"Neue Generation von Datenbanksystemen"

O.Univ.-Prof. Dr. Johann Eder
Institut für Informatik
Universität Klagenfurt
Universitätsstr. 65-67
A-9020 Klagenfurt

Abstrakt

In der wissenschaftlichen Forschung wurde eine ganze Reihe von Konzepten für die Gestaltung der nächsten Generation von Datenbanksystemen erarbeitet: erweiterte relationale Systeme, aktive Datenbanken, deduktive Datenbanken, objektorientierte Datenbanken, erweiterbare Datenbanken, heterogene und föderierte Datenbanken. In diesem Vortrag werden diese Konzepte vorgestellt und es wird analysiert, welchen Einfluß sie auf zukünftige Datenbanksysteme haben werden.

Kurzbeschreibung des Autors

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 Kommunikationssysteme, sowie in der Planung, Modellierung und Entwicklung von Informationssystemen.

1. Einleitung

Zur Abgrenzung von anderen Softwaresystemen werden Datenbanken durch folgende Charakteristika beschrieben:

- Die Daten werden persistent gehalten.
- Die Verwaltung der Daten erfolgt typischerweise auf Sekundärspeichern.
- Die Konsistenz der Daten wird durch ein Transaktionskonzept sowie durch die Steuerung der Nebenläufigkeit auch bei gleichzeitiger Verwendung/Veränderung durch mehrere Benutzer gewährleistet.
- Datenbanken bieten Schutz vor unbefugter Verwendung/Veränderung der Daten und ermöglichen die Wiederherstellung der Daten nach Hardware- und Softwarefehlern.
- Mit einer Ad-hoc Abfragesprache können in deklarativer Weise ungeplante assoziative Fragen formuliert werden.

Wie so vieles in der Informatik erfolgt auch die Entwicklung im Datenbankbereich in Generationsfolgen. Die ursprünglich für persistente Datenhaltung eingesetzten Dateisysteme wurden von der ersten Datenbankgeneration, den graphenorientierten Datenbanken, abgelöst (Hierarchische DB und Netzwerk-DB). Die zweite Generation von Datenbanken bilden die relationalen Datenbanken, die seit nunmehr ca. 15 Jahren verfügbar sind. Derzeit werden für die meisten neuen Applikationsentwicklungen relationale Datenbanksysteme eingesetzt. Allerdings sind natürlich noch viele bestehende Applikationen mit graphenorientierten Datenbanken oder mit Filesystemen organisiert.

Neue Anwendungsbereiche und weitere Anforderungen haben Grenzen der relationalen Datenbanken aufgezeigt. Auf diese Anforderungen haben Forschung und Entwicklung mit einer Fülle von Konzepten reagiert, die vielfach in Form von Prototypen ersten Evaluierungen unterzogen wurden und nunmehr bereits in am Markt angebotenen Produkten realisiert sind oder als neue Features in den nächsten Versionen bestehender Produkte angekündigt wurden. Die Fülle der neuen Konzepte und ihre zum Teil großen strukturellen Unterschiede zu den derzeitigen Datenbanksystemen werden neue Produkte hervorbringen bzw. zu enormen

Weiterentwicklungen bestehender Produkte führen, sodaß diese Änderung allgemein als Generationswechsel aufgefaßt wird. Da die Entscheidung für ein Datenbankmanagementsystem von strategischer Bedeutung für Entwicklung und Pflege der Informationsinfrastruktur von Unternehmen und Organisationen ist, ist es wichtig künftige Entwicklungen einschätzen zu können.

Dieser Beitrag setzt sich das Ziel, diese neuen Entwicklungen auf dem Gebiet der Datenbanken zu skizzieren, ihre Ziele, Grundlagen und Methoden kurz zu beschreiben und Hinweise darauf zu geben, welche Arten von Datenbanksystemen in naher Zukunft angeboten werden.

Neue Produkte haben meist zwei wichtige Voraussetzungen: neue Applikationen und Anforderungen, die mit bestehender Technik nur unzureichend unterstützt werden (der sog. Anwendungssog) sowie neue Techniken, Methoden und Verfahren, die von Forschung und Entwicklung hervorgebracht wurden. Der Aufbau dieses Beitrags folgt diesen Voraussetzungen. Im zweiten Abschnitt werden die hervorstechendsten neuen Anforderungen und Anwendungsbereiche beschrieben und diskutiert, warum die derzeitige Technik dafür nicht immer adäquat ist. Im dritten Abschnitt werden neue Paradigmen für Datenbanken (objektorientierte, deduktive, aktive, heterogene, erweiterbare Datenbanken) beschrieben und im vierten Abschnitt wird schließlich ausgeführt, welche neuen Arten von Datenbankmanagementsystemen auf den Markt kommen werden.

2. Der Anwendungssog

Was sind nun die neuen Anwendungsbereiche und wie unterscheiden sie sich von den traditionellen? Beispiele für solche Anwendungsbereiche sind Computer Aided Design / Computer Aided Manufacturing, geographische Informationssysteme, wissenschaftliche Datenbanken (z.B. Human Genome Project, Satellitendaten, chemische Strukturen),

Büroinformationssysteme, etc.

Die wichtigsten neuen Anforderungen an Datenbankmanagementsysteme, die durch diese Anwendungsbereiche impliziert werden, sind im folgenden aufgelistet:

- Komplexe Strukturen und rekursiv definierte Objekte:

  Die Objekte in den neuen Anwendungsbereichen erfordern komplexe Datenstrukturen, um sie adäquat repräsentieren zu können. Zum Beispiel würde die Abbildung von Konstruktionszeichnungen auf flache Tabellen hohen Aufwand erfordern.

- Lange, geschachtelte, kooperierende Transaktionen:

  Derzeitige Datenbanken wurden für hohe Transaktionsraten ausgelegt, wobei die einzelnen Transaktionen vergleichsweise einfach und kurz sind. In Designumgebungen sind allerdings lange Transaktionen vorherrschend. Andere Anwendungen (insbesondere auch in vernetzten und/oder heterogenen Umgebungen) erfordern auch geschachtelte Transaktionen und kooperierende Transaktionen. Das hat natürlich auch Auswirkungen auf das Wiederherstellungssystem im Fehlerfall und auf Protokolle für die Steuerung der Nebenläufigkeit. Zum Beispiel ist ein Rücksetzen aller offenen Transaktionen bei langen Transaktionen meist nicht akzeptabel.

- Multimedia:

  Multimediale Daten wie Bild, Video und Ton erfordern neue Techniken der Speicherung und der Wiedergabe von Daten.

- Sehr große Objekte und riesiges Datenvolumen:

  Sehr viele der neuen Anwendungsbereiche erfordern die Verwaltung riesiger Datenvolumina (im Terabyte-Bereich). Darüber hinaus können einzelne Objekte (z.B. eine Satellitenaufnahme, ein Video) sehr groß sein, was durch eine für solche Daten

adäquate Systemstruktur unterstützt werden muß. Manche der Anwendungen erfordern auch eine Unterstützung der Verwaltung von Tertiärspeichern.

- Kooperation, heterogene Integration bzw. Föderation:

  Durch zunehmende Verdichtung des Informationsflusses (Stichworte: CIM, lean production, lean management), durch stetige Weiterentwicklung von Informationssystemen und der Notwendigkeit der Integration überkommener Systeme (Legacy Systems) wird ein Datenbanksystem nicht mehr zentral für alle sie benutzenden Applikationen sein. Vielmehr werden in einer Informationssystemarchitektur mehrere Datenbanken aufscheinen, die räumlich disloziert sind, unterschiedlichen Organisationseinheiten bzw. Organisationen zugeordnet sind, semantisch und in bezug auf das Datenmodell und das Datenbankmanagementsystem heterogen sind und trotzdem kooperieren müssen und von Applikationen gemeinsam verwendet werden. Neue Datenbanksysteme müssen solche Architekturen unterstützen.

- Aktive Komponenten:

  Für einige Anwendungsbereiche sind aktive Komponenten in Datenbanksystemen erforderlich, die es erleichtern, komplexe Datenstrukturen zu manipulieren, die Integrität der Daten zu gewährleisten bzw. Änderungen zu propagieren.

Darüberhinaus sollen neue Datenbanksysteme moderne Methoden des Systementwurfs unterstützen und die Produktivität der Programmierers erhöhen helfen. In diesem Zusammenhang sei auch auf einen wichtigen Kritikpunkt an derzeitigen Datenbanksystemen hingewiesen, dem sogenannten 'impedance mismatch'. Darunter versteht man die Probleme bei der Kopplung von prozeduralen Sprachen, in denen die Applikationen geschrieben werden, mit relationalen Datenbanken:

- Die Applikationsentwicklung erfolgt in einer prozeduralen Sprache, der Zugriff auf die Daten in einer deklarativen Sprache (typischerweise SQL) und die Programmierer müssen beide Sprachen beherrschen und entscheiden, was in welcher Sprache

programmiert wird.

- Die beiden Sprachen haben unterschiedliche Typsysteme.

- Die prozeduralen Sprachen bearbeiten Tupel für Tupel, während die Datenbanksprachen mengenorientiert sind, was komplexe Zwischenstrukturen (Cursor) erfordert.

- Die Optimierung erfolgt an zwei unterschiedlichen Stellen, die nichts voneinander wissen.

All diesen Anforderungen kann nicht mehr durch einfache Adaptierungen der bestehenden Technik begegnet werden. Vielmehr sind grundlegend neue Konzepte auf allen Ebenen eines Datenbanksystems notwendig.

3. Der Technologieschub

3.1 Objektorientierung

Die Entwicklung objektorientierter Datenbanken hat ihre Grundlagen in der objektorientierten Programmierung, in der semantischen Datenmodellierung, im Software Engineering und in der Wissensrepräsentation. Es gibt zwar noch keine allgemein anerkannte Definition von objektorientierten Datenbankmanagementsystemen (OODBMS), doch werden meist die folgenden Konzepte (neben den allgemeinen Charakteristiken von Datenbanken) als grundlegend angeführt:

a) Komplexe Objekte

Ein Objekt kann wieder aus Objekten aufgebaut sein. Zum Bespiel kann ein Objekt 'Graphik' aus einer Menge von Polygonen zusammengesetzt sein, und jedes dieser

Polygone besteht selbst wieder aus einer Folge von Linien. Komplexe Objekte werden mit Typkonstruktoren erzeugt, wobei meist Tupel-, Mengen- und Listenkonstruktoren zur Verfügung stehen. Komplexe Objekte können mit einfachen Operationen als Ganzes manipuliert werden (z.B. räumliches Verschieben eines Graphikobjektes).

b) Objektidentität

Objekte haben eine Identität unabhängig vom Wert (ihrer Attribute). Ein Objekt 'Dreieck' zum Beispiel behält seine Identität, auch wenn es räumlich verschoben wird, seine Farbe, Strichstärke, etc. geändert wird. Der 'object identifier' ist eine Art systemverwalteter Schlüssel, der ein Objekt über seine gesamte Lebenszeit eindeutig identifiziert.

c) Abstrakte Datentypen und Kapselung

Das für Programmiersprachen entwickelte Prinzip der Kapselung verlangt, daß auf die eigentlichen Daten eines Objektes nicht direkt, sondern nur durch Aufruf der dem Typ des Objektes zugeordneten Methoden zugegriffen werden kann. Die statische und dynamische Beschreibung von Objekten bildet somit abstrakte Datentypen.

d) Typen und Klassen

Objekte mit denselben Eigenschaften (gleiche Struktur und gleiches Verhalten) werden zu Typen oder Klassen zusammengefaßt. Eine Klasse besitzt eine Beschreibung (sowohl Spezifikation als auch Implementierung) von Objekten, eine Objektfabrik, um neue Objekte zu erzeugen und einen Objektcontainer, um die Objekte zu verwalten.

e) Vererbung

Das Konzept der Vererbung baut auf dem Prinzip der Generalisationshierarchien auf. Wenn eine Klasse A Unterklasse einer Klasse B ist, so wird die Beschreibung von B auf A vererbt, d.h. daß alle in B definierten Attribute und Methoden automatisch auch in A

definiert sind.

f) Überschreiben, Überladen, spätes Binden

Eine Unterklasse kann ererbte Attribute und Methoden auch ändern. Diese Redefinition wird Überschreiben (overriding) genannt. Es ist auch möglich, daß in verschiedenen Klassen Attribute oder Methoden mit demselben Namen definiert sind. Das wird als Überladen (overloading) des Namens (der Methode) bezeichnet.
Durch Überladen und Überschreiben ist es oft nicht möglich zur Übersetzungszeit festzustellen, welche Implementierung einer Methode an welcher Aufrufstelle eingebunden werden muß. Erst beim aktuellen Aufruf einer Methode zur Laufzeit ist bekannt, welches Objekt betroffen ist und daher wird erst zur Laufzeit die Methode gebunden. Diese Technik wird spätes Binden genannt.

g) Turing vollständige Berechenbarkeit

Die Programmiersprache eines OODBMS (zur Implementierung der Methoden) soll ausdrucksstark genug sein, um jeden beliebigen Algorithmus in dieser Sprache formulieren zu können.

h) Erweiterbarkeit

Ein OODBMS muß dem Programmierer die Möglichkeit bieten, neue Typen bzw. Klassen zu definieren und diese müssen in der gleichen Weise verwendbar sein, wie die im System vordefinierten Typen bzw. Klassen.
Objektorientierte Datenbanken wurden zuerst als eine Art persistente Programmiersprache entwickelt. Nach und nach kamen wichtige Datenbankfeatures, wie assoziative Abfragesprache, Schichtenarchitektur, etc. hinzu. Sie überwinden jedenfalls das Problem des impedance mismatch, da für das Schreiben von Applikationsprogrammen dieselbe Sprache wie für das Schreiben der Methoden herangezogen werden kann. Objektorientierte Konzepte (wenn auch vielleicht nicht alle oben angeführten, bzw. einige weitere mehr) werden jedenfalls in allen

Datenbanksystemen der nächsten Generation zu finden sein.

3.2 Deduktive Datenbanken

Die Entwicklung deduktiver Datenbanken wurde von der logischen Programmierung und von Expertensystemen beeinflußt. Während relationale Datenbanken große Datenmengen sicher, effizient und zuverlässig verwalten können, ist es aufgrund der beschränkten Ausdruckskraft relationaler Sprachen nicht möglich, sämtliche Information, die in den Daten repräsentiert wird, mittels relationaler Abfragen auch tatsächlich zu gewinnen. Als wesentliches Hemmnis hat sich dabei herausgestellt, daß rekursive Abfragen in relationalen Sprachen nicht formuliert werden können. Aus der Sicht der Wissensverarbeitung ist es in relationalen Datenbanken lediglich möglich, extensionales Wissen zu repräsentieren (eben Daten in Relationen), aber nur sehr eingeschränkt möglich, intensionales Wissen, wie z.B. Regelwissen, auszudrücken. Diesem Problem versuchen deduktive Datenbanken abzuhelfen. Dabei werden im wesentlichen drei Ansätze verfolgt:

a) Kopplung

   Eine Datenbank wird mit einem für die Wissensverarbeitung geeigneten System gekoppelt, z.B. mit Prolog oder einer Expertensystem-Shell. Die Datenbank verwaltet dabei die Fakten, während die Formulierung und Verarbeitung der Regeln im anderen System erfolgt. Dem Kopplungsansatz inhärent ist allerdings das Problem des bereits erwähnten impedance mismatch sowie das Problem der Optimierung des Gesamtsystems.

b) Erweiterung von Abfragesprachen

   Dabei werden Abfragesprachen (vor allem SQL) und insbesondere der Mechanismus der View-Definition erweitert, um rekursive Abfragen und Regeln formulieren zu können. Diese neuen Sprachen werden dann durch Erweiterung der Interpreter bzw. Compiler sowie der Abfrageoptimierer und gelegentlich durch spezielle Speicherstrukturen unterstützt.

c) Deduktive Datenbanken

Ausgangspunkt für den dritten Ansatz bilden ausdrucksstarke, auf logischen Grundlagen beruhende, deklarative Datenbanksprachen, etwa Sprachen, die auf Datalog, einer Teilmenge der Prädikatenlogik erster Stufe, basieren. In dieser Sprache ist es möglich sowohl Daten extensional zu repräsentieren und Regeln zu definieren als auch komplexe Ad-hoc-Abfragen durchzuführen. Die Sprache ist deklarativ, d.h. der tatsächliche Algorithmus zur Auswertung einer Abfrage wird vom System (Optimierer) ausgewählt.

Als Beispiel für eine solche Datenbank wird im folgenden die Auflösung einer Stückliste angegeben. Dafür wird eine Prolog-ähnliche Syntax verwendet, allerdings nicht die Evaluierungsstrategie von Prolog.

```
Teil(A, B) :- stückliste(A, B).
Teil(A, B) :- Teil(A, C), stückliste(C, D).
?- Teil(motor, X).
```

Die beiden ersten Zeilen definieren dabei Teil(A, B) als transitive Hülle der Stückliste. In der dritten Zeile wird eine Abfrage angegeben, die alle Bestandteile eines Motors ermittelt. Deduktive Datenbanken ermöglichen eine vergleichsweise einfache, flexible und vor allem deklarative Programmierung von Teilen eines Anwendungssystems. Die Integration von deduktiven Datenbanken und Objektorientierten Datenbanken hat großes Forschungsinteresse geweckt und ist noch nicht endgültig gelöst.

3.3 Aktive Datenbanken

Konventionelle Datenbanken sind passiv, d.h. 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.

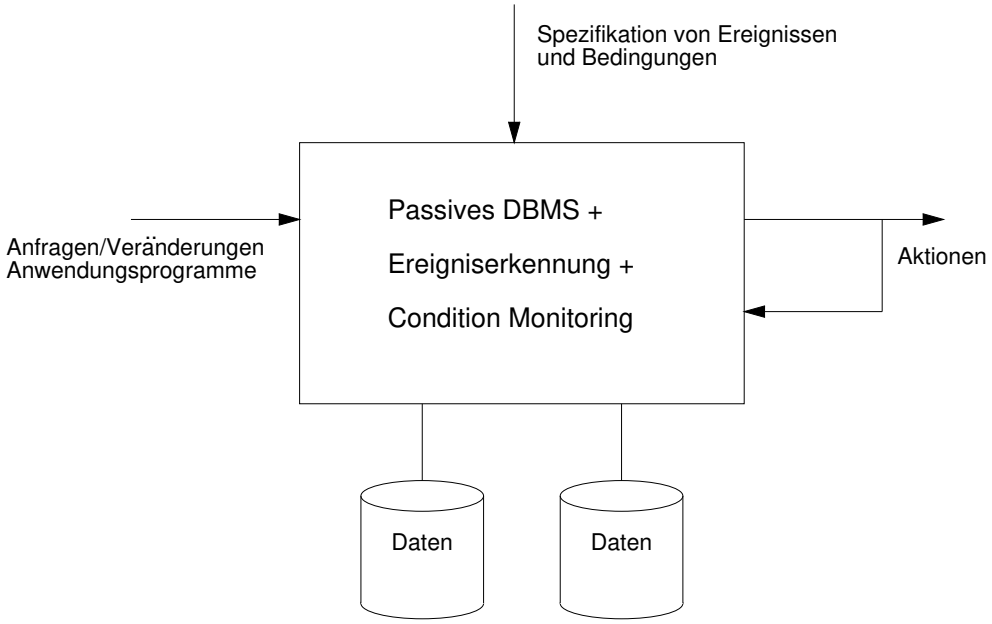

Abb. 1 Prinzip einer aktiven Datenbank

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. Wird eine Regel durch ein Ereignis getriggert, werden die Bedingungen dieser Regel überprüft. Sind sie erfüllt, werden die Aktionen der Regel ausgeführt.

Bedingungen sind dabei Beschreibungen von Zuständen der Datenbank. Aktionen können wiederum Veränderungen in der Datenbank sein, aber auch Aufrufe von externen Prozeduren.

Beispiel:

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 quantitiy 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.

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. Monitoring: Überwachen der Veränderungen zum Aufspüren atypischer Zustände.

3.4 Erweiterbare Datenbanken

Derzeitige Datenbanksysteme sind eher monolithisch und bieten dem Anwender nur geringe Möglichkeiten, die Systeme zu erweitern oder zu adaptieren. Erweiterbare Datenbanken versuchen dem abzuhelfen. Erweiterbarkeit wird dabei auf mehreren Ebenen angeboten:

- Definition neuer Datentypen und Funktionen, die auf diesen Datentypen arbeiten, um mächtigere Konzepte für die Modellierung von Datenbanken zur Verfügung zu haben.

- Erweiterung der Abfragesprache um neue Operatoren (wie z.B. transitive Hülle), um die Ausdrucksstärke der Abfragesprache zu erhöhen oder sie spezifischen Erfordernissen anzupassen.

- Einbindung neuer Algorithmen zur Implementierung bestehender und neuer Operatoren (z.B. neue effizientere Algorithmen für Joins).

- Erweiterung des Abfrageoptimierers um neue Optimierungsverfahren.

- Integration neuer Zugriffspfade, Indexstrukturen, etc.

- Definition und Einbindung neuer Speicherstrukturen, um neue Speichermedien für die Datenbank verwenden zu können.

- Erweiterung/Austausch von Transaktionsmanagern, Definition neuer Sperren und Sperrprotokolle.

Dabei werden von den derzeitigen Prototypen zwei unterschiedliche Wege beschritten. Der eine Weg bietet dem Anwender die größtmögliche Flexibilität, bei allen (oder einigen) der oben beschriebenen Erweiterungsmöglichkeiten. Häufig beschränkt sich die Erweiterungsmöglichkeit allerdings auf die Definition neuer Datentypen und die Definition entsprechender Information für den (generischen) Abfrageoptimierer. Der zweite Weg bietet

einen Baukastenansatz, d.h es wird ein generisches Datenbanksystem angeboten, bei dem der Anwender bei der Installation Systemkomponenten (z.B. Transaktionsmanager, Datentypen, Speicherstrukturen, etc.) auswählt und dann ein für seine Anforderungen spezifisches DBMS generiert.

3.5 Heterogene verteilte Datenbanken

In komplexen Informationssystemen werden die Daten nicht in einer einzigen zentralen Datenbank gehalten werden, sondern in mehreren verteilten und wahrscheinlich heterogenen Datenbanken. Als Beispiel sei die informationstechnische Vernetzung eines Produktionsbetriebes mit seinen Zulieferern angeführt.

Solche dezentralen Informationssysteme können in 3 Dimensionen beschrieben werden:

a) Verteilung: zentral - dezentral
b) Autonomie: integriert - autonom
c) Heterogenität: homogen - heterogen

Jede dieser Dimensionen hat nun wieder mehrere Aspekte:

a) Verteilung

   Art der Dislozierung (LAN, WAN, etc.)

b) Autonomie

   Beschreibt, inwieweit eine einzelne Datenbank (bzw. deren Betreiber) (im folgenden Knoten genannt) selbständige Entscheidungen treffen kann:

   - Kommunikationsautonomie:
     Knoten entscheidet, ob und wann mit anderen Knoten kommuniziert wird.

- Verarbeitungsautonomie:
  Knoten kann lokale Prozesse durchführen, ohne auf externe Prozesse achten zu müssen.

- Föderationsautonomie:
  Knoten entscheidet, welche Ressourcen er anderen Knoten zur Verfügung stellt.

- Realisierungsautonomie:
  Knoten entscheidet, wie (mit welchem System, Modell, etc.) er technisch realisiert wird.

c) Heterogenität

Verschiedenartigkeit der Realisierung der einzelnen Knoten (z.B. unterschiedliche DBMS/Hersteller, Datenmodelle, Abfragesprachen, Schemata, Genauigkeit, Einheiten, Granularitäten, Kommunikationsprotokolle, Transaktionsprotokolle, etc.).

Um trotz all dieser Heterogenität eine Kooperation (bzw. Föderation) von solchen Datenbanken zu ermöglichen, sind eine Reihe von technischen Voraussetzungen notwendig, wie etwa Zweiphasen-Commit-Protokolle, Schemaintegrationsmethoden, Remote Data Access, etc.

Ein Ansatz für die Realisierung solcher heterogener Systeme sind föderierte Datenbanken. Die Teilnehmer an einer Föderation geben bestimmte Kompetenzen an die Föderation als ganzes ab und stellen den anderen Teilnehmern bestimmte Ressourcen zur Verfügung. Ein Knoten kann dabei an mehreren Föderationen teilnehmen. Es werden nicht die ganzen Datenbanken integriert, sondern nur jene Teile, die in die Föderation eingebracht wurden.

4. Neue Systeme

Durch welche Produkte wird die neue Generation von Datenbanksystemen am Markt repräsentiert werden? Hier scheinen drei unterschiedliche Klassen angeboten zu werden: objektorientierte DBMS, erweiterte relationale DBMS und unifizierte DBMS.

a) Objektorientierte Datenbanken

Hier sind in den letzten Jahren einige Produkte am Markt erschienen. Die Palette reicht dabei von persistenten Programmiersprachen, denen einige bewährte Datenbankfeatures wie 3-Schichten-Architektur oder Ad-hoc-Abfragesprache fehlen, bis zu vollständigen Datenbanksystemen, die auch Schnittstellen zu mehreren Programmiersprachen anbieten. Sehr einflußreich auf diese Systeme werden die Standardisierungsbemühungen der ODMG (Object Database Management Group) sein, die Standards für eine Datendefinitionssprache, eine assoziative Abfragesprache sowie für das Binding für C++ und Smalltalk erarbeitet. Produkte, die diesen Standards genügen, bieten dem Anwender eine einheitliche Schnittstelle und werden sich vor allem in den Eigenschaften der Datenbankmaschine und den Tools unterscheiden. In einigen Prototypen werden auch bereits aktive und deduktive Konzepte in Objektorientierte Datenbanken integriert.

b) Erweiterte relationale Systeme

Alle großen Hersteller relationaler Systeme haben für die nächsten Versionen die Aufnahme objektorientierter, aktiver und deduktiver Konzepte angekündigt. Diese neuen Features ersetzen allerdings nicht alte Features, sondern stellen echte Erweiterungen dar. Insbesondere wollen diese Systeme auch ermöglichen, daß alle alten Applikationen unverändert auch auf den neuen Systemversionen lauffähig sind. Maßgeblich beeinflußt wird diese Entwicklung auch durch die Entwicklung eines neuen SQL Standards (SQL-3), dessen Entwurf bereits sehr umfangreich ist und dessen Fertigstellung für 1996 erwartet wird. Die wichtigsten neuen Features von SQL-3 sind abstrakte Datentypen, Vererbung, Object-Identifier, gespeicherte Prozeduren und Funktionen, rekursive

Abfragen und Trigger.

c) Unifizierte Datenbanken

Diese Systeme versuchen eine Integration von relationalen Systemen mit den in Abschnitt 3 aufgeführten Konzepten. Es wird abzuwarten sein, wieweit sie sich von den erweiterten relationalen Systemen unterscheiden werden. Jedenfalls werden in nächster Zeit Produkte unter diesem Etikett am Markt erscheinen.

5. Zusammmenfassung

Zusammenfassend kann festgehalten werden, daß durch neue Anwendungsbereiche und neue Anforderungen eine neue Generation von Datenbanksystemen zum Einsatz kommen wird. Welche Systeme langfristig in welchen Anwendungsbereichen erfolgreich sein werden, ob sich der eher revolutionäre Ansatz der objektorientierten Datenbanken oder der eher evolutionäre Ansatz der erweiterten relationalen und der unifizierten Datenbanken durchsetzen wird, kann derzeit noch nicht beantwortet werden. Wie wir aus der Vergangenheit wissen, hängt das nicht nur von technischen Eigenschaften der Produkte ab.