auf Veränderungen in der Tabelle `store`, das triggernde Ereignis ist eine Veränderung des Attributs `quantity` bei einem Tupel dieser Tabelle. Die Bedingung ist eine SQL-Abfrage, die überprüft, ob die neue `quantity` kleiner `5` ist. Trifft dies zu, wird eine Eintragung in die Tabelle `order_list` durchgeführt.

Mit konventionellen Programmen lassen sich diese Funktionalitäten freilich ebenfalls implementieren, es stehen dabei grundsätzlich zwei Möglichkeiten mit jeweils bedeutenden Nachteilen zur Auswahl:

1. Ein spezielles Anwendungsprogramm stellt regelmäßig Anfragen an die Datenbank (engl. polling), registriert Veränderungen und reagiert dementsprechend. Dieses Anwendungsprogramm kann ein konventionelles Programm, ein Regelinterpreter oder ein Expertensystem sein.

   Dieser Ansatz ist aus mehreren Gründen unzureichend: Wenn die Abfragefrequenz zu gering ist, dann könnte damit die Anwortzeit für manche Applikationen zu hoch sein, da Veränderungen in der Datenbank zu spät bemerkt werden bzw. sogar Änderungen verloren gehen. Ist die Abfragefrequenz zu hoch, wird der Rechner ständig - auch wenn kein Ereignis vorliegt - unnötig stark belastet.

   Beispiel 2. Wenn die Temparatur an einer Meßstelle nur kurzzeitig über einen Grenzwert steigt, kann bei zu niedriger Abtastfrequenz diese Situation unbemerkt bleiben und ein Eingreifen bzw. eine spätere Diagnose vereitelt werden.

2. Die zweite Möglichkeit besteht darin, die Anwendungsprogramme, die Datenbankänderungen durchführen, so zu erweitern, daß diese Programme bei bestimmten Situationsveränderungen die entsprechenden Aktionen ausführen.

   Dieser Ansatz ist unzureichend, da gleiche Funktionalität auf mehrere Programme verteilt wird, jede Veränderung dieser Funktionalitäten müßte in allen Applikationen nachgeführt werden. Dies führt zu einem hohen Wartungsaufwand.

   Beispiel 3. Mehrere Programme ändern den Lagerbestand, jedes dieser Programme müßte kontrollieren, ob der Mindestbestand unterschritten wurde (siehe dazu Bsp. 1).

Die Durchführung einer Änderung eines Datums in der Datenbank kann nun verschiedene Arten von Aktionen triggern, die im folgenden in vier Klassen eingeteilt werden:

1. Die Überprüfung, ob diese Veränderung (oder Eingabe) korrekt ist, d.h. ob sie bestimmten, vorher definierten Bedingungen gehorcht.

2. Auslösung von Aktionen, die durch die Veränderung des Datenbankzustands notwendig, bzw. möglich geworden sind.

3. Auslösung von zeitlich verzögerten Aktionen oder Überprüfung von zeitbezogenen Constraints.

4. Überwachen der Veränderungen zum Aufspüren atypischer Zustände (Monitoring).

Entsprechend dieser Einteilung soll im nächsten Abschnitt der Einsatz von aktiven Datenbanken für typische Anwendungen in CIM-Systemen unersucht werden.

Den Vorteilen von aktiven Datenbanken stehen natürlich auch gewisse Nachteile gegenüber. Zum einen ist die Verarbeitung der Regeln eine zusätzliche Belastung für den Datenbankserver. Aktive Datenbanken stellen daher größere Anforderungen an die Rechnerleistung der Server. Zum anderen ist die Strukturierung sehr großer Regelmengen noch nicht wirklich zufriedenstellend gelöst.

3 Einsatz von aktiven Datenbanken für CIM

3.1 Integritätsbedingungen

Aktive Datenbanken erlauben eine einfache Spezifikation von Integritätsbedingungen, die bei jeder Veränderung in der Datenbank überprüft werden. Falls die durchgeführte Modifikation nicht den definierten Bedingungen genügt, wird die Änderung zurückgewiesen oder aber selbständig - durch die Regelaktionen - ein integrer Zustand hergestellt (change propagation).

In herkömmlichen, passiven Datenbanken müssen die Integritätsbedingungen in allen Applikationen überprüft werden, d.h. jedes einzelne Anwendungsprogramm enthält Teilfunktionen, die für die Erhaltung der Integrität sorgen. Das führt natürlich zu größerem Aufwand bei der Entwicklung von Anwendungsprogrammen und ist eine nicht unbeträchtliche Fehlerquelle, da die Nichtbeachtung der Integritätsbedingungen in einem einzigen Programm einen unkorrekten Datenbankzustand zur Folge haben kann, was sich wieder auf das korrekte Funktionieren der anderen Programme auswirken kann. Weiters erhöht es den Wartungsaufwand bei Änderung von Integritätsbedingungen, weil in diesem Fall alle Programme untersucht und ggf. geändert werden müssen. Darüberhinaus wird in passiven Datenbanksystemen die Integrität von Manipulationen, die mit Endbenutzerwerkzeugen wie interaktivem SQL direkt in der Datenbank durchgeführt werden, überhaupt nicht überprüft. In aktiven Datenbanken wird die Integrität nach *jeder* Änderung kontrolliert.

Beispiel 4. Die folgende Regel ist ein Beispiel für die Wiederherstellung der referentiellen Integrität. Es wird spezifiziert, daß bei Löschung eines Departments auch alle Mitarbeiter dieses Departments gelöscht werden.

```
create rule cascade on dept
when deleted,
then 'delete from emp
   where emp.deptno in
       (select dno from deleted)'
```

In den Standard SQL-92 [Ins92] wurden bereits Sprachkonstrukte zur Spezifikation von Constraints aufgenommen. Die `check` und `references` Konstrukte dienen in erster Linie zur Überprüfung und gegebenenfalls Zurückweisung unerlaubter Zustände. Die Propagierung von Veränderungen mittels benutzerdefinierter Funktionen ist noch nicht möglich.

3.2 Steuerung von nichtlinearem Datenfluß

Werden Daten nicht linear von einer Verarbeitungseinheit an genau eine nachfolgende weitergereicht, spricht man von nichtlinearem Datenfluß. Im Bereich von CIM ist dies äußerst häufig anzutreffen. Abb. 2 zeigt den Datenfluß bei der Auftragsbearbeitung, wo zum Beispiel die Erstellung der Fertigungsaufträge an eine Reihe von Vorbedingungen geknüpft ist und die Ausgangsdaten an mehrere Module weitergeleitet werden.

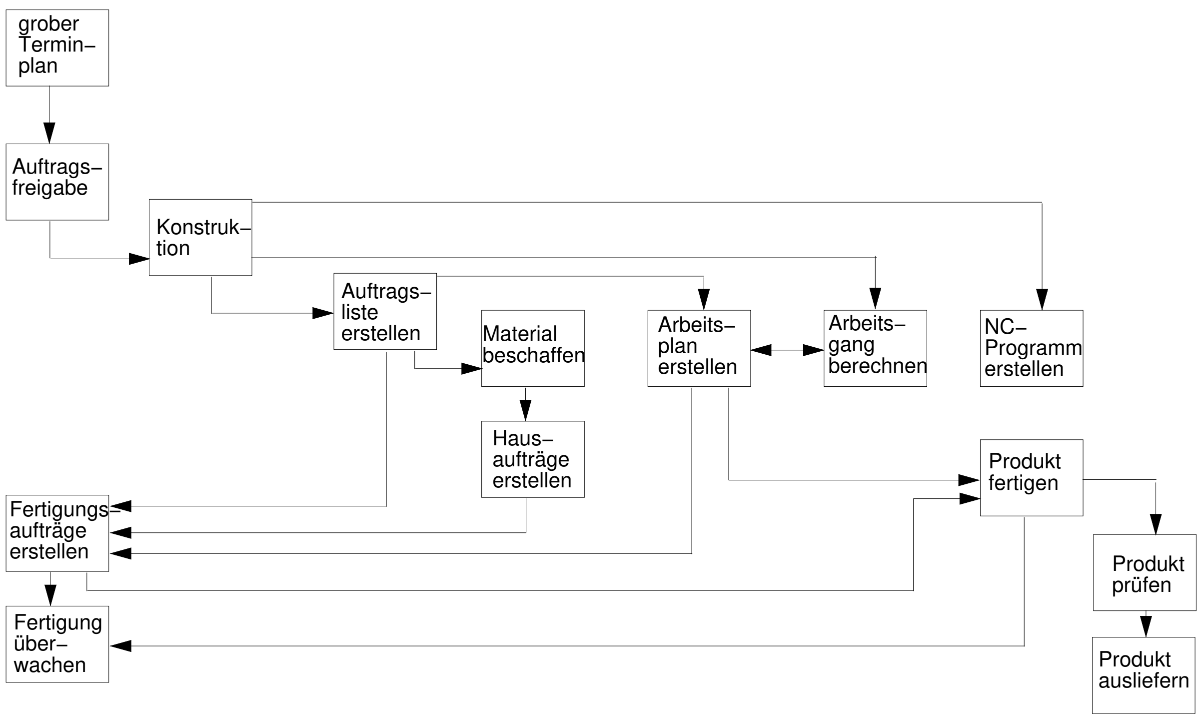

Abb. 2 Nichtlinearer Datenfluß bei Auftragsbearbeitung (aus [Sch90])

Das Zusammenspiel von CIM-Komponenten ist darüberhinaus nur durch flexible Weitergabe von Information (Bestellungen, 'Laufzettel', NC-Programme, Arbeitspläne) zu bewältigen. Abhängig von Ressourcenauslastungen, zeitlichen Kriterien, Materialverfügbarkeit, usw. müssen die Abläufe ständig flexibel gehalten werden. Das heißt, daß permanent Schedulingentscheidungen getroffen werden müssen.

Die herkömmliche Technologie bietet zwei Möglichkeiten der Implementierung:

1. Erweiterung jedes Prozesses um eine Überprüfung, ob ein Folgeprozeß gestartet werden kann.

2. Der Folgeprozeß pollt.

Jede dieser Methoden besitzt somit wieder die bereits erwähnten Nachteile.

Aktive Datenbanken eignen sich hervorragend als Scheduler. Die zugrundeliegende Datenbank dient zur Haltung aller relevanten Informationen, die einzelnen Prozesse kommunizieren über Veränderungen in der Datenbank miteinander. Die Regeln der aktiven Datenbank erkennen, wann alle Voraussetzungen (Eingangsdaten) für die Durchführung eines Prozesses vorliegen, und stossen ihn an. Die einzelnen Verarbeitungskomponenten können dabei von Steuerungsfunktionen entbunden werden. Damit wird außerdem eine sehr viel größere Flexibilität in der Konfigurierung und Integration dieser Verarbeitungskomponenten zu CIM-Systemen erreicht. Die Steuerung erfolgt durch asynchrone Ereignisse und Datenfluß, wobei natürlich auch explizite Steuerinformation (wie z.B. manuelle Intervention durch Benutzer) in der Datenbank und somit auch in der Regelmenge modelliert werden kann.

Die aktive Datenbank dient als zentrales System zum Informationsaustausch zwischen den einzelnen Komponenten, dies bedeutet eine Zentralisierung der Steuerungsinformation an einem Ort. Dieses Konzept entspricht dem sogenannten CIM Handler, wie in [Sch90] vorgeschlagen.

Beispiel 5. Im Fertigungsprozeß ist oft ein Schritt (vor allem Montagearbeiten) von der Fertigstellung mehrerer vorangehender Schritte abhängig. Die unten formulierte Regel stößt bei Vorhandensein aller Voraussetzungen den nächsten Schritt in einem Arbeitsplan an:

```
create rule next_step on prod_status
 when updated(status),
 if 'exists (select superpart from new_updated() as new, part_of as pf
             where new.status = 'finished' and
                   pf.subpart = new.part) and
    not exists (select * from part_of, prod_status
                where superpart = pf.superpart and
                      subpart = part and
                      status <> 'finished')',
 then execute 'start_process(assemble(superpart))'
```

Die Regel triggert bei Veränderung des Attributs `status` in der Tabelle `prod_status` Wenn der Status eines Teils den Wert `finished` erhält, wird untersucht, ob alle anderen Teile der übergeordneten Komponente fertig sind. Wenn ja, wird eine Aktion ausgelöst.

3.3 Zeitbezogene Constraints

Der einheitliche Mechanismus der Event-Condition-Action Regeln eignet sich auch für die Steuerung zeitabhängiger Tätigkeiten und die Behandlung zeitbezogener Konsistenzbedingungen. Dadurch brauchen verschiedene Arten von Constraints nicht mehr mit verschiedenen Systemen realisiert werden.

Beispiele dafür sind:

- Anstoß zu regelmäßigen Wartungsarbeiten, die Wartungsintervalle können in einer Datenbank gespeichert sein.

- Kontrolle von Sollzeiten: Wenn zu Bestellungen die erwartete Lieferzeit vermerkt wird, kann bei nicht termingerechtem Eintreffen eine entsprechende Meldung erzeugt werden.
- Ebenso können Durchlaufzeiten im Produktionsprozeß kontrolliert werden.
- Viele an fixen Zeiten - z.B. Monatsanfang - durchzuführenden Tätigkeiten, manipulieren in einer Datenbank gespeicherte Informationen, der Anstoß bzw. die Durchführung dieser Tätigkeiten kann von einer aktiven Datenbank übernommen werden.

3.4 Monitoring

Ein wesentlicher Bestandteil von CIM ist die mit umfangreicher werdender Automatisierung an Bedeutung gewinnende Überwachung der Abläufe. Ist eine Reihe wichtiger Unternehmensdaten (z.B. alle Informationen über Aufträge oder Lagerbestände) in einer Datenbank gespeichert, kann leicht eine Zustandsüberwachung für diesen Bereich realisiert werden.

Eine automatische Betriebsdatenerfassung und -verdichtung kann mit aktiven Datenbanken realisiert werden. Atypische Zustände können sofort bei ihrem Auftreten erkannt, gemeldet oder behoben werden. Zwei Hauptanwendungen können unterschieden werden:

1. Kontrolle von Solldaten, z.B.: Sollzeiten, Lagermengen, Kontostände.
2. Datenverdichtung für Management-Informationsysteme: z.B. Summe der Auftragswerte, Erstellung von regelmäßigen Statistiken.

4 State of the Art - Prototypen

Derzeit sind die Konzepte von aktiven Datenbanken vor allem in Forschungsprototypen implementiert. Außerdem sind bereits erste Produkte am Markt erschienen. Die einzelnen Ansätze sind sehr unterschiedlich bezüglich ihrer Ziele, der Mächigkeit, sowie der Syntax und Semantik der Regeln. In erster Linie handelt es sich um Erweiterungen relationaler Datenbanken.

Eine Vereinheitlichung der verschiedenen Formalismen ist durch die Standardisierung von SQL3 zu erwarten, wo Konstrukte zur Definition von Triggern aufgenommen werden sollen [MS93].

Im folgenden sollen die Systeme POSTGRES, Starburst, ODE und INGRES kurz beschrieben werden:

POSTGRES ist ein Datenbanksystem, das aufbauend auf INGRES an der University of California, Berkely, entwickelt wurde [SHP89] und mehrere Ziele verfolgt: Unterstützung für komplexe Objekte, benutzerdefinierte Datentypen und Zugriffsoperationen, Sprachelemente für aktive Datenbanken, Inferenzmechanismen für forward- und backward-chaining. POSTGRES ist vollständig implementiert, Design dieser Implementierung und dabei gewonnene Erfahrungen sind in [SRH90] beschrieben.

Ebenfalls zu einer aktiven Datenbank erweitert wurde das relationale Datenbanksystem Starburst ([WF90], [WCL91]). Die Besonderheit dieses Systems ist die mengenund nicht tupelorientierte Abarbeitung der Regeln. Dies ermöglicht einfachere Optimierung und effiziente Berechnung.

Ein Beispiel für die Integration von aktiven Elementen in eine objektorientierte Datenbank ist ODE [GJ91]. Die Regeln sind hier Teil von Objektdefinitionen und werden in Trigger und Constraints unterschieden, beziehen sich allerdings immer nur auf ein - wenngleich auch komplexes - Objekt.

INGRES [ING90] ist ein Beispiel für ein kommerzielles Datenbanksystem, das einen Regelmechanismus enthält. Die Regel in folgendem Beispiel feuert, wenn das Attribut `salary` in der Relation `employee` verändert wird, und ruft die Prozedur `check_sal` auf:

```
create rule emp_salary after update(salary) of employee
execute procedure check_sal
(name=old.name, oldsal = old.salary, newsal = new.salary)
```

Als Events sind Update, Delete und Insert erlaubt, als Bedingungen alles, was in einem where-Ausdruck eines SQL Select Statements zulässig ist. Es kann dabei sowohl auf die Tupel des Zustands vor der Veränderungsoperation als auch auf die neuen – veränderten – Tupel zugegriffen werden.

Aktionen sind in Prozeduren verpackt, die prozedurale Kontrollstrukturen enthalten können. Veränderungen in der Datenbank und die Ausgabe von Meldungen sind möglich. Regeln, die der Integritätskontrolle dienen, können Fehlermeldungen erzeugen.

5 Zusammenfassung

Aktive Datenbanken erscheinen als eine sehr gut geeignete Basistechnologie um komplexe CIM-Systeme zu realisieren. Da Datenbanken die gesamten Unternehmensdaten integriert verwalten, sind sie auch der Ort an dem die verschiedensten Informationsstränge zusammenlaufen. Es ist nur natürlich, daß dies auch der Ort sein soll, an dem Steuerungsentscheidungen getroffen werden sollen. Durch die Einbeziehung der Steuerung des Informationsflusses in die Datenbanken können die einzelnen verarbeitenden Komponenten von Steuerungsaufgaben entlastet werden. Das hat den Vorteil, daß Applikationsprogramme einfacher erstellt werden können, der Wartungsaufwand bei Änderung der Steuerungsfunktionen wesentlich geringer ist und somit die Applikationsprogramme wesentlich flexibler kombiniert werden können. Die im herkömmlichen Ansatz auf viele Programme verteilten Steuerungsfunktionen werden durch die Verwendung von aktiven Datenbanken an einer Stelle integriert angegeben und verwaltet. Dies führt ebenfalls zu einer größeren Flexibilität und Übersichtlichkeit. Diese Vorteilen stehen - wie in der Technik nicht anders zu erwarten - auch gewisse Nachteile gegenüber. Die Zentralisierung der Steuerungsfunktion belastet natürlich die Datenbankserver stärker als bei rein passiven Datenbanken. Ein weiteres Problem ist darin zu sehen, daß es bis jetzt noch keine wirklich zufriedenstellenden Methoden für die Strukturierung sehr großer Regelmengen gibt. Da Konzepte aktiver Datenbanken zunehmend in neuere Versionen weitverbreiteter Datenbankmanagementsysteme integriert werden, ist in naher Zukunft mit großer Verfügbarkeit zu rechnen. Die Entwicklung komplexer leistungsfähiger CIM-Systeme sollte davon profitieren.

Literaturverzeichnis

[GJ91] N. Gehani and H.V. Jagadish. Ode as an Active Database: Constraints and Triggers. In *Proc. of the 17th VLDB*, 1991.

[GRW91] Ch. Gierlinger, W. Retschitzegger, and R. R. Wagner. Datenbanken für CIM. In P. Kopacek, editor, *Tagungsband der 7. österreichischen Automatisierungstagung*, 1991.

[ING90] INGRES. *Ingres/SQL Reference Manual*, 1990.

[Ins92] American National Standard Institute. Database Language SQL. X3.135-1992, 1992.

[MS93] Jim Melton and Alan R. Simon. *Understanding the new SQL*. Morgan Kaufmann, 1993.

[Sch90] A.-W. Scheer. *CIM - Der computergesteuerte Industriebetrieb*. Springer-Verlag, 1990.

[SHP89] M. Stonebraker, Marti Hearst, and Spyros Potamianos. A Commentary on the Postgres Rules System. *Sigmod Record*, 18(3), September 1989.

[Spu92] G. Spur. *Datenbanken für CIM*. Springer-Verlag, 1992.

[SRH90] Michael Stonebraker, Lawrence A. Rowe, and Michael Hirohama. The Implementation of Postgres. *IEEE Transactions on Knowledge and Data Engineering*, 2(7):125–142, March 1990.

[WCL91] Jennifer Widom, Roberta Jo Cochrane, and Bruce G. Lindsay. Implementing Set-Oriented Production Rules as an Extension to Starburst. In *Proc. of the 17th VLDB*, 1991.

[WF90] Jennifer Widom and Sheldon J. Finkelstein. Set-Oriented Production Rules in Relational Database Systems. *SIGMOD-90*, 1990.

Die Autoren

Dr. Herbert Groiss ist Universitätsassistent am Institut für Informatik der Universität Klagenfurt. Schwerpunkte seiner Forschung sind aktive und deduktive Datenbanken, sowie die Integration von Technologien der Künstlichen Intelligenz in Informationssysteme.

Dr. Johann Eder ist Ordentlicher Universitätsprofessor für Betriebliche Informations- und Kommunikationssysteme am Institut für Informatik der Universität Klagenfurt. Die Schwerpunkte seiner Forschungs- und Beratungstätigkeit liegen in der Weiterentwicklung der Datenbanktechnologie und in ihrer Nutzbarmachung für betriebliche Anwendungsfelder durch Integration in Informations- und Kommunikationsysteme, sowie in der Planung, Modellierung und Entwicklung von Informationssystemen.