In: CIM-Expertenwissen für die Praxis, H. Krallmann (Hrsg.), Oldenbourg-Verlag, 1990, pp. 265-277.

Die Rolle von Datenbanksystemen in CIM

Theo Härder
Universität Kaiserslautern

Datenbanksysteme können die Rolle eines Integrationsvehikels in einer CIM-Umgebung spielen. Dabei sollten alle Vorteile des Datenbankkonzeptes, das sich bereits in vielen zentralen Systemen in der Praxis bewährt hat, erhalten bleiben. Der Autor skizziert einen Weg, wie Datenbanksysteme weiterzuentwickeln sind, um in heterogenen und verteilten CIM-Umgebungen mit zeitkritischen und komplexen Verarbeitungsanforderungen bestehen zu können.

Anforderungen an die Datenhaltung

"Computer-Integrated Manufacturing ist die umfassende Nutzung moderner Informationstechnologie für die Gestaltungs- und Umwandlungsprozesse in der Entwicklung, Konstruktion und Fertigung und für deren Planung und Kontrolle" [1]. Im Kern bedeutet dies, daß ein zu fertigendes Produkt von seiner ersten Entwurfsphase, über die Planung der Fertigung, der Fertigungsausführung bis zum Versand durch ein integriertes EDV-System begleitet wird [2]. Eine sachgerechte Realisierung dieser Zielvorstellung setzt voraus, daß das Produkt mit seiner gesamten Konstruktions-, Planungs- und Fertigungsumgebung (alle operationalen Daten des Betriebs) als rechnerinternes Modell mit hoher Genauigkeit und großem Detaillierungsgrad nachgebildet wird. Der Intergrationsgedanke legt es ferner nahe, daß alle Modelloperationen in einheitlicher Weise und konsistenzerhaltend durchgeführt werden sollten, was wiederum dem Datenbankkonzept eine herausgehobene Rolle bei der CIM-Realisierung zuweist.

Sichtet man die Anforderungen an die Datenhaltung in einer CIM-Umgebung [3,4], so ergibt sich eine recht stattliche Wunschliste, die hier nicht erschöpfend dargestellt werden kann. Einige wesentliche Aspekte seien hier skizziert:

- Integration von betriebswirtschaftlichen und technischen Daten
- Handhabung komplexer Objekte
- Darstellung mehrerer Beschreibungsformen (Partialmodelle) eines Objektes
- Unterstützung des Konstruktionsvorganges durch Versionen und Alternativen
- Kontrolle komplexer, ggf. bereichsübergreifender Integritätsbedingungen
- Verteilte Verarbeitung der Daten in einer heterogenen Rechnerumgebung
- Gewährleistung von Zeitbedingungen (Realzeitaspekte).

Diese breit gestreuten Anforderungen sind zu erklären aus den speziellen Bedürfnissen und der Vielfalt der Anwendungen [2], deren Funktionalität nun im Rahmen des CIM-Konzeptes in einheitlicher Weise bereitgestellt werden soll. Bisher wurden solche Funktionen nur teilweise erbracht durch zugeschnittene Anwendungsprogramme oder spezielle Datenhaltungssysteme, die gezielt, aber auf Kosten der Flexibilität und Unabhängigkeit die Besonderheiten der Anwendungsumgebung bei der Implementierung ausnutzen. Auf diese Weise entstanden die vielzitierten Insellösungen innerhalb der betrieblichen Datenhaltung, die einen funktionsgerechten Zugriff oder Austausch der Daten behinderten oder oft unmöglich machten. An eine wie auch immer geartete Integration aller betrieblichen Abläufe über die Daten ist hierbei nicht zu denken.

Welcher Weg zur Integration verspricht nun im Rahmen von CIM die größten Erfolgsaussichten? Da die operationalen Daten Modelle der Objekte und Ereignisse der Miniwelt "Betrieb" verkörpern, die es durch mannigfache (zeitgerechte) Entwurfs- und Aktualisierungsoperationen (-transaktionen) in Übereinstimmung mit dem betrieblichen Geschehen zu halten gilt, fällt der Datenhaltung bei der Integration die wichtigste Rolle zu. Eine integrierte Verarbeitungsumgebung für CIM sollte die Durchgängigkeit aller angewendeten Konzepte erlauben und sämtliche Phasen der Fertigungsvorbereitung und Fertigung unterstützen. Dazu ist es erforderlich,

- allen Anwendungen eine Teilsicht auf die verschiedenartigen Aspekte eines Entwurfs- und Fertigungsobjektes (vereinheitlichtes konzeptuelles DB-Schema) zu bieten
- Konsistenz- und Integritätsbedingungen auch phasenübergreifend definieren und kontrollieren zu können
- ein hohes Maß an Flexibilität und Erweiterbarkeit (durch ein geeignetes Datenmodell) gewährleisten zu können.

Wege zur Integration

In [5] werden verschiedene Integrationsstufen der Datenhaltung zur Kopplung der Anwendungen (Entwurfsschritte/Werkzeuge) diskutiert. Es ist offensichtlich, daß sich die obigen Kernanforderungen weder durch isolierte Dateischemata noch durch anwendungsbezogene Dateikonversion erfüllen lassen. Um gewachsene Anwendungen über gemeinsame Datenstrukturen verknüpfen zu können, wird die sternförmige Konversion (Bild 1a) vorgeschlagen, wodurch für die einzelnen Anwendungen spezielle Sichten aus anwendungsunabhängigen Dateiformaten erzeugt werden.

Sternförmige Konversion

Dieser Konverter-Ansatz wird sicherlich momentan durch den Markt erzwungen und gilt zumindest für CAD-Systeme als "state of the art" [5]. Einher gehen damit eine Reihe von Normierungsbemühungen für die vielfältigen Schnittstellen (IGES, STEP usw.), die wegen der raschen Entwicklung der Technologie und des zusätzlichen Bedarfs an Funktionalität manchmal schon zum Zeitpunkt des Inkrafttretens veraltet sind. Recht kritisch zu sehen ist bei diesem Ansatz auch die Frage der Redundanzerkennung und -wartung bei inhaltlich überlappenden Dateischemata. Weiterhin wird dadurch ein Verlust an Semantik in Kauf genommen, da für die Anwendung nur syntaktische Zusammenhänge spezifiziert werden; beispielsweise wird nur die Objektgeometrie (z.B. Zylinder) beschrieben, seine technische Bedeutung (z.B. Welle mit bestimmten Oberflächeneigenschaften und Toleranzen) geht jedoch verloren [6]. Durch die Datenorganisation und ihre anwendungsbezogene Konvertierung werden bereichsübergreifende Kontrollen und Prüfungen von Integritätsbedingungen verhindert; z.B. sollte es möglich sein, während des Entwurfs zu überprüfen, ob ein Werkstück (unter Nutzung der vorhandenen Maschinen und der vorrätigen Standardteile) auch gefertigt werden kann. Schließlich ist ein Verlust an systembezogenen Optimierungsmöglichkeiten festzustellen, da durch die selektive Konversion der ganzheitliche Objektbezug und der ursprüngliche Verarbeitungskontext verlorengeht.

Neben den diskutierten Unzulänglichkeiten ist als weiterer prinzipieller Mangel festzuhalten, daß beim Konverter-Ansatz die gesamte Verantwortung für die Konsistenz der Daten an die (unabhängig voneinander entworfenen) Anwendungsprogramme delegiert wird und ein Konzept zur Strukturierung und Sicherung der Verarbeitung (Transaktionskonzept) nicht zur Verfügung steht.

Datenbankkonzept

Eine weitergehende Integration mit einer gleichzeitigen Verbesserung/Behebung der angesprochenen Mängel verspricht das Datenbankkonzept, dessen generelle Stärken durch folgende Eigenschaften skizziert seien:

- explizite, klar strukturierte und redundanzfreie Beschreibung der relevanten Informationen und deren Abbildung auf ein DB-Schema (hierdurch erwartet man im wesentlichen eine Unterstützung bei der Standardisierung, dem Austausch und der Übertragung der im Ingenieurbereich meist vorherrschenden komplexen Datenstrukturen),
- angemessene Handhabungskonzepte (Auswahl- und Zugriffsoperationen) auf großen Datenmengen,
- Strukturierung und Organisation der Verarbeitung durch ein Transaktionskonzept,
  und als Konsequenz
- zentralisierte Maßnahmen zur Sicherung der Daten (Durchführen von Integritätskontrollen, Kontrolle des Mehrbenutzerbetriebs, Mitführen von Protokollinformation usw.).

Das für den Datenbankansatz fundamentale Transaktionskonzept [7] garantiert für die DB-Verarbeitung die Eigenschaften (ACID-Prinzip)

- Atomarität der Transaktionsausführung (Alles-oder-Nichts)
- Konsistenz des DB-Zustandes und der Ergebnisdaten
- isolierte Ausführung aller parallel ablaufenden Transaktionen (Serialisierbarkeit)
- Dauerhaftigkeit der Änderungen beendeter Transaktionen (Persistenz der Daten).

Es bestimmt damit einerseits wesentlich die internen Systemabläufe zur konsistenten Ausführung von Transaktionen und Sicherungsmaßnahmen für alle permanenten Daten; andererseits ist es Voraussetzung für die zuverlässige und auf die Anwendungsaspekte reduzierte Programmierung von Anwendungsprogrammen (AP). Es veranlaßt weiterhin die Maskierung aller Fehler und aller Aspekte des Mehrbenutzerbetriebs für die Anwendung, so daß einem AP stets eine fehlerfreie Einbenutzerumgebung geboten wird. Durch die COMMIT-Operation wird eine erfolgreiche Transaktion abgeschlossen.

Eine DB-Integration aller Verarbeitungsschritte/Werkzeuge bietet für das CIM-Konzept im Vergleich zum Konverter-Ansatz noch weitere Vorteile:

- Zumindest konzeptionell können alle Anwendungen von redundanzfreien Objektrepräsentationen ausgehen; jedes Objekt besitzt nur eine Objektrepräsentation, die alle seine Aspekte (Sichten, Partialmodelle) umfaßt.

- Bei der Objektspezifikation kann mehr Semantik erfaßt werden. Deshalb ist die systemseitige Kontrolle komplexer Integritätsbedingungen und bereichsübergreifender Beziehungen prinzipiell möglich.
- Bei Verarbeitungsanforderungen, die durch deskriptive Sprachen oder mächtige anwendungsbezogene Operatoren spezifiziert werden, bleibt Kontextwissen erhalten, das systemseitig zur Optimierung ausgenutzt werden kann.
- Wenn zur Leistungssteigerung oder aus Gründen der Verteilung bzw. der lokalen Verfügbarkeit Replikate oder Objektredundanz eingeführt werden, so lassen sie sich vor der Anwendung verbergen.

Bild 1 veranschaulicht das Prinzip des Datenbankansatzes und versucht, die prinzipiellen Unterschiede zum Konversionsansatz zu verdeutlichen. Zum Anschluß von abgeschlossenen Systemen (Turnkey-Systeme) kann es mit zugeschnittenem Konvertereinsatz kombiniert werden.

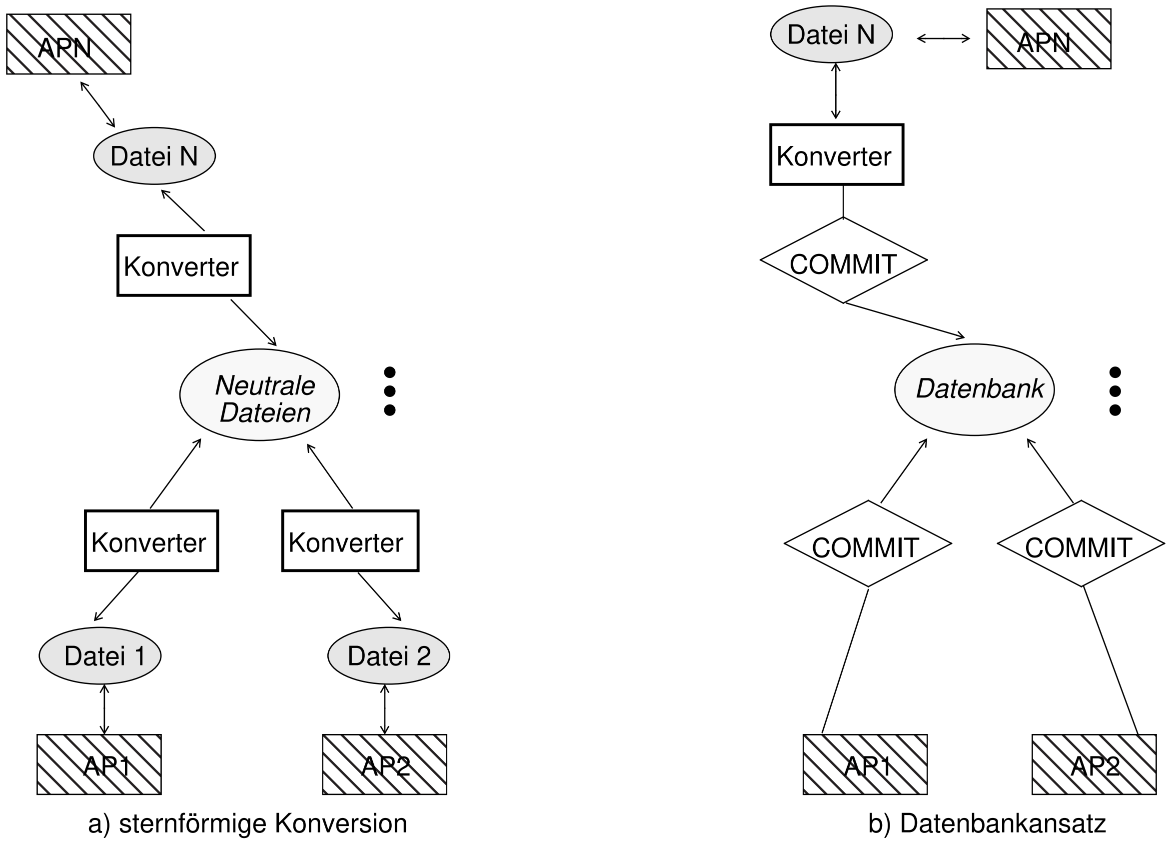

Bild 1: Wege zur Integration

Bei unserer Bewertung wurde unterstellt, daß komplex-strukturierte Objekte, wie sie in einer CIM-Umgebung zu erwarten sind, in angemessener und natürlicher Weise durch das verfügbare Datenmodell zu beschreiben sind. Solche neuen Datenmodelle müssen eine im Vergleich zu den klassischen Datenmodellen (Netzwerk-, Hierarchie- oder Relationenmodell) beträchtlich höhere Modellierungsmächtigkeit aufweisen und auch neue Verarbeitungskonzepte anbieten, um beispielsweise eine ganzheitliche Handhabung aller ein Anwendungsobjekt beschreibenden DB-Strukturen (komplexes Objekt, Molekül) zu erlauben. Klassische Datenmodelle erzwingen dagegen eine Zerlegung komplexer Objekte in eine heterogene Menge "unabhängiger" Sätze; die Rekonstruktion des komplexen Objektes hat dann navigierend durch lange Folgen von DB-Operationen seitens des Anwendungsprogrammes zu erfolgen.

Diese Vorgehensweise bei klassischen Datenmodellen resultiert in einem Modellierungs- und einem Verarbeitungsproblem [6], was in praktischen Fällen zu einem nicht tolerierbaren Leistungsverhalten führt; dies ist durch eine Reihe von Einsätzen herkömmlicher DBS bei verschiedenartigen Ingenieuranwendungen belegt [8]. Zur Lösung dieser Modellierungs- und Verarbeitungsproblematik werden zur Zeit mehrere neue Datenmodelle entwickelt, bzw. bestehende verfeinert und in neuartigen Architekturansätzen als Prototypsysteme realisiert [9,10,11]. Die dabei verwendeten DBS-Architekturen werden mit ihren Haupteigenschaften später separat diskutiert.

Das bisher skizzierte Datenbankkonzept bietet allen Anwendungsprogrammen eine logische Sicht auf alle DB-Daten; es unterstellt zunächst eine zentralisierte Datenverwaltung, eine homogene Verarbeitungsumgebung und "kurze" Transaktionen, für die die "Alles-oder-Nichts"-Eigenschaft angemessen ist. In der CIM-Welt sind solche Annahmen nicht gerechtfertigt. Bei der Vielfalt der Funktionen, ihren möglichen Lösungen und Leistungsbereichen sowie vor allem bei dem schnellen technologischen Fortschritt ist es unmöglich, ein homogenes System (ein Hersteller) vorauszu-setzen. Die Grundstruktur und Lösungsphilosophie des CIM-Konzeptes muß eine extrem heterogene Welt (Funktionen, Daten, Kommunikation) unterstellen, in der das Gesamtsystem unter Einbeziehung vorhandener Teilsysteme aufgebaut wird. Diese Grundstruktur muß sich natürlich auch in den Konzepten der Datenhaltung widerspiegeln. Wesentlich sind hierfür die Aspekte Verteilung, heterogene Verarbeitungsumgebung, Realzeitanforderung und die sich daraus ergebenden Auswirkungen auf das Transaktionskonzept.

Die verteilte Datenbank als Integrationsvehikel?

Seit etwa 10 Jahren werden verteilte Datenbanksysteme als Forschungsthema behandelt; erste (im Funktionsumfang eingeschränkte) Implementierungen sind heute bereits verfügbar. Da aus einer abstrakten Sichtweise heraus verteilte DBS die Lösung vieler Probleme der CIM-Datenhaltung versprechen, soll ihre spezielle Eignung nun näher untersucht werden.

Ein (ideales) verteiltes DBS zur Verwaltung verteilter Datenbestände ist so konzipiert, daß es jedem beliebigen Anwendungsprogramm das Bild einer zentralen Datenbank vermittelt; d.h., es hat den Eindruck, die Daten würden an einem Ort, in einem DBS und auf dem Rechner gehalten, auf dem es selbst ausgeführt wird. Die Datenbestände können dabei auf beliebig vielen verschiedenartigen Rechnern unter verschiedenen Betriebssystemen mit unterschiedlichen lokalen DBS verwaltet werden, wobei auch keine Einschränkungen hinsichtlich der verwendeten Kommunikationsprotokolle gemacht werden. Ein verteiltes DBS sollte es ferner gestatten, daß ein Objekt (Relation) partitioniert an mehreren Orten gespeichert sein kann (Fragmentierung) und daß dieselben Daten an verschiedenen Stellen (u.U. in unterschiedlichen Repräsentationen) gehalten werden können (Replikation), ohne daß solche Maßnahmen zur Steigerung der Verfügbarkeit und Leistungsfähigkeit des Gesamtsystems für das Anwendungsprogramm sichtbar werden. Ebenso hat es dafür zu sorgen, daß neben den bekannten Fehlerfällen in zentralen DBS (wie Systemcrash und Gerätefehler) zusätzlich auch Ausfälle von Knoten, Kommunikationsverbindungen usw. soweit wie möglich vor den Anwendungsprogrammen verborgen bleiben. Schließlich dürfen Auswirkungen der verteilten Verarbeitung im Mehrbenutzerbetrieb nicht bis auf die Ebene des Anwendungsprogrammes durchschlagen, da ihm wie im zentralen Fall ein "logischer Einbenutzerbetrieb"

(Serialisierbarkeit) zu garantieren ist. Kurz, ein verteiltes DBS hat dem Anwendungsprogramm gegenüber alle Aspekte der Verteilung (Orte, Fragmente, Replikate, Fehler und Nebenläufigkeit) zu verdecken.

Selbstbedienungsladen oder Fachgeschäft?

Das dem verteilten DBS zugrundeliegende Verarbeitungsmodell [12] läßt sich in Analogie mit dem eines Selbstbedienungsladens charakterisieren. Jeder Kunde kann ungestört herumstöbern und vergleichen, wobei es dann die Aufgabe des Systems ist, bei jedem "Warenzugriff" eine lokale Sicht zu vermitteln, d.h., jeden Kunden, dessen Wünsche im voraus nicht bekannt sind, ist eine aktuelle, konsistente und lokale Sicht auf alle Daten zu gewährleisten. Dieses Ubiquitätsprinzip [5] bietet einen hohen Grad an "Freizügigkeit" und verlangt keinerlei Vorplanung der Verarbeitung durch die Anwendung; es führt aber insbesondere bei nicht-lokalen Datenzugriffen und bei der Datenaktualisierung auf einem enormen Verarbeitungsoverhead (Änderungen in Fragmenten, Konsistenzerhaltung von Replikaten), der in realen Systemen kaum (in vollem Umfang) in Kauf genommen werden dürfte. In einer CIM-Umgebung, in der viele Funktionen und Datenzugriffe vorplanbar sind, erscheint das Ubiquitätsprinzip als unnötig allgemein; seine Freiheitsgrade können gar nicht angemessen genutzt werden, um die hohen Laufzeitkosten (die im übrigen der Realisierung zeitkritischer Funktionen entgegenstehen) zu rechtfertigen. Aus unserer Sicht kann deshalb die verteilte Datenbank mit dem Ubiquitätsprinzip, das sie verkörpert, nicht als Integrationsverhikel für CIM herangezogen werden.

Wie kann nun eine Integration aller CIM-Funktionen über die Daten erreicht werden? Die Vorteile des DB-Konzeptes bleiben unstrittig; sie müssen nur besser mit den CIM-Verarbeitungsanforderungen verknüpft und an die heterogene und verteilte Anwendungsumgebung angepaßt werden.

Unser Lösungsvorschlag läßt sich modellhaft und abstrakt veranschaulichen als "Fachgeschäft mit individueller Bedienung" (oder Verbund von Fachgeschäften). Das Ubiquitätsprinzip, charakterisiert durch ein globales konzeptionelles DB-Schema und beliebige DB-Anforderungen (z.B. SQL-Anweisungen) durch die Anwendungen, wird aufgegeben zugunsten eines DBS-Angebots von problemunterstützenden Funktionen, die mit Parametern versehen von den Anwendungsprogrammen aufgerufen werden. Ein solches Requestor/Server-System läßt sich wesentlich einfacher planen, auf spezielle Anforderungen zuschneiden und betreiben, da Anwendungswissen bei der Konzeption der DB-Funktionen herangezogen wird. Im Vergleich zum Ubiquitätsprinzip ist es auch wesentlich einfacher, bei Strukturänderungen oder Migrationen (Fragmente, Replikate) die notwendigen Anpassungen vorzunehmen und die Funktionen stabil zu halten, d.h., die Anwendungsprogramme von den Schemaänderungen zu isolieren. Bei diesem Ansatz ist es für die Anwendung von untergeordneter Bedeutung, ob im System ein oder mehrere Server ihre Leistungen anbieten, d.h., ob die Daten durch ein DB-Schema oder partitioniert durch mehrere unabhängige beschrieben werden. Durch die eingeschränkte Art der Datenanforderungen in solchen Serversystemen wird ein Großteil der Verwaltungs-/Organisationsprobleme von verteilten DBS vermieden.

Verarbeitungsmodell für DB-Daten

Umfangreiche Studien in den letzten Jahren [8] haben gezeigt, daß es beim Einsatz herkömmlicher DBS im Ingenieurbereich neben der Modellierungsproblematik zu einer Verarbeitungsproblematik, d.h. zu einer extrem ineffizienten DB-Verarbeitung kommt. Wie bereits erwähnt, sollen neue Datenmodelle den Modellierungsaspekt verbessern und die Darstellung und Verwaltung komplexer Objekte unterstützen. Hier soll nun ein neues Verarbeitungsmodell skizziert werden, das die Leistungsproblematik bei vielen CIM-Anwendungen lindern dürfte.

Herkömmliche DBS (am Beispiel des Relationenmodells) erlauben eine mengenorientierte Qualifikation von "Verarbeitungsgegenständen" als sogenannte Sichten, die homogene Satzmengen umfassen. Die Bereitstellung dieser qualifizierten Sätze geschieht satzweise mit Hilfe eines Cursors; Sätze werden einzeln ins Anwendungsprogramm geholt und nach ihrer Änderung direkt wieder in die DB zurückgeschrieben. Eine Lokalitätsnutzung auf größeren Verarbeitungseinheiten in Anwendungsnähe wird durch diese satzbezogene Verarbeitungslogik verhindert.

In allen Ingenieuranwendungen, die oft lange Operationsfolgen auf meist großen heterogenen und komplex-strukturierten Verarbeitungsgegenständen ausführen, ist gerade die Lokalitätsnutzung in Anwendungsnähe für das Leistungsverhalten besonders wichtig, da typischerweise einerseits ein hoher Anteil der Datenreferenzen (bei 10^5-10^7 Referenzen pro größerer CAD-Operation [8]) die gleichen Daten betrifft und andererseits die Anwendung in einer Workstation einen "weiten Weg" bis zum DBS auf einem Server-Rechner hat. Als Lösungsprinzip für die Nutzung der Referenzlokalität bei der Verarbeitung der komplexen Anwendungsobjekte wurde ihre anwendungsnahe Pufferung (Objektpuffer) vorgeschlagen [13]. Änderungen erfolgen in den Objektkopien im Puffer, was aus der Sicht des DBS dem Konzept des verzögerten Änderns (deferred update) entspricht. Am Ende der Verarbeitung (Transaktion) werden dann die akkumulierten Änderungen in die DB eingebracht. Die gravierendste Verbesserung, die sich bei diesem "externen" Verarbeitungsmodell im Vergleich zur herkömmlichen satzbezogenen DB-Verarbeitunglogik ergibt, besteht darin, daß die Anwendungsobjekte während ihrer Bearbeitung als Hauptspeicherdatenstruktur gehalten werden. Dadurch lassen sich die (wiederholten) Zugriffe des Anwendungsprogrammes (fast) unmittelbar und sehr effizient abwickeln.

Als aussichtsreiche Systemarchitektur für DB-basierte Ingenieuranwendungen wurde in den letzten Jahren vielerorts die DBS-Kern-Architektur konzipiert und als Prototypimplementierung untersucht [10, 14, 15]. Sie besteht - grob skizziert - aus einem anwendungsunabhängigen DBS-Kern, der die allgemein verwendbaren Darstellungs- und Zugriffstechniken vereinigt, und einer anwendungsbezogenen Modellabbildung, die jeweils für eine Anwendungsklasse die benötigten Objekte und Operationen an ihrer Schnittstelle (Anwendungsmodell-Schnittstelle) bereitstellt. Die Aufgabe der Systemkomponente Modellabbildung, die zur Unterstützung unseres bereits eingeführten Verarbeitungsmodells mit einem Objektpuffer ausgestattet ist, besteht nun darin, ein konkretes Anwendungsmodell auf ein allgemeines, der Komplexität der Anwendungsobjekte angepaßtes Datenmodell des Kerns abzubilden und bei der Verarbeitung die Referenzlokalität in Anwendungsnähe auszunutzen. Dieser zweigeteilte Architekturansatz läßt sich, wie in Bild 2 gezeigt, in natürlicher Weise auf eine lokal verteilte Workstation/Server-Umgebung zuschneiden. Um die rechnerübergreifende Kommunikation gering zu halten, muß neben der Nutzung der verarbeitungsinhärenten Lokalität auf Workstation-Seite der Datentransport zur Versorgung der Anwendung und zum Einbringen der Ergebnisse mengenorientiert organisiert sein. Beispielsweise wurde in [13] das Konzept einer Entwurfstransaktion vorgeschlagen, bei der durch eine oder mehrere Checkout-Operationen die benötigten Daten geholt und am Transaktionsende durch genau eine Checkin-Operation die geänderten Objekte in die DB eingebracht werden.

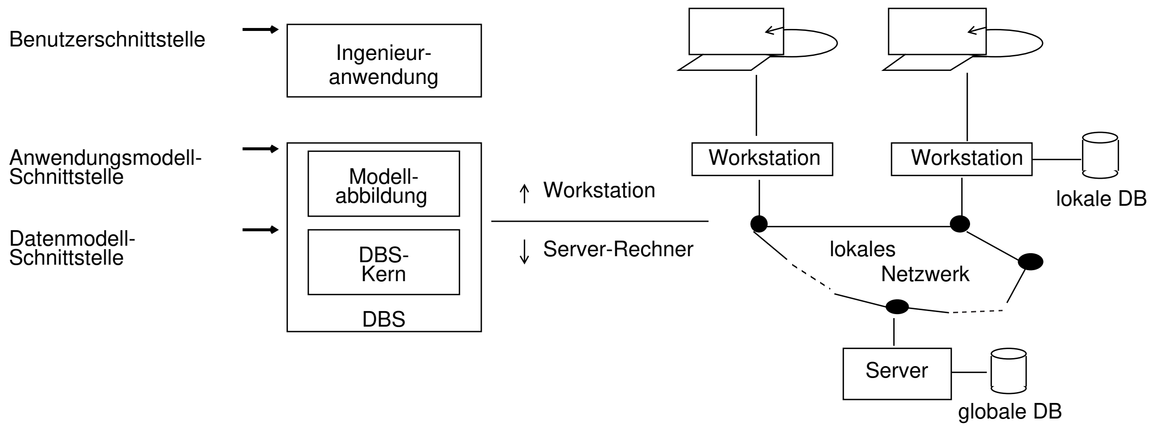

Bild 2 : Gesamtarchitektur eines DB-basierten Ingenieursystems

Das diskutierte Verarbeitungsmodell wurde ebenso wie die DBS-Kern-Architektur für die Bedürfnisse von Ingenieuranwendungen, vor allem im Rahmen von workstation-orientierten Hardware-Umgebungen, entwickelt. Bei näherer Betrachtung der CIM-Anwendungen, die künftig verstärkt (auch bei PPS-Aufgaben) in verteilten Rechnersystemen (Spezialrechner, Workstations usw.) ablaufen werden, wird deutlich, daß unser Verarbeitungsmodell auch für den Einsatz bei PPS- und CAM-Anwendungen verallgemeinert werden kann. Beispielsweise lassen sich die von der PPS-Anwendung benötigten Daten (Sichten) aus der DB kurzzeitig extrahieren und direkt den entsprechenden Planungs- oder Verwaltungsalgorithmen in der Anwendungsumgebung verfügbar machen. Auf diese Weise ist es möglich, DB- und Anwendungsverarbeitung auch räumlich zu entkoppeln und ggf. die Rechenkapazität von Arbeitsplatz- oder Abteilungsrechnern auszunutzen [16]. Ebenso läßt sich dieses Verarbeitungsmodell auf den CAM-Bereich, vor allem bei zeitkritischen Aufgaben, anwenden, um die Daten, die z.B. laut Fertigungsprogramm gebraucht werden, zur rechten Zeit als Kopie "vor Ort" zu haben. Falls die Anwendung nicht unbedingt auf die aktuelle Sicht der Daten angewiesen ist, vereinfacht sich das Problem der Datenversorgung durch das Verfügbarmachen sogenannter Schnappschüsse (snapshots) beträchtlich. Weitere Aspekte der Datenallokation und Datenaktualisierung werden in [17] diskutiert. Abschließend sei noch festgestellt, daß das vorgeschlagene Verarbeitungsmodell für den praktischen Einsatz weiterentwickelt und verfeinert werden muß. Schwierige Probleme, hervorgerufen durch externe Verarbeitung in Kopien und die Akkumulation der Änderungen, ergeben sich insbesondere bei überlappenden Sichten, beim mengenorientierten Einbringen von Änderungen und bei Konsistenzprüfungen während der Verarbeitung, die notwendigerweise nur lokal sein können.

Künftige Erweiterungen des Transaktionsparadigmas

Die Vielfalt der Aufgaben und Verarbeitungsanforderungen in einem CIM-System verbietet es, das Konzept der "kurzen" Transaktionen unmittelbar und ohne Anpassung als Kontroll- und Strukturierungsmechanismus einzusetzen. Seit Jahren schon arbeitet man in der Forschung an Erweiterungen oder Variationen des Transaktionsbegriffs. Durch geschachtelte Transaktionen soll eine sichere und effiziente Bearbeitung von Aufgaben in verteilten Systemen unterstützt werden. Zur Kontrolle von komplexen Entwurfsaufgaben (z.B. in CAD) wurden "lange" Transaktionen mit einer internen Strukturierung eingeführt [13]. Diese Vorschläge sind in mannigfacher Weise an die besonderen Bedingungen einer CIM-Umgebung, was Struktur und Verarbeitungssemantik angeht, anzupassen. Beispielsweise bedingt das Ende einer Entwurfstransaktion (COMMIT) nicht nur das Einbringen der Datenstrukturen eines neu entwickelten Werkstücks, wobei seitens des DBS als Qualitätsmaß DB-Schema-Konsistenz zu fordern ist, sondern es schließt auch die Freigabe durch eine zentrale betriebliche Instanz (z.B. durch ein Normenbüro) ein, was wiederum ein komplexer Entscheidungsvorgang ist ("Freigabe umfaßt insbesondere auch das Gültigwerden von Entwicklungsleistung" [5]).

In der wissenschaftlichen Diskussion wird heute erwartet, daß künftig der Transaktionsbegriff über sein ursprüngliches Anwendungsgebiet "DBS" hinaus eine Verallgemeinerung erfährt. Unseres Erachtens wird in nicht allzu ferner Zukunft jede "bedeutungsvolle" Ressourcennutzung in einem Rechensystem, insbesondere in einem CIM-System, als Transaktion definiert. Das gilt vor allem dann, wenn eine gemeinsame Nutzung durch mehrere Anwendungen vorliegt. Dadurch werden die Vorteile der allgemeinen Transaktionseigenschaften (ACID) jeder Anwendung zuteil. Durch Erweiterung dieser Eigenschaften könnten eine Reihe von wichtigen Vorteilen für die Systemverwaltung erzielt werden. So könnte eine Transaktion beispielsweise als Einheit der Planung bei Lastkontrolle und Optimierung, zur Abrechnung von Systemleistungen oder zur Sicherstellung der Überprüfbarkeit von Vorgängen im Sinne der Revision herangezogen werden. Durch eine interne Strukturierung der Transaktion läßt sich eine Verteilbarkeit von großen Aufgaben oder eine Verknüpfung von unabhängig ausgeführten Teilaufgaben erreichen.

Aus diesen technischen Gründen ist zu erwarten, daß in künftigen Rechensystemen, die in der Regel verteilte Systeme sein werden, nur noch Transaktionen ausgeführt werden. Sie dienen als dynamische Kontrollstrukturen und verknüpfen zusammengehörige Berechnungen in verteilten Systemen. Dabei gewährleisten sie für den Benutzer

- eine lokale Sicht auf alle Betriebsmittel
- einen logischen Einbenutzerbetrieb, d.h. Serialisierbarkeit aller abgewickelten Transaktionen
- das Verbergen von Fehlern ("Fehlerfreiheit" auf der Ebene der Programmabwicklung).

Nach unserer Einschätzung wird sich aus diesen Gründen der verallgemeinerte Transaktionsbegriff zu einem neuen Paradigma der Informatik entwickeln, das sowohl Programmierung als auch Entwurf großer Systeme tiefgreifend beeinflussen wird.

Literatur

[1] Maier-Rothe, Ch.: Computer-Integrated Manufacturing: Bedeutung, Stand und Ausblick, in: Elektronische Rechenanlagen, 27. Jg., Heft 3/1985, S. 174-180.

[2] Scheer, A.-W.: Schnittstellen zwischen betriebswirtschaftlicher und technischer Datenverarbeitung in der Fabrik der Zukunft, in: Proc. 14. GI-Jahrestagung 1984, IFB 88, Springer-Verlag, 1984, S. 56-75.

[3] Lockemann, P.C., et al.: Anforderungen technischer Anwendungen an Datenbanksysteme, in: Proc. der GI-Fachtagung "Datenbanksysteme für Büro, Technik und Wissenschaft, IFB 94, Springer-Verlag, 1985, S. 1-26.

[4] Zörntlein, G.: Flexible Fertigungssysteme: Belegung, Steuerung, Datenorganisation, Hanser-Verlag, München, 1988.

[5] Wedekind, H.: Die Problematik des Computer Integrated Manufacturing (CIM) - Zu den Grundlagen eines "strapazierten" Begriffes -, in: Informatik-Spektrum, Vol. 11, 1988, S. 22-39.

[6] Hübel, Ch., Paul, R., Sutter, B.: Datenbank-gestützte technische Modellierung - ein Ansatz für die CAD/CAP-Integration, in: CIM-Management, 4. Jg., Heft 6, 1989.

[7] Härder, T., Reuter, A.: Principles of Transaction-Oriented Database Recovery, in: ACM Computing Surveys, Vol. 15, No. 4, 1983, pp. 287-317.

[8] Härder, T.: Non-Standard DBMS for Support of Emerging Applications - Requirement Analysis and Architectural Concepts, in: Proc. 22nd Annual Hawaii Int. Conf. on System Sciences, Hawaii, Jan. 1989, pp. 549-558.

[9] Schek, H,-J., Scholl, M.H.: The Relational Model with Relation-Valued Attributes, in: Information Systems, Vol. 11, No. 2, 1986, pp. 137-147.

[10] Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13th VLDB Conf., Brighton, UK, 1987, pp. 433-442.

[11] Stonebraker, M., Rowe, L.A.: The Design of POSTGRES, in: Proc. ACM SIGMOD Conf., Washington, D.C., 1986, pp. 340-355.

[12] Gray, J.N.: An Approach to Decentralized Data Management Systems, in: IEEE Transactions on Software Engineering, Vol. 12, No. 1986, pp. 684-692.

[13] Härder, T., Hübel, Ch., Meyer-Wegener, K., Mitschang, B.: Processing and Transaction Concepts for Cooperation of Engineering Workstations and a Database Server, in: Data and Knowledge Engineering 3, North-Holland, 1988, pp. 87-107.

[14] Schek, H.-J., Weikum, G.: DASDBS: Konzepte und Architektur eines neuartigen Datenbanksystems, Informatik Forsch. Entw. 2, 1987, S. 105-121.

[15] Dadam, P. et al.: A DBMS prototype to support extended NF^2-relations: an integrated view on flat tables and hierarchies, in: C. Zaniolo (ed.), Proc. ACM SIGMOD Conf., Washington, D.C., 1986, pp. 356-367.

[16] Härder, T., Meyer-Wegener, K.: Transaktionssysteme in Workstation/Server-Umgebungen, Interner Bericht, Universität Kaiserslautern, eingereicht zu Veröffentlichung.

[17] Wedekind, H., Zörntlein, G.: Eine konzeptionelle Basis für den Einsatz von Datenbanken in Flexiblen Fertigungssystemen, in: Informatik Forsch. Entw. 2, 1987, S. 83-96.