Datenmanagement

Datenbanken und betriebliche Datenmodellierung

Von
Prof. Dr. Joachim Fischer

unter Mitarbeit von
Dipl.-Inf. Holger Dresing

R. Oldenbourg Verlag München Wien

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Fischer, Joachim:
Datenmanagement : Datenbanken und betriebliche Datenmodellierung / von Joachim Fischer. Unter Mitarb. von Holger Dresing. – München ; Wien : Oldenbourg, 1992
ISBN 3-486-22357-7

Gesamtherstellung: R. Oldenbourg Graphische Betriebe GmbH, München

ISBN 3-486-22357-7

Vorwort

Die Wirtschaftsinformatiker befassen sich seit einigen Jahren verstärkt mit „Daten" oder besser „Informationen" als zentralem Produktionsfaktor. Damit wird anerkannt, daß der Nutzen der betrieblichen Datenverarbeitung für das Unternehmensmanagement aus der Kombination von technologischen Möglichkeiten mit zielgerichtet strukturierten und verfügbaren Daten folgt. Daten bilden den Rohstoff für alle Datenverarbeitungsprozesse. Ihre Qualität bestimmt die Ergebnisse der DV-Produktion.

Ein zielgerichtetes Datenmanagement betrachtet die wirtschaftlichen Potentiale der Informationsverarbeitung. Ziel ist es, der Skepsis der Unternehmensleitungen entgegenzuwirken, für die die Datenverarbeitung heute „ein notwendiges Übel"ist, das „unkontrolliert Ressourcen verschlingt", sich anders als Vertrieb, Produktion oder Logistik der wirtschaftlichen Steuerung entzieht und nicht zu den wirtschaftlichen Zielen der Unternehmung beiträgt. (Umfrage der "COMPUTERWORLD" im Herbst 1989 bei europäischen und US-amerikanischen Unternehmen). Aus der „respektvollen Anerkennung" der DV-Technologie und Organisation wird im Management zunehmend eine „kritische Distanz", die nach dem Beitrag der DV zu den Erfolgsfaktoren der Unternehmung fragt. Verstärkt wird diese Skepsis im Management durch steigende DV-Kosten trotz sinkender Hardware-Preise, durch das Beharren der DV-Organisation auf Zentralisierung ungeachtet der billigen und komfortablen dezentralen DV am Arbeitsplatz, durch Unsicherheiten in der DV-Gemeinde über den zukünftig richtigen Weg und über immer neue "Heilslehren", die eben diesen versprechen und preisen.

Gefragt ist ein ökonomisch ausgerichtetes Informationsmanagement statt der technisch dominierten DV-Organisation, das den wirtschaftlich sinnvollen Informations-Output stärker gewichtet als die technischen Aspekte der Datenverarbeitung. Ein Schlüsselfaktor sind dafür die verfügbaren und gewinnbaren Datenbestände. Deren Struktur soll sich aus den Unternehmenszielen ergeben, deren Aktualität und Qualität aus den Erfordernissen des Unternehmensmanagements. Und schließlich soll sich aus dem zu bearbeitenden Rohstoff "Daten" den Produktionsapparat mit seiner Hard- und Software ergeben.

Mit dem Rohstoff "Daten" beschäftigt sich das vorliegende Manuskript aus der Sicht der Wirtschaftsinformatik. Es will damit nicht konkurrieren mit der Vielzahl ausgezeichneter Werke über Datenbanken und Datenorganisation, die meist aus der Informatik stammen. Und es befaßt sich auch nicht mit speziellen betriebswirtschaftliche Informationsversorgungssystemen, die beispielsweise Gegenstand des Rechnungswesens oder des Controlling sind. Stattdessen werden Datenmodelle und Datenbanken als zentrale Bestandteile integrierter betrieblicher und überbetrieblicher Anwendungssysteme betrachtet. Technische Einzelheiten werden dabei zugunsten wirtschaftlicher Anwendungsaspekte weniger stark gewichtet.

Viele haben dazu beigetragen, daß dieses Buch entstehen konnte: Geholfen haben mir die kritischen Fragen meiner Studenten und die Diskussionen mit meinen Mitarbeiter an der Universität-GH-Paderborn. Herr Diplom-Kaufmann Uwe Kern hat mit mir eingehend das Entity-Relationship-Modell diskutiert und Herr Diplom-Informatiker Holger Dresing hat wesentliche Teile zu den Kapiteln über Datenbanksysteme und -sprachen beigetragen und die Endredaktion übernommen. Bei der technischen Fertigstellung haben mir meine Sekretärinnen Frau Jöhren (bis Juli 1991) und seitdem Frau Petermeier sowie meine studentischen Hilfskräfte (insbesondere Frau Antje Koch) sehr geholfen. Doch wo wären wir alle heute ohne unsere MAC(intosh-Rechner)? Ihnen allen sei ganz herzlich gedankt.

Joachim Fischer

Paderborn, im Herbst 1992

Inhaltsverzeichnis

Kapitel 1: Betriebliche Datenmodelle

1. Funktions- versus datengetriebene DV-Systementwicklung

Daten bilden den Rohstoff der elektronischen Datenverarbeitung. Entsprechend dem Wortpaar DATEN-VERARBEITUNG kann die Entwicklung von DV-Systemen entweder von den Ein- und Ausgabedaten oder von den benötigten Verarbeitungsfunktionen geleitet werden:

(1) Die **funktionsgetriebene Vorgehensweise** legt zunächst die Verarbeitungsprozesse fest, bestimmt anschließend die benötigten Daten und strukturiert dann diese Daten im Sinne optimaler Zugriffe der Prozesse. Hierzu analysieren DV-Spezialisten die bisherigen oder denkbaren Prozesse und leiten daraus die benötigten Dateninhalte und Datenstrukturen ab.

(2) Die **datenorientierte Vorgehensweise** definiert zunächst die Datenstruktur in Form eines Datenmodells mit dem Ziel einer semantischen Beschreibung der Daten vor dem Hintergrund ihrer pragmatischen Verwendung im Unternehmen. Anliegen ist hier eine stärkere Beteiligung der Benutzer am Systementwurf, da sich diese der inhaltlichen Bedeutung der Daten und ihrer logischen Abhängigkeiten besser bewußt sind.
Erst nach Klärung der semantischen Struktur der Daten und ihrer inhaltlichen Nutzung erfolgt ein zugriffsorientiertes Datenbankdesign und ein Funktionsmodell.

(3) **Integrierende Ansätze** versuchen die Vorteile beider Ansätze zu kombinieren, indem sie sowohl die einheitliche Datenstruktur in unterschiedlichen Funktionen/Prozessen als auch die einheitliche Funktionsstruktur bei unterschiedlichen Datenstrukturen betrachten. Sie verlangen daher sowohl die breite, auf Integration ausgerichtete Beschreibung (semantischer) Datenmodelle als auch einheitlich strukturierte Funktionsmodelle.

Somit scheint der integrierte Ansatz die Vorteile der beiden Vorgehensweisen zu verbinden. Allerdings verkennt ein solcher Schluß, daß die datengetriebene Systementwicklung grundsätzlich anders vorgeht als die funktionsgetriebene.

In einem zeitaufwendigen Prozeß werden zunächst einmal die Informationsstrukturen der Unternehmung ermittelt und in einer Art „Dateninfrastruktur" definiert.

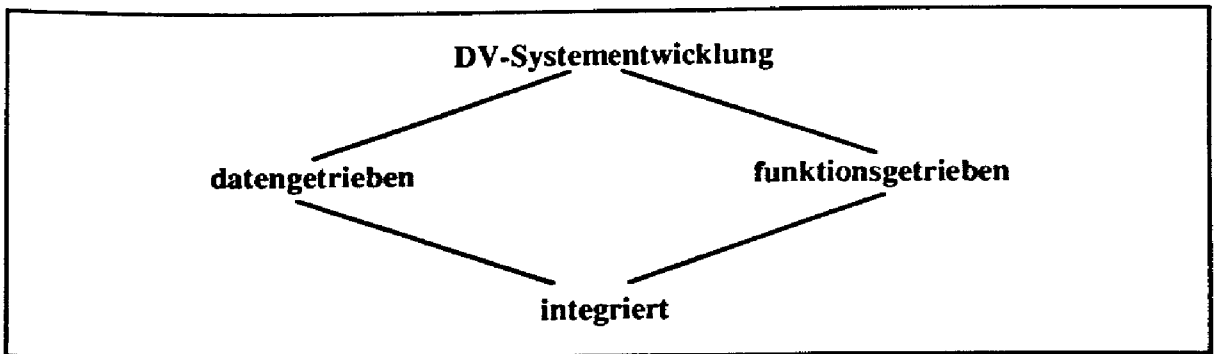


Abb. 1/1: Wege der DV-Systementwicklung

Diese „Dateninfrastruktur" wird zunächst nur in einem abstrakten Modell, dem sogenannten Datenmodell, beschrieben und nur insoweit konkretisiert, wie das für die aktuell anstehenden Anwendungssystementwicklungen notwendig ist.

Im Gegensatz zu dieser infrastrukturellen Sicht steht bei der funktionsgetriebenen Vorgehensweise die Lösung des jeweiligen konkreten Anwendungsproblems im Vordergrund.

Bei der datengetriebenen Vorgehensweise wird das Problem immer mit dem Ziel spezifiziert und analysiert, es später in einem allgemeineren Datenzusammenhang zu erfassen.

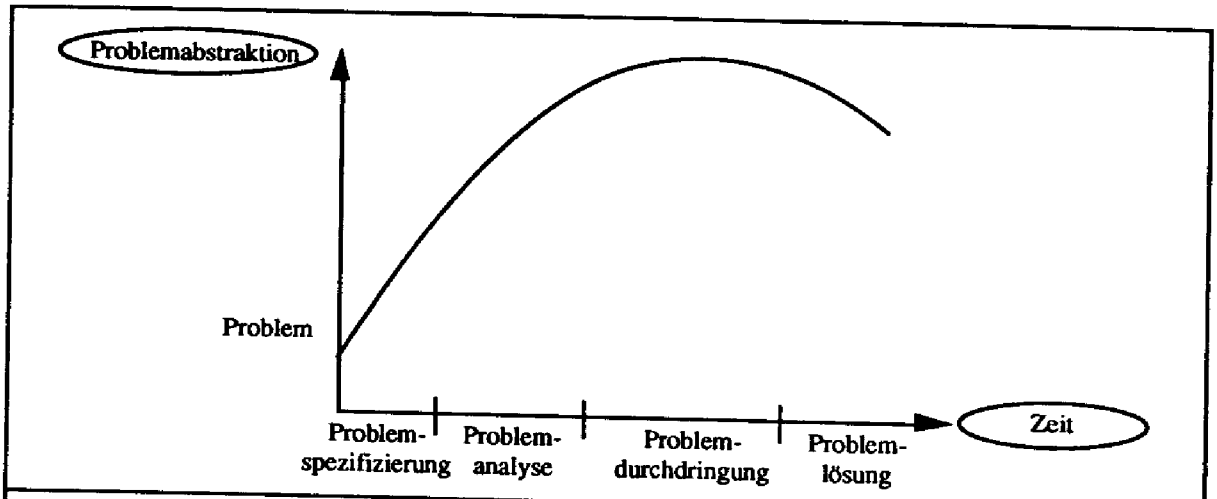


Abb. 1/2: Problembezug der datengetriebenen Vorgehensweise in Abhängigkeit von der Zeit

Erst nach der Durchdringung des aktuellen Problems kann der Systementwickler von dessen spezifischen Eigenschaften abstrahieren und nach einer dauerhaften, strukturell ausgerichteten Lösung streben.

Dieser zeitaufwendige Prozeß wird heute mit Hilfe der Erfahrungen und Methoden zur Datenmodellierung verkürzt.

Integrierte Ansätze der Systementwicklung laufen Gefahr, die auf die Schaffung einer DV-Infrastruktur ausgerichtete, konzeptionell und zeitlich aufwendige, datengetriebene Vorgehensweise" mit einer auf die konkrete Problemlösung abzielenden, eher kurzfristig orientierten „funktionsgetriebenen Vorgehensweise" zu verbinden.

Gründe für datenorientiertes Vorgehen aus DV-Sicht:

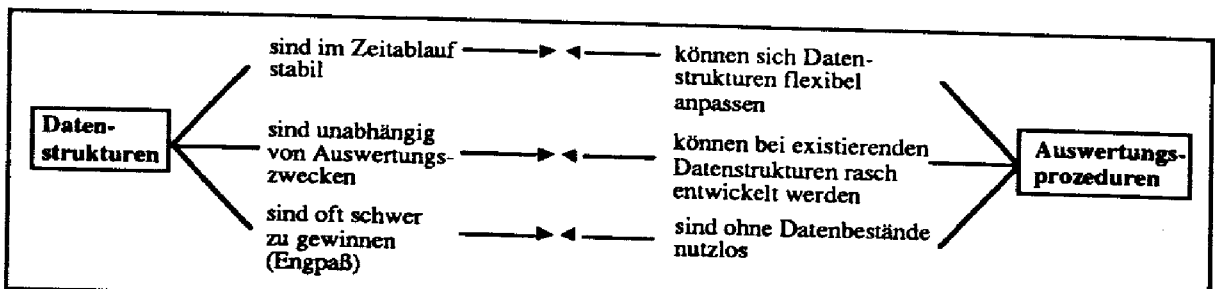


Abb. 1/3: Gründe für datengetriebene Systementwicklung

(1) Intertemporale Stabilität der Datenstruktur versus häufig wechselnde Anwendungsfunktionen

Funktionsmodelle bilden in aller Regel einen erheblichen Teil der Aufbau- und Ablauforganisation einer Unternehmung mit ab, die häufigen Veränderungen unterworfen sind. Demgegenüber stellen Datenmodelle auf die für den Geschäftsverkehr benötigten Daten ab, die in der Regel solange konstant bleiben, solange sich das Geschäftsfeld oder die Gesamtzielsetzung der Unternehmung nicht verändert.

Hinsichtlich der zeitlichen Stabilität von Datenstrukturen sind zwei Arten von Informationsstrukturen zu unterscheiden.

	Grund-Informations-Strukturen	**Zusatz-Informations-Strukturen**
Kennzeichen	Vorhersehbare Informationselemente zur Abwicklung des Geschäftsverkehrs	fallweise Informationsstrukturen zur Beantwortung spezifischer Fragestellungen
Beispiele:	- Kontostrukturen - Geschäftspartnerangaben	- Umweltschutzausgaben im Werk XY

Abb. 1/4: Stabile und fallweise Informationsstrukturen (vgl. Czap (1991))

(2) Anwendungen können sich auf wechselnde Datenstrukturen flexibel einstellen; generieren bei isolierter Entwicklung jedoch durchaus unterschiedliche Datenstrukturen

Die Entwicklung neuer Anwendungsprogramme ist relativ einfach, sobald eine einheitliche und stimmige Datenstruktur im Unternehmen existiert.

Aus dieser einheitlichen und übergreifenden Datenstruktur (Unternehmensdatenmodell) werden die für die konkrete Anwendung erforderlichen spezifischen Datenelemente als sogenannte „Datensichten" (Views) extrahiert. Erst nachdem diese Sichten auf das Unternehmensmodell definiert sind, werden die aus Anwendungssicht erforderlichen Funktionen implementiert.

Dieser Ablauf garantiert, daß die Datenbestände für alle Anwendungsprogramme in einheitlicher Struktur verfügbar sind und Schnittstellenprobleme, d. h. die Definition der zu empfangenden und an nachfolgende Programme weiterzugebenden Daten, weitgehend vermieden werden.

(3) Datenstruktur ist gestaltender Engpaßfaktor (Datenverfügbarkeit/Datengenerierbarkeit)

Insbesondere für Anwendungssysteme der höheren vertikalen Stufen (Informations- und Planungssysteme) fehlen häufig die notwendigen Datenbestände, um die zur Verfügung stehenden Methoden und Modelle sinnvoll einsetzen zu können. Besonders deutlich wird das bei der Diskussion um Management-Informationssysteme (oder die Abwandlungen Entscheidungsunterstützungssysteme oder Vorstands-Informationssysteme).

Die komfortablen Methoden oder Modelle zur Auswertung und Aufbereitung von Daten für die Unterstützung des Managements bei Entscheidungen sind nutzlos, so lange diese Daten nicht zur Verfügung stehen.

(4) Die Daten einer Unternehmung existieren unabhängig von ihrer Verwendung

Informationsstrukturen existieren auch bevor die konkrete Nutzung für spezifische Zwecke in bestimmten Programmen bekannt ist. Sie lassen sich auch vorbeugend für spätere, heute noch nicht absehbare Auswertungszwecke, festlegen.

Ansatzpunkte für eine Analyse der Informationsgrundstrukturen lassen sich beispielsweise ableiten aus Formular- oder Dokumentstrukturen, aus dem Informationsfluß mit Marktpartnern oder innerhalb der Organisation der Unternehmung.

Zum Beispiel wurde bei vielen Banken und Versicherungen die konten- oder vertragsorientierte Verarbeitung auf eine kundenorientierte Verarbeitung umgestellt, um besser das Geschäft auf die Markterfordernisse auszurichten (vgl. Tulowitzki (1991), S. 95).

Eine datenorientierte Systementwicklung hätte diese Umstellung erleichtert; durch die jahrelange funktionsorientierte DV-Ausrichtung sind jedoch programmorientierte Datenbestände entstanden, die im Zuge dieser kundenorientierten Verarbeitung zu unnützen Datenfriedhöfen werden.

Viele Unternehmen scheuen daher solche Veränderungen aus DV-Gründen, obwohl diese aus strategischer Geschäftssicht erforderlich sind. Die Datenverarbeitung behindert bei funktionsorientierter DV-Systementwicklung die strategische Ausrichtung des Unternehmens anstatt sie zu fördern.

(5) Schnelle Implementierbarkeit von Applikationen

Existieren Datenbestände einheitlicher Struktur im Unternehmen (z. B. in der Form eines integrierten Datenbanksystems), so lassen sich mit Hilfe benutzerfreundlicher Systementwicklungswerkzeuge rasch und mit starker Einbindung der Endbenutzer Anwendungssysteme entwickeln. So existieren für Datenbanksysteme heute eine Reihe komfortabler Programmiersprachen (sogenannte Sprachen der 4. Generation) und viele Endbenutzerwerkzeuge (wie Tabellenkalkulationsprogramme, PC-Datenbanksysteme) verfügen über Schnittstellen zu den verbreiteten Datenbanksystemen.

Werden solche Werkzeuge zu einer unkoordinierten Systementwicklung durch die dezentralen Organisationseinheiten genutzt, so führt dieses zu einem "DV-Wildwuchs" und zu uneinheitlichen Datenbeständen. Die weitverbreitete "zentrale DV-Planung" vermeidet zwar den "Wildwuchs", doch erfüllt sie nur in seltenen Fällen den Wunsch der Fachabteilungen nach "benutzerfreundlichen" und rasch verfügbaren DV-Lösungen.

Die Einbindung der Endnutzer in die Systementwicklung führt außerdem zu einer schnelleren Durchdringung der Problemstellung, da die Sprachbarrieren zwischen Fachleuten und DV-Spezialisten durch die gemeinsame Arbeit am Produkt (= Programm) besser überwunden werden.

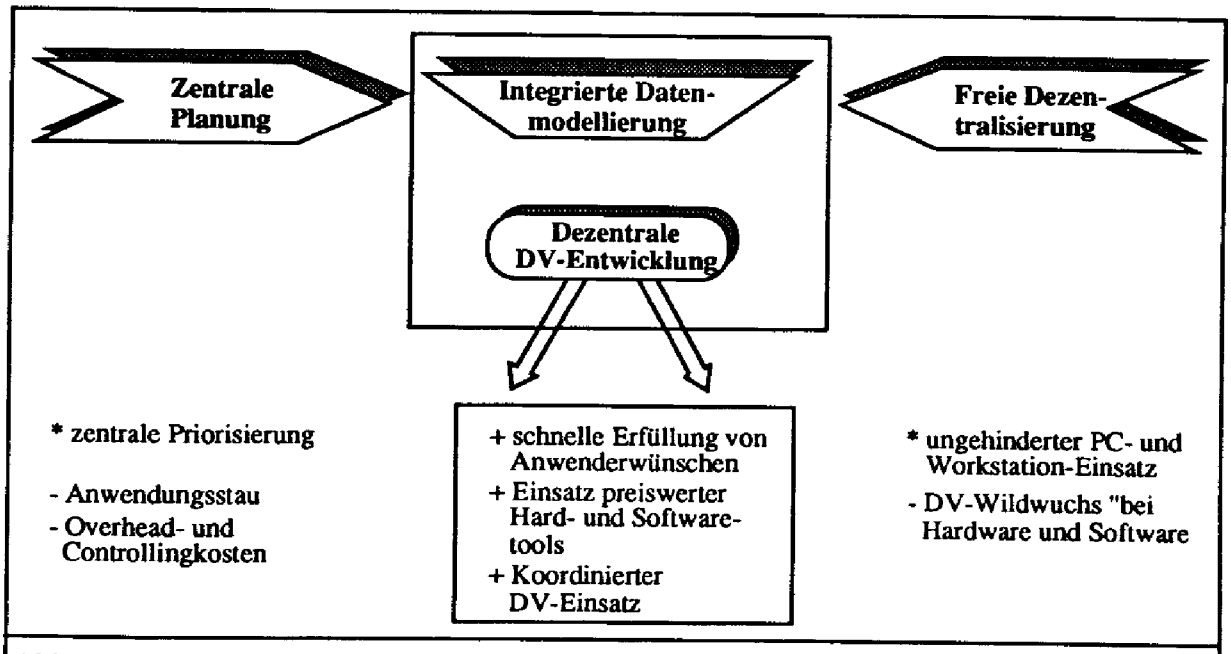


Abb. 1/5: Integrierte Datenmodellierung

Dieser Vorteil ist insbesondere dort von Belang, wo die Problemlösungsprozeduren des Fachspezialisten durch die normalen systemanalytischen Methoden nicht ermittelbar sind.

(6) Funktionsorientiertes Vorgehen führt unter Umständen zu redundanten Datenbeständen

Die funktionsgetriebene Systementwicklung legt zuerst die Verteilung der Datenverarbeitungsfunktion auf Programmsysteme fest, deren Aufbau dann die Strukturierung der Daten bestimmt.

Im Unterschied dazu strebt die datengetriebene Systementwicklung nach einer unabhängigen Struktur und Verwaltung der Datenbestände in sogenannten „Datenbanksystemen". Diese Datenunabhängigkeit bietet die Möglichkeit zur Datenintegration, d. h. die Nutzung gemeinsamer Datenbestände durch mehrere Programmsysteme. Dadurch wird die doppelte Speicherung von Datenbeständen vermieden, die nicht nur den Speicherplatzbedarf senkt, sondern auch die logische Integrität der Daten erhöht.

Gründe für datenorientiertes Vorgehen aus betriebswirtschaftlicher Sicht:

(1) Datenbestände (über Kunden, Produkte, Lieferanten) mit ihren Spezifika und Auswertungsmöglichkeiten stellen eine Quelle strategischer Vorteile dar:

Die geschäftspolitischen Möglichkeiten einer Unternehmung werden entscheidend dadurch bestimmt, welche Informationen diese über ihre Marktpartner und ihren internen Leistungserstellungsprozeß besitzen.

Bereich	Beispiel
Banken	Kundendaten geben verbesserte Möglichkeiten zur Kreditrisikoeinschätzung und Anlageberatung
Fluggesellschaften	Umfangreiche Kundendatenbank mit detaillierten Angaben gibt die Möglichkeit zur verbesserten Kundenbedienung (Service an Bord, Sitzplatzwahl, Sonderangebote, Reiseberatung)
Autohaus	Kundendaten eröffnen - Werbemöglichkeiten für Werkstattgeschäft (TÜV-Prüfung, ASU, Inspektion) - Direktansprache im Gebrauchtwagengeschäft (Kauf, Verkauf und Neuwagengeschäft)

Abb. 1/6: Nutzen von umfangreichen Kundendatenbeständen in drei Bereichen

Datenbestände über Kunden eröffnen bessere Möglichkeiten zum Einsatz der absatzpolitischen Instrumente (Werbung, Preise, Produktgestaltung, Vertrieb); Informationen über Lieferanten ermöglichen Wege zur rationalen Leistungserstellung.

(2) Traditionelle Managementinformationsinstrumente sind datenorientiert und überlassen die Interpretation dem jeweiligen Leser:

Die traditionellen Informationssysteme des Managements basieren auf den umfangreichen Datenbeständen, die durch die verschiedenen Buchhaltungssysteme im Unternehmen gewonnen werden und werten diese mit verschiedenen Analyseinstrumenten aus. Dazu gehören die Verfahren des handels- und steuerrechtlichen Jahresabschlusses (Bilanz-, Gewinn- und Verlustrechnung), die Instrumente der Kostenrechnung und der Finanzierungsrechnungen.

Die Buchhaltung ist darauf ausgerichtet, möglichst differenzierte Datenbestände für unterschiedliche Auswertungszwecke zu gewinnen und vorzuhalten. Die Prozeduren der Buchhaltung (z. B. Kontierung, Trennung in Grundbuch und Hauptbuch, Differenzierung von Erfassung und Bewertung etc.) sind für diesen Zweck konzipiert.

Das Management ist daran gewöhnt, aus der Buchhaltung spezielle Grundinformationen abfragen zu können und diese mit systematischen oder ad-hoc-Prozeduren auszuwerten.

(3) Datenbestände sind eine wesentliche Unternehmensressource, mit denen ähnlich verfahren werden muß wie mit anderen Produktionsfaktoren:

Über den wirtschaftlichen Nutzen von Informationsbeständen existieren eine Reihe historischer, z. T. anekdotischer Aussagen, jedoch immer noch zu wenig systematische Untersuchungen. Bekannte Beispiele sind das Flugreservierungssystem von American Airlines oder das Distributionssystem von Hospital Supplies.

Die bisherigen Untersuchungen, z. B. im Rahmen der empirischen Bilanztheorie und Kapitalmarkttheorie sowie in der Industrieökonomik, begründen jedoch die These, daß Informationen eine knappe Unternehmensressource sind, mit der wirtschaftlich, zielgerichtet und nach den Grundsätzen der Ordnungsmäßigkeit umzugehen ist.

Diese Grundsätze wurden rechtlich verankert, um eine nicht ordnungsgemäße Ausnutzung von Informationsvorsprüngen für Wettbewerbsvorteile bzw. Benachteiligung von

Marktpartnern zu vermeiden. Man denke etwa an die Bilanzvorschriften, Insiderregelungen für Börsengeschäfte oder die Regelungen gegen unlauteren Wettbewerb.

Allerdings hängt der wirtschaftliche Nutzen von Informationen nicht nur von deren Güte und Menge ab, sondern entscheidend davon, wie diese Daten im Leistungs- und Wertschöpfungsprozeß von Unternehmen durch die handelnden Personen genutzt werden und ob für eine zielgerichtete Nutzung ausreichend Ressourcen zur Verfügung stehen. Beispiele aus dem politischen Bereich zeigen, daß der beste Nachrichtendienst nichts nützt, wenn die ökonomischen oder militärischen Mittel zur Nutzung der Erkenntnisse fehlen.

Infolgedessen sind Fragen der Informationsgewinnung und -verarbeitung immer nachgelagerte Probleme, die von denen im eigentlichen Leistungsprozeß dominiert werden.

Die gezielte Gewinnung und Nutzung von Datenbeständen ist damit zwar eine nicht hinreichende, wohl aber notwendige Bedingung zur Realisierung der Geschäftsstrategie.

(4) Datenintegration schafft erst die Möglichkeit und die Basis für integrierte Anwendungssysteme:

Anwendungssysteme unterscheiden sich in ihren Verarbeitungsschritten, methodischen Grundlagen und in der programmtechnischen Umsetzung erheblich. Ihre gemeinsame Basis und die Verkettung ist der integrierte Datenbestand. Dies gilt sowohl für die

- horizontale Integration über Ablaufketten des Leistungs- und Wertschöpfungsflusses,
- vertikale Integration über Anwendungssysteme für die verschiedenen Managementhierarchien,
- zeitliche Integration über Plan-Ist-Betrachtungen mehrerer Perioden.

Hinzu kommt die Verknüpfung von Mengen- und Wertebene und die Flexibilität der Auswertungsstrukturen.

Gemeinsame Grunddatenbestände werden sowohl von den Mengensystemen im Beschaffungs-, Produktions- und Absatzbereich als auch von den Wertsystemen des Rechnungswesens, Controlling- und Finanzbereiches genutzt.

Es werden üblicherweise vier Ebenen von Anwendungssystemen unterschieden. Administrationssysteme übernehmen die Verwaltung, Abwicklung und Abrechnung des betrieblichen Leistungsprozesses. Ziel ist die Rationalisierung der Routinetätigkeiten und Massendatenverarbeitung. Dispositionssysteme dienen der Vorbereitung menschlicher Entscheidungen auf unteren organisatorischen Ebenen. Informationssysteme präsentieren Führungsinformationen speziell für die mittleren und höheren Managementebenen. Sie selektieren und verdichten Daten aus den Administrations- und Dispositionssystemen und ergänzen diese um unternehmensextern gewonnene Informationen. Planungssysteme sollen strategische und operative Planungsprozesse unterstützen, indem sie dem Management nicht nur die Informationsauswahl und -aufbereitung erleichtern, sondern auch Methoden zur Informationsbewertung anbieten.

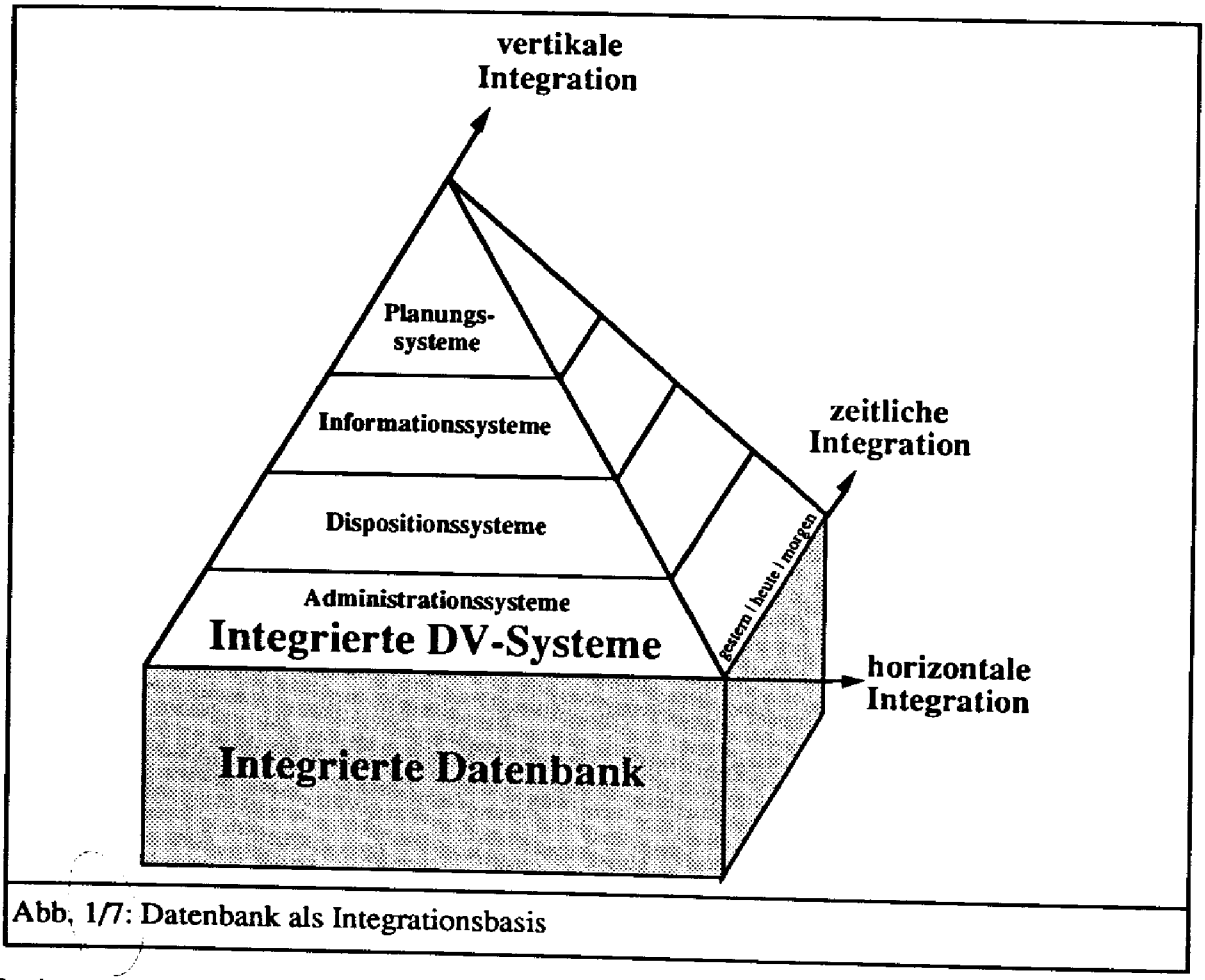


Abb. 1/7: Datenbank als Integrationsbasis

In den Unternehmen existieren auf den vier Ebenen eine Fülle von Anwendungssystemen. In einem Industriebetrieb dienen diese zum einen der Abwicklung des operativen Geschäfts vom Einkauf über die Produktion/Technik bis hin zum Verkauf, zum zweiten der Steuerung des Geschäfts. Auf dieser Steuerungsebene hängt die Ausgestaltung der Systeme davon ab, wo die Haupterfolgsfaktoren des Unternehmens liegen. Bei einem vertriebsorientierten Unternehmen werden auf dieser Dispositionsebene die Vertriebssteuerungssysteme eine große Rolle spielen, in einem mehr produktionsorientierten Unternehmen würden Produktionsplanungssysteme ein großes Gewicht einnehmen.

Die oberen beiden Systemebenen dienen der geschäftsübergreifenden Steuerung der Aktivitäten des Unternehmens durch Planungs- und Informationssysteme des Controllings.

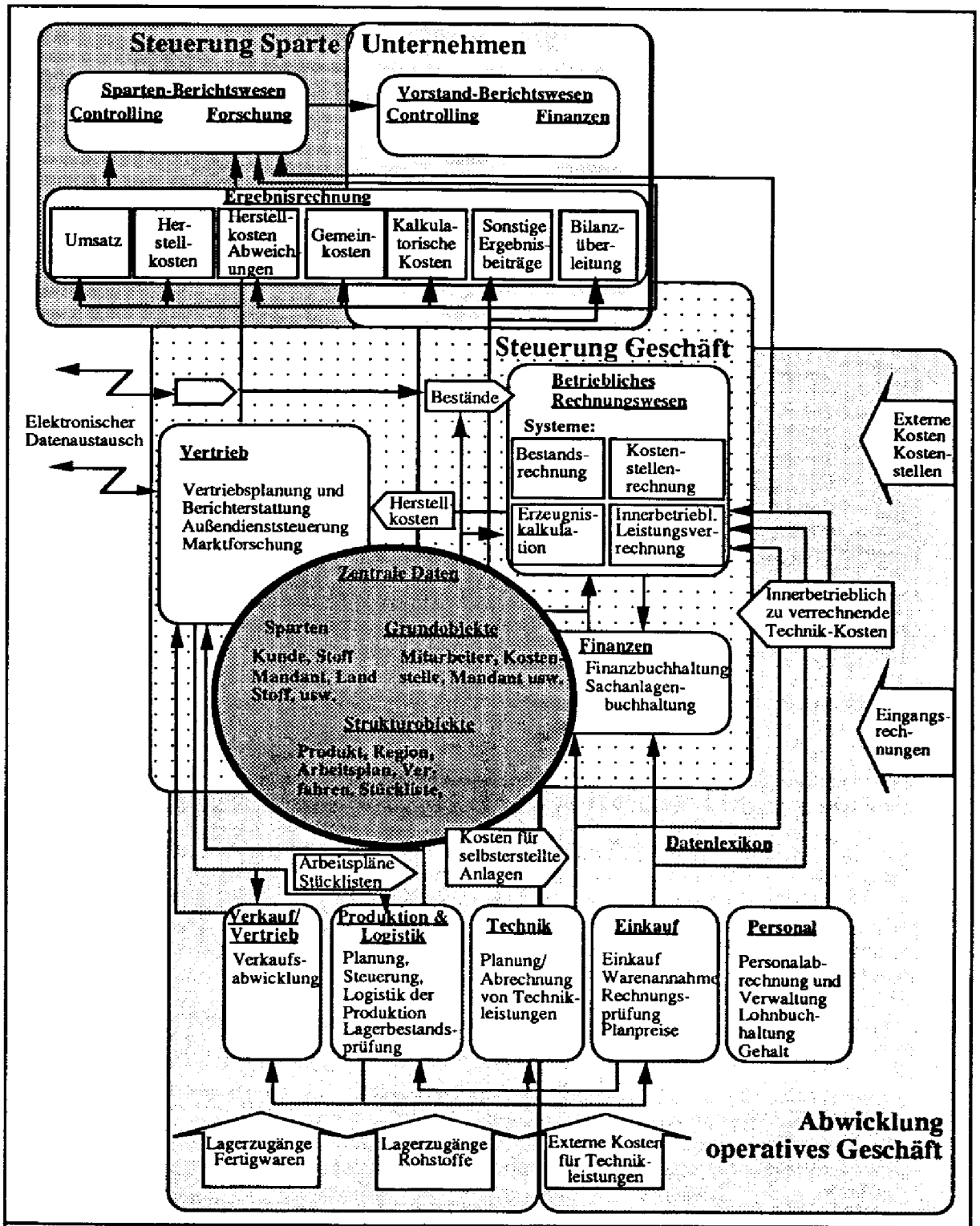


Abb. 1/8: Datenintegration bei den Anwendungssystemen eines Industrieunternehmens

(5) Datenstrukturierung schafft die Möglichkeit für Managementinformationssysteme:

Sollen DV-Systeme nicht nur administrative, sondern auch dispositive Tätigkeiten des Managements unterstützen, so sind führungsrelevante Informationen in der DV zu erfassen.

Management-Informationssysteme scheiterten bisher nicht an den mangelnden Methoden zur Auswertung von gut strukturierten Problemstellungen, sondern daran, daß die vom Management zu lösenden Probleme und die dazu notwendigen Informationen falsch eingeschätzt wurden.

Informationssysteme müssen über umfangreiche Datenbestände aus unternehmensinternen und externen Quellen und über Möglichkeiten verfügen, diesen Basis-Datenbestand durch situativ gewonnene Informationen zu ergänzen.

Sie sollten flexible, problemangepaßte Analyse- und Lernprozesse in einer sich stetig wandelnden Unternehmenswelt unterstützen und dabei bedarfsorientiert dem Management zuarbeiten.

Das Gestaltungsmotto sollte heißen „Problem Finding" statt „Problem Solving", da der Engpaß des Managements nicht darin besteht, Lösungen für gut definierte Probleme zu finden, sondern darin, Probleme zu identifizieren und mit ihren Randbedingungen, Handlungsmöglichkeiten und Ziele zu definieren.

Durch Datenmodelle lassen sich interne und externe Informationsquellen für situativ zu gewinnende Datenbestände und die Routine-Datenbestände aufeinander abstimmen und den Managementerfordernissen anpassen.

(6) Datenhaltung stellt die wesentliche strategische Leistung der Datenverarbeitung dar:

Datenbanken bieten bei der heutigen Speichertechnologie gute Möglichkeiten zur Speicherung großer Datenbestände mit schnellem, strukturiertem Zugriff.

Nur hinsichtlich programmierbarer Verarbeitungen übertrifft die Leistungsfähigkeit der elektronischen Datenverarbeitung die des Menschen. Bei nicht programmierbaren Prozessen ist die Auswertungsleistung des menschlichen Gehirns unübertroffen; da es bisher noch nicht gelungen ist, die Fähigkeit des Menschen zu Analogieschlüssen, intuitiven Vergleichen und wissensbasierten Interpretationen durch DV-Systeme nachzuahmen.

Datenintegration ist somit eine Voraussetzung für ein strategisches Informationsmanagement, das

- eine Informations- und Kommunikationsarchitektur mit integrierten zentralen und dezentralen Komponenten nutzen kann,
- interne und externe Informationserzeuger identifiziert,
- den Informationsbedarf der Unternehmung konsequent aus den Haupterfolgsfaktoren ableitet,
- die Informationsnutzer mit endnutzergerechten Analyseinstrumenten versorgt,
- die strategischen und operativen Prozesse der Unternehmensführung/Unternehmensplanung unterstützt.

2. Datenorientierte DV-Systementwicklung

2.1. Daten und Informationen

Daten (genauer Informationen) sind neben Rohstoffen und Betriebsmitteln, Kapital, Arbeit usw. eine wichtige Unternehmensressource, deren Nutzung Aufgabe eines speziellen Informations-(ressourcen)-Managements ist. Sie sind kein „freies Gut" (das nichts kostet), sondern ihre Gewinnung und Verarbeitung bindet wertvolle materielle und personelle Ressourcen.

Bei exakter Begriffsverwendung ist zu unterscheiden zwischen Informationen (als Bedeutungsinhalt von Daten) und Daten als dem Träger von Informationen (Ortner/Söllner (1989)). "Daten sind Aussagen (Informationen) über Informationsobjekte und ihre Eigenschaften auf der Basis von Fachbegriffen" (Ortner/Söllner (1989), S.83).

Informationsobjekte der Datenverarbeitung sind z. B. Kunden, Lieferanten, Rechnungen, Konten etc., deren Eigenschaften z. B. identifizierende Nummern, Namen, Orts- und Zeitangaben sind.

Fachbegriffe erklären Informationsobjekte der Fachabteilungen und deren Eigenschaften. Beispielsweise wird definiert, welche Eigenschaften ein Kunde besitzen muß, wann ein Marktpartner ein Kunde wird und wann er diesen Status wieder verliert.

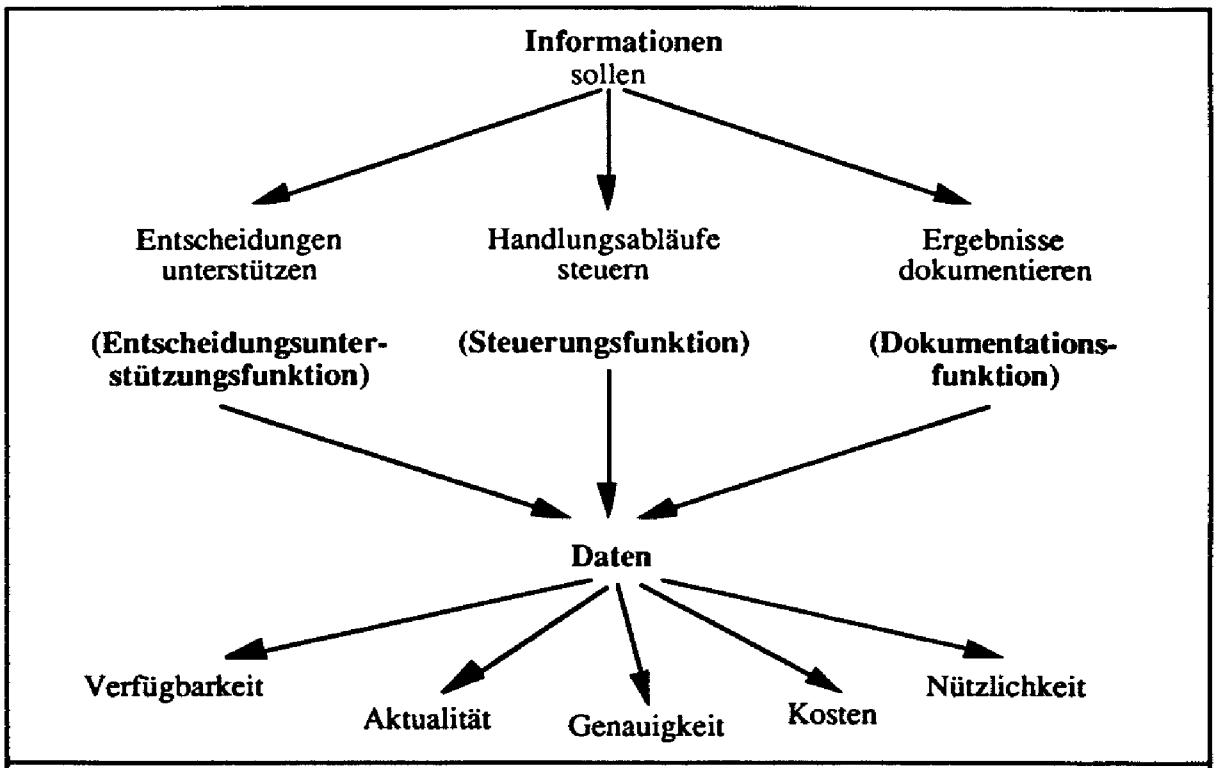


Abb. 2.1/1: Funktionen von Informationen und Anforderungen an Daten

Informationen sind aus betriebswirtschaftlicher Sicht entscheidungsorientiert zu gewinnen, zu verarbeiten und zu dokumentieren. Die Relevanz einer Information für den Entscheidungsprozeß oder für spätere Lernprozesse bestimmt deren Nützlichkeit.

Ein Informationssystem besteht aus mindestens drei Komponenten: Informationsgewinnung, der Informationsverarbeitung und der Informationsspeicherung. In allen drei Komponenten sind Entscheidungen darüber zu treffen,

-> welche Informationen,
-> auf welche Weise,
-> in welcher Häufigkeit (Frequenz) und
-> für welche Adressaten

gewonnen, verarbeitet und gespeichert werden sollen. Ein Informationssystem muß sich befassen mit Fragen

-> der Informationsgewinnung,
-> der Informationsdarstellung, z. B. Meßvorschriften,
-> der Informationsspeicherung,
-> der Informationsverarbeitung.

Im Rahmen der Informationsgewinnung sind Prozeduren zu entwickeln, wie als notwendig erachtetes Wissen aus der Umwelt der Unternehmung oder aus der Unternehmung selbst abzuleiten ist. Es kann sich dabei um primäre oder sekundäre Informationsgewinnungsprozeduren handeln. Sekundäre Prozeduren beschränken sich auf die Auswertung vorhandener Informationsquellen (z. B. Zeitschriften, Patentregister), während primäre Prozeduren sich um die aktive Gewinnung neuer Informationen bemühen (z. B. Marktforschung).

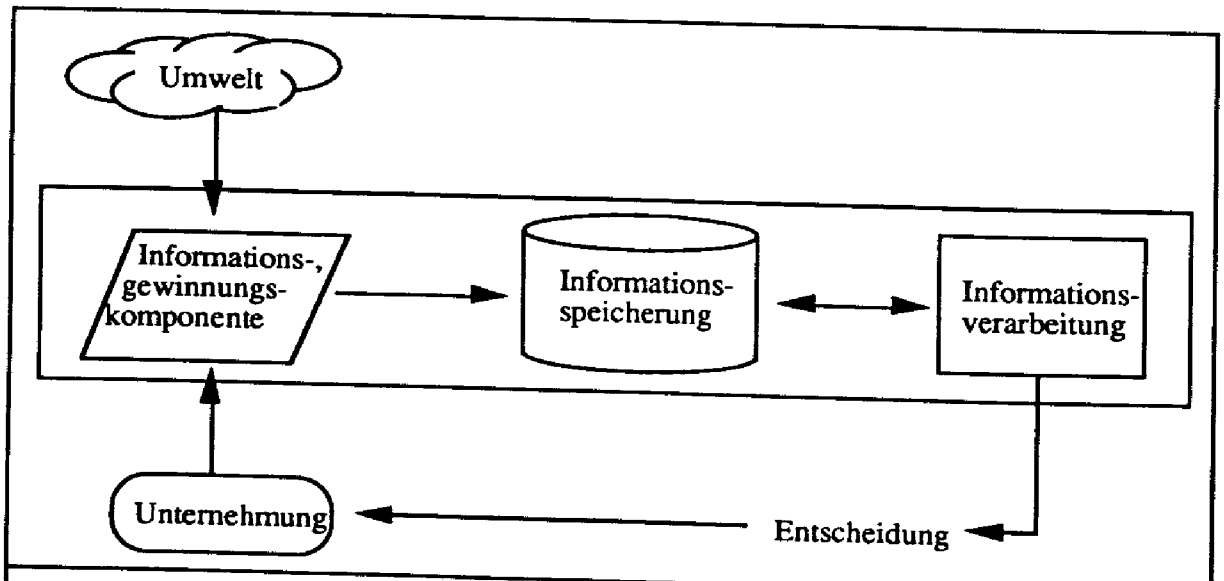


Abb. 2.1/2: Komponenten eines Informationssystems

Die Informationsspeicherung erfolgt innerhalb der elektronischen Datenverarbeitung auf zwei unterschiedliche Weisen. Bei der „dateiorientierten Datenspeicherung" wird die logische und physische Struktur der Datenhaltung von den Anwendungsprogrammen bestimmt, die auch die Funktionen Abspeicherung, Suchen und Lesen übernehmen. Bei der Verwendung einer „Datenbank" werden diese Funktionen und die physische und logische Datenstrukturierung von einem separaten DV-System (dem sogenannten „Datenbanksystem") übernommen.

Dateistruktur	Datenbankstruktur
* Verarbeitungsprogramme bestimmen die Struktur der Datenspeicherung * Datenhandling wird von Anwendungssystemen übernommen	* Struktur der Datenspeicherung unabhängig von den Verarbeitungsprogrammen * Separates DV-System übernimmt Funktionen des Datenhandling
+ Auf Verarbeitungsprogramm abgestimmte Datenhaltung, u.U. Geschwindigkeitsvorteile	+ Datenintegration = einheitliche Datenbasis für alle Anwendungsprogramme + Redundanz-Optimierung = Mehrfach verwendete Daten werden nur einmal gespeichert + Verarbeitungsprogramme werden vom Datenhandling entlastet

Abb. 2.1/3: Unterschiede zwischen Dateistruktur und Datenbankstruktur

Datenbestände können aus unterschiedlichen Blickwinkeln betrachtet werden:

- aus der Sicht des Nutzers, der damit bestimmte Fragen gelöst haben möchte, die ihn in seinem aktuellen Problemlösungsprozeß bewegen,
- aus der Sicht der Organisation, für die die Verfügung über Datenbestände bestimmte strategische Wettbewerbsvorteile bedeutet,
- aus der Sicht der DV-Systemspezialisten, die bestimmte formale Strukturen für die Abbildung von Informationen bereitstellen.
- aus der Techniksicht, für die die Struktur und Quantität der abzuspeichernden Daten bestimmte Anforderungen an die Art und Menge der vorzuhaltenen Speichermedien stellt.

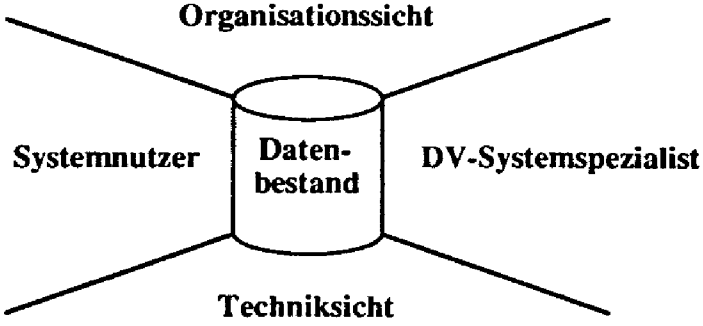


Abb. 2.1/4: Sichten auf einen Datenbestand

Aus der Sicht der DV-Technik ist zum Beispiel zu entscheiden, mit welcher Hardware- und Software-Technologie der Datenbestand gespeichert werden soll. Homogene Datenhaltungssysteme versuchen, ein einheitliches Datenbanksystem für die gesamte Datenhaltung im Unternehmen zu implementieren. Werden demgegenüber heterogene Datenbanksysteme in einem Unternehmen verwendet, so erfolgt die Vereinheitlichung nicht auf technischer, sondern auf der sogenannten „konzeptuell-logischen Ebene".

Diese Ebene spiegelt die Sichtweise der Organisation und der Systemnutzer wider. Ergebnis ist ein sogenanntes „konzeptuell-logisches Datenmodell" des Datenbestandes, das die grundsätzlichen Strukturen unbelastet von technischen Einzelheiten darstellt.

Bei der Konstruktion von betrieblichen Datenmodellen ist dabei zum einen eine „betriebswirtschaftliche", zum anderen eine „Informatik-bezogene" Perspektive zu verwenden.

Aus betriebswirtschaftlicher Perspektive läßt sich die Unternehmung beispielsweise strukturieren

- nach den eingesetzten Produktionsfaktoren (z. B. im System von Gutenberg (1983)),
- nach den betrieblichen Funktionsbereichen,
- nach den Koalitionsteilnehmern im Umfeld der Unternehmung,
- nach den Organisationseinheiten,
- nach den Mengen- und den Wertprozessen (Güter- und Geldkreislauf) sowie
- nach den Abrechnungssystemen und deren Strukturen (Buchungskreise, Kostenstellen, Kostenarten, Konten).

Aus informatikbezogener Perspektive sind wichtige Strukturierungselemente beim Datenmodell
- die Eingabe-, Verarbeitungs- und Ausgabeprozesse,
- die erforderlichen Kommunikations- und Steuerungsprozesse,
- die verfügbare Hardware- und Softwaretechnologie

2.2. Daten und Datenunabhängigkeit

Die datenorientierte Systementwicklung strebt auf verschiedenen Ebenen nach Unabhängigkeit. Zum einen wird durch ein übergreifendes (konzeptionelles) Daten-Grobmodell versucht, einen von speziellen Applikationen und Implementierungen unabhängigen Rahmen der Systementwicklung zu schaffen. Zum zweiten wird durch eine eindeutige logische Strukturierungstechnik (semantische Datenmodellierung) eine Datenstruktur gewonnen, die unabhängig von der Umsetzung in bestimmten Software- und Hardware-Systemen ist.

Ziel	Mittel
Unabhängigkeit von spezifischen Verwendungszwecken	Konzeptuelles Schema
Unabhängigkeit von DV-Systemstruktur	Semantische Datenmodellierung
Unabhängigkeit von Anwendungssystemen	Einsatz von Datenbanken

Abb. 2.2/1: Ebenen der Unabhängigkeit

Zum dritten wird schließlich durch den Einsatz von Datenbanken erreicht, daß die Datenstrukturen unabhängig von Anwendungsprogrammen sind.

(1) Unabhängigkeit von Verwendungszwecken durch konzeptionelles Daten-Grobmodell

Bei der konzeptionellen Datenmodellierung werden Sachverhalte und Tatbestände des zu modellierenden Realitätsausschnittes in "logisch-abstrakter", allgemeingültiger Form festgehalten. Dabei bemüht man sich um eine eindeutige syntaktische und semantische Struktur der Aussagen und um eine einheitliche logische Modellierungstechnik. Das

Datenmodell dient als zentraler Bezugspunkt für applikationsnahe logische und physische Datenstrukturen.

In einem logisch orientierten und eindeutig strukturierten Top-Down-Ansatz wird zunächst ein grobes, möglichst unternehmensweites Datenmodell typmäßig festgelegt. Dieses grobe Datenmodell wird dann projektbezogen detailliert und mit den anderen Teilprojekten integriert. Auf der Basis dieses globalen logischen Datenmodells werden die physischen Datenstrukturen abgeleitet, mit deren Hilfe die Datenbestände zentral oder dezentral gespeichert werden. (vgl. Vetter (1988), S. 18ff.)

(2) Unabhängigkeit von DV-Systemstruktur durch semantische Datenmodellierung

Die logische Modellierung von Datenstrukturen erfolgt unabhängig davon

- welche Datenbankschemata (primitiv, hierarchisch, relational etc.)
- welche Datenbanksysteme
- welche Rechnerarchitektur

heute/morgen zur Verfügung stehen.

Das logische Datenschema ist unabhängig von der Verteilung der Daten auf Software- und Hardwaresystemen und im Zeitablauf relativ stabil (Postulat der Datenunabhängigkeit).

Mit einem einheitlichen logischen Datenmodell wird versucht, integrierte (betriebswirtschaftliche) Gesamtlösungen mit arbeitsplatznaher, dezentraler DV-Technologie zu vereinbaren (integrierte, verteilte Datenhaltung).

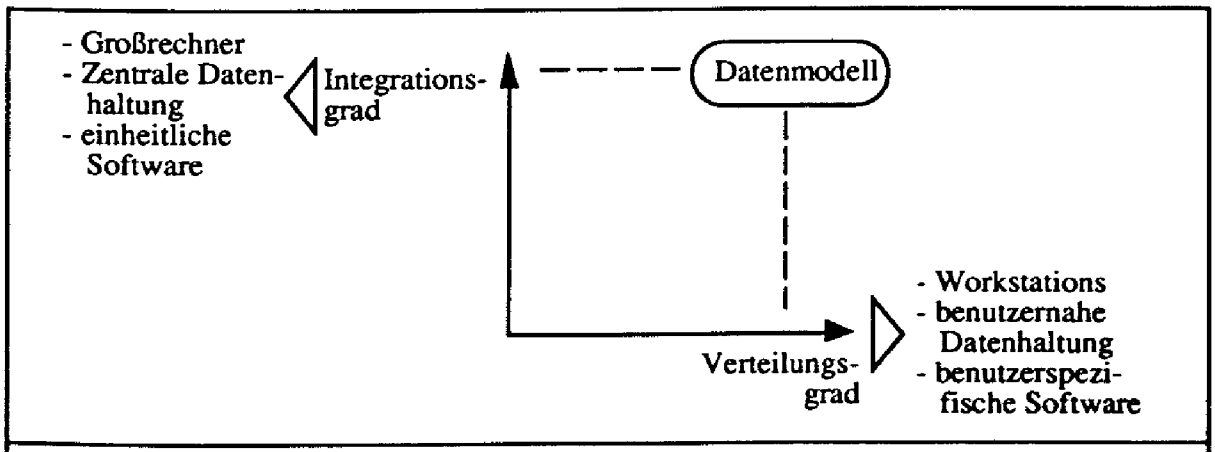


Abb. 2.2/2: Datenmodell als Weg zur logischen Integration bei hoher physischer Verteilung

Das Datenmodell bildet nicht nur eine einheitliche Orientierungsbasis für den Systementwurf, sondern es verpflichtet auch zur Einhaltung bestimmter inhaltlicher und technischer Standards (vgl. Ortner (1991)).

Die logische Konstruktion des Datenmodells wird durch die semantische Datenmodellierung (vgl. Abschnitt 2.3) unterstützt. Zum Konzept semantischer Datenmodellierung gehört zum einen die Verwendung entsprechender Instrumente der Informatik; wichtiger aber aus betriebswirtschaftlicher Sicht sind folgende Punkte:

-> die eindeutige Definition aller betriebswirtschaftlichen Fachbegriffe in ihrer unternehmensspezifischen Ausprägung,
-> deren Dokumentation in einem betriebswirtschaftlichen Begriffslexikon,
-> dessen Überleitung in ein Datenlexikon, das Grundlage für jede Stufe des Datenbankentwurfs ist.

Das Konzept dient der begrifflichen Verzahnung zwischen Fachabteilung und DV-Abteilung und als gemeinsame sprachliche Basis für die Kommunikation der an der Systementwicklung Beteiligten.

Eine solche Vorgehensweise ist auch dann sinnvoll, wenn zur Zeit keine DV-technische Implementierung eines Datenmodells beabsichtigt ist. Die einheitliche begriffliche Basis

-> rationalisiert den Entwicklungsprozeß/Modifikationsprozeß von betriebswirtschaftlichen Steuerungsinstrumenten,
-> vereinfacht Kommunikationsprozesse zwischen unterschiedlichen Managementebenen und ist ein wichtiges Element einer Corporate Culture,
-> macht Lücken/Doppelgleisigkeiten im bisherigen Steuerungssystem der Unternehmung deutlich.

(3) Trennung von Programm-/Anwendungssystem und der Datenstruktur

Als Basissoftware werden Datenbank-Systeme verwendet. Datenbanken besitzen an der Schnittstelle zwischen Daten - und - Programm eine formale Sprache, in der die Anwendungen die relevanten Daten strukturell beschreiben können und in der Operatoren zum Umgang mit den so strukturierten Daten definiert sind.

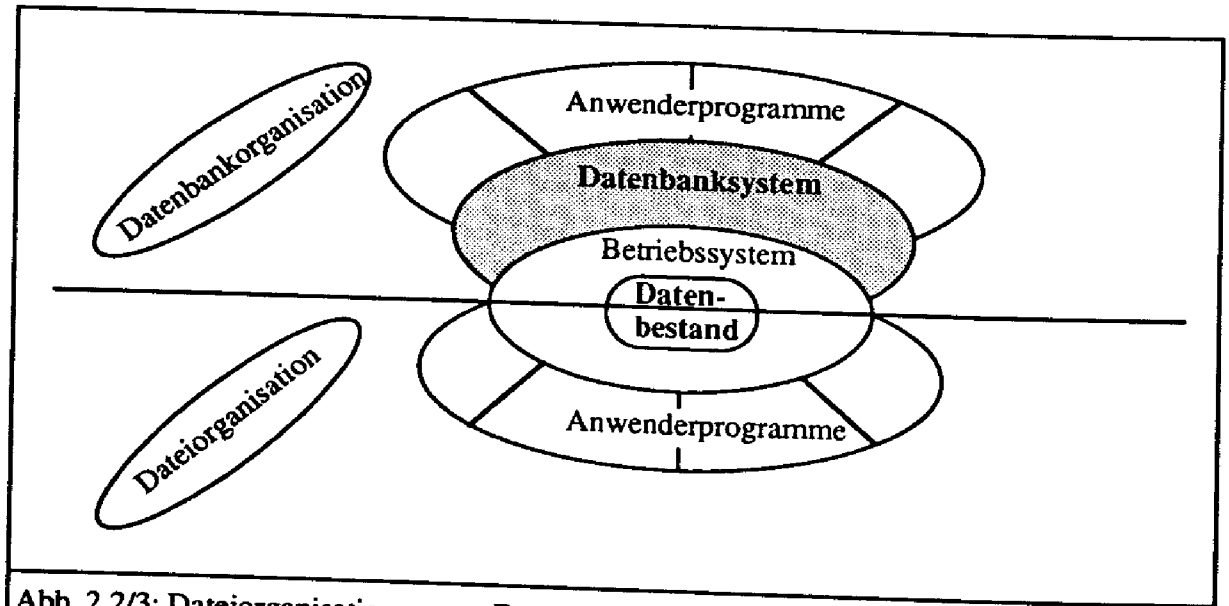


Abb. 2.2/3: Dateiorganisation versus Datenbankorganisation

Durch den Einsatz der Datenbanksysteme wird erreicht, daß die Anwendungsprogramme sich nicht um die Handhabung der Datenbestände bemühen müssen (Anwendungssysteme sind datenunabhängig). Ferner sind die Datenstrukturen unabhängig von den Anwendungssystemen, d. h. die Daten orientieren sich an den Informationsobjekten und Informationsbeziehungen im Unternehmen und nicht an den Strukturen der Anwendungssysteme. Damit sind sie auf der einen Seite stabil bei Änderungen der Anwendungssysteme, auf der anderen Seite jedoch flexibel hinsichtlich sich verändernden wirtschaftlichen, organisatorischen oder technischen Anforderungen.

Das sogenannte „Postulat der Datenneutralität" wird ergänzt durch das „Postulat der Datenintegrität" und das „Postulat der Datenintegration". Datenintegrität bedeutet, daß der einheitliche Datenbestand und die spezielle Datenverwaltungssoftware die Vermeidung widersprüchlicher Daten (Datenkonsistenz), die Datensicherung gegen technische Fehler sowie den Datenschutz gegen mißbräuchliche Verwendung erleichtert. Datenintegration heißt, daß der einheitliche Datenbestand die Mehrfachspeicherung gleicher Dateninhalte vermeidet und damit die Redundanz minimiert.

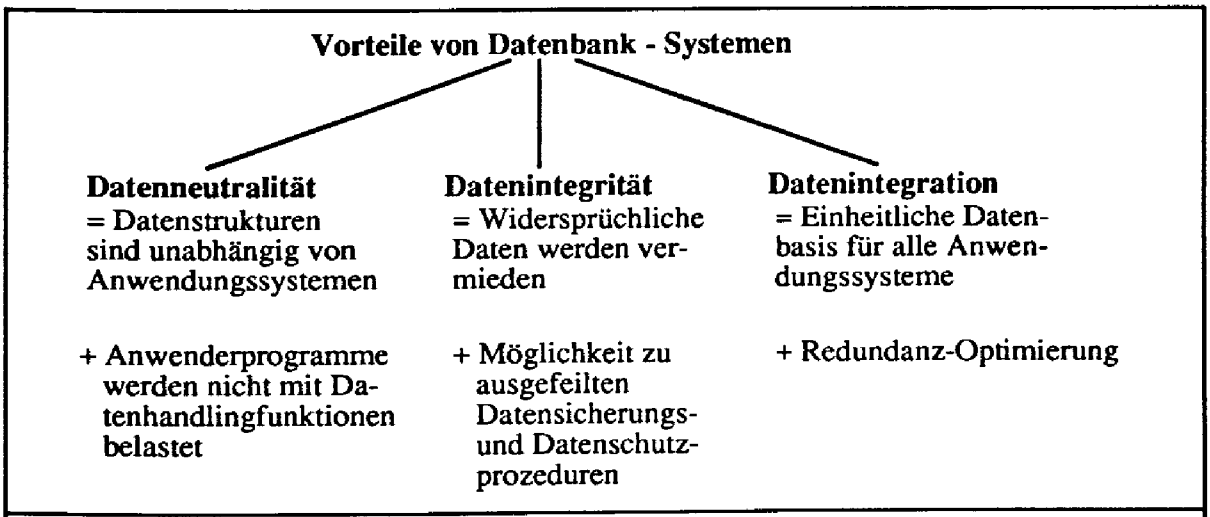


Abb. 2.2/4: Vorteile von Datenbanksystemen

3. Datenmodelle als Instrumente des Informationsmanagements

3.1. Aufgaben des Informationsmanagements

3.1.1. Informationsmanagement als Teil der Unternehmensführung

Informationsmanagement ist ein „schillernder" Begriff, der im Spannungsfeld von klassischen Unternehmensfunktionen wie Rechnungswesen, Revision und auch Marktforschung und neueren Funktionen wie Controlling, Unternehmensplanung und Datenverarbeitung steht.

Informationsmanagement ist nicht gleichzusetzen mit dem Management der elektronischen Datenverarbeitung. Die grundlegende Idee des Konzeptes ist die Integration des Informationsflusses in den Leistungs- und Wertschöpfungsprozeß der Unternehmen als drittes Element neben dem Güter- und Geldfluß. Informationen sind ein Wirtschaftsgut, mit dem ähnlich effektiv und effizient umzugehen ist, wie mit dem Geld und den traditionellen Produktionsfaktoren (vgl. Ortner (1991c)).

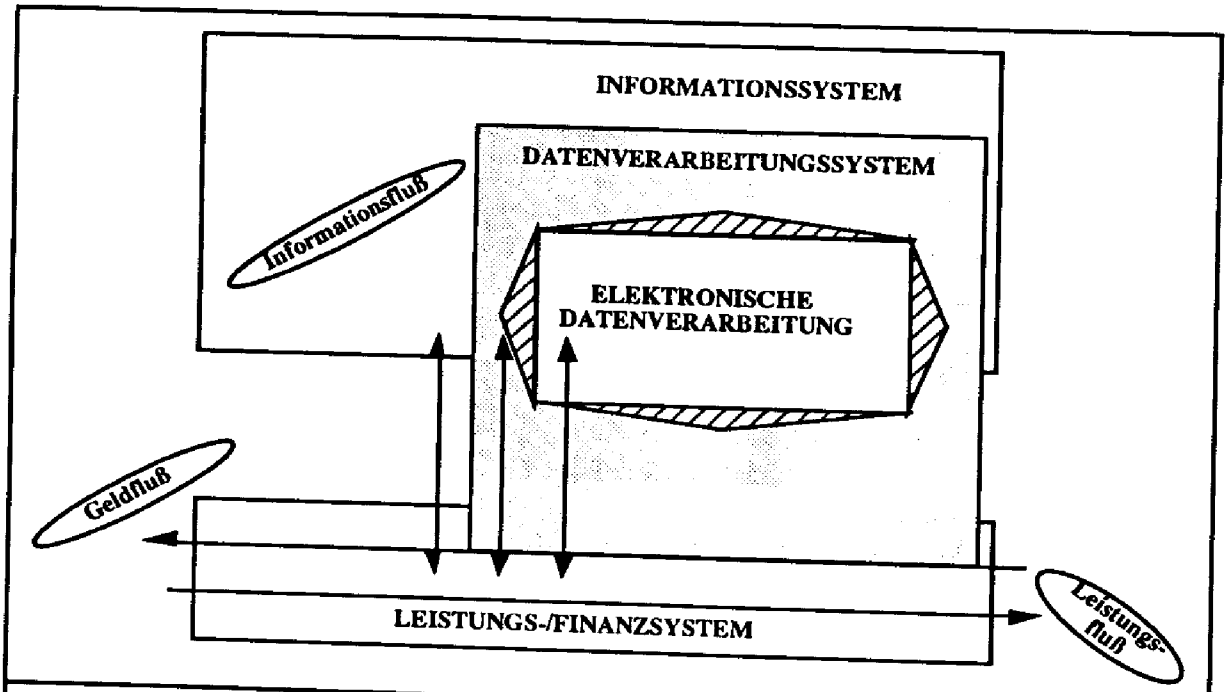


Abb. 3.1.1/1: Stellung der elektronischen Datenverarbeitung innerhalb des Informationssystems und der Unternehmung (vgl. Grochla/Meller (1974))

Nach einem Grundmodell des Informationsmanagements besteht die Unternehmung aus zwei Systemkomponenten: Im Basissystem erfolgt die Leistungserstellung und -verwertung (Güter- oder Leistungsfluß aus Beschaffung, Produktion und Absatz) sowie die Verwaltung und Abrechnung des gegenläufigen Geldflusses. In einem darüber liegenden Informationssystem werden die zur Planung, Disposition und Kontrolle des Leistungs- und Geldflusses nach wirtschaftlichen Zielen erforderlichen Informationen gewonnen und für die handelnden Personen im Unternehmen aufbereitet.

Das Informationssystem einer Unternehmung besteht aus den Personen, Organisationseinheiten und Instrumenten, die über das Geschehen im betrieblichen Leistungs- und Geldfluß nach bestimmten Abläufen informieren und diesen kontrollieren und steuern. Es ist unter Berücksichtigung des Informationsbedarfs des Unternehmensmanagements und der verfügbaren Informationsquellen im Unternehmen und in dessen Umfeld nach wirtschaftlichen Kriterien zu gestalten.

Mit dem Einsatz der elektronischen Datenverarbeitung wird grundsätzlich die Leistungsfähigkeit des betrieblichen Informationssystems gesteigert. Diese Leistungssteigerung resultiert aus der Eignung der EDV zur dauerhaften Speicherung und zur Kommunikation großer Datenmengen sowie zu effizienten und schnellen Auswertungsoperationen. Allerdings beschränken sich diese Fähigkeiten auf gut strukturierte Daten und programmierbare Operationen. Aus diesem Grunde sind EDV-Systeme trotz vieler Anstrengungen nur begrenzt zur Unterstützung von Top-Management Entscheidungen geeignet (vgl. Pohle (1990)).

Auch der Einsatz der elektronischen Datenverarbeitung hat eine personelle, eine organisatorische und eine instrumentelle Komponente. Die personelle Komponente beschreibt die Fähigkeiten und Fertigkeiten der in der Unternehmung mit der Datenverarbeitung befaßten Personen. In der organisatorischen Komponente wird zunächst die Aufbauorganisation der mit der Informationsverarbeitung beauftragten Aufgabenträger (Personen, Maschinen),gekennzeichnet. In der instrumentellen Komponente wird festgelegt, welche Programmsysteme auf welcher DV-Technologie betrieben werden.

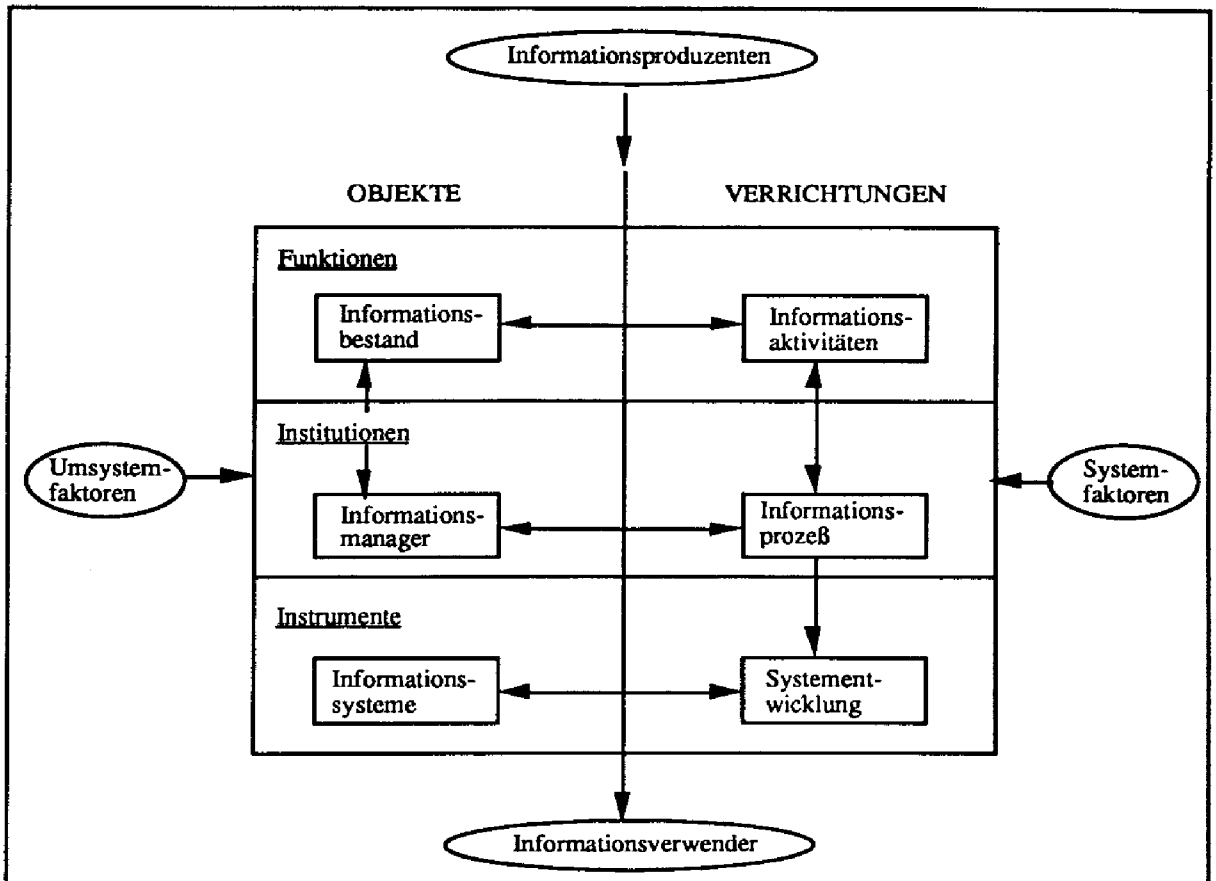


Abb. 3.1.1/2: System des Informationsmanagements (vgl. Horvath (1990), S. 199)

Das Informationsmanagement läßt sich in drei Sichten einteilen:

In der funktionalen Sicht werden die Aktivitäten betrachtet, die eine Unternehmung bei der Beschaffung, Verarbeitung, Speicherung und Verwendung von Informationen betreibt und deren Ergebnis der Bestand an Informationen ist.

In der institutionalen Sicht werden die dazu verankerten Elemente der Aufbau- und Ablauforganisation erfaßt, d. h. welche Organisationseinheiten betreiben welche Teilaufgaben bei der Beschaffung, Verarbeitung, Speicherung und Verwendung von Informationen. Da letztlich damit alle Organisationseinheiten und Mitglieder der Unternehmung angesprochen sind, sind spezifisch die Elemente gemeint, deren Hauptaufgabe in der Bildung und Koordination der entsprechenden Institutionen liegt.

Die Instrumente dienen schließlich den Institutionen zur Erfüllung ihrer Aufgaben. Institutionen sind in den Unternehmen beispielsweise das Rechnungswesen, das Controlling und die DV-Abteilung.

Das Informationsmanagement umfaßt alle Planungen und Handlungen hinsichtlich der Informationsgewinnung, der Informationsverarbeitung und Informationsverwendungen. Diese haben jeweils einen

-> inhaltlichen Aspekt (z. B. der Relevanz der Informationsinhalte)
-> zeitlichen Aspekt (der Rechtzeitigkeit und Aktualität)

-> personellen Aspekt (der Adressaten und der Quellen)

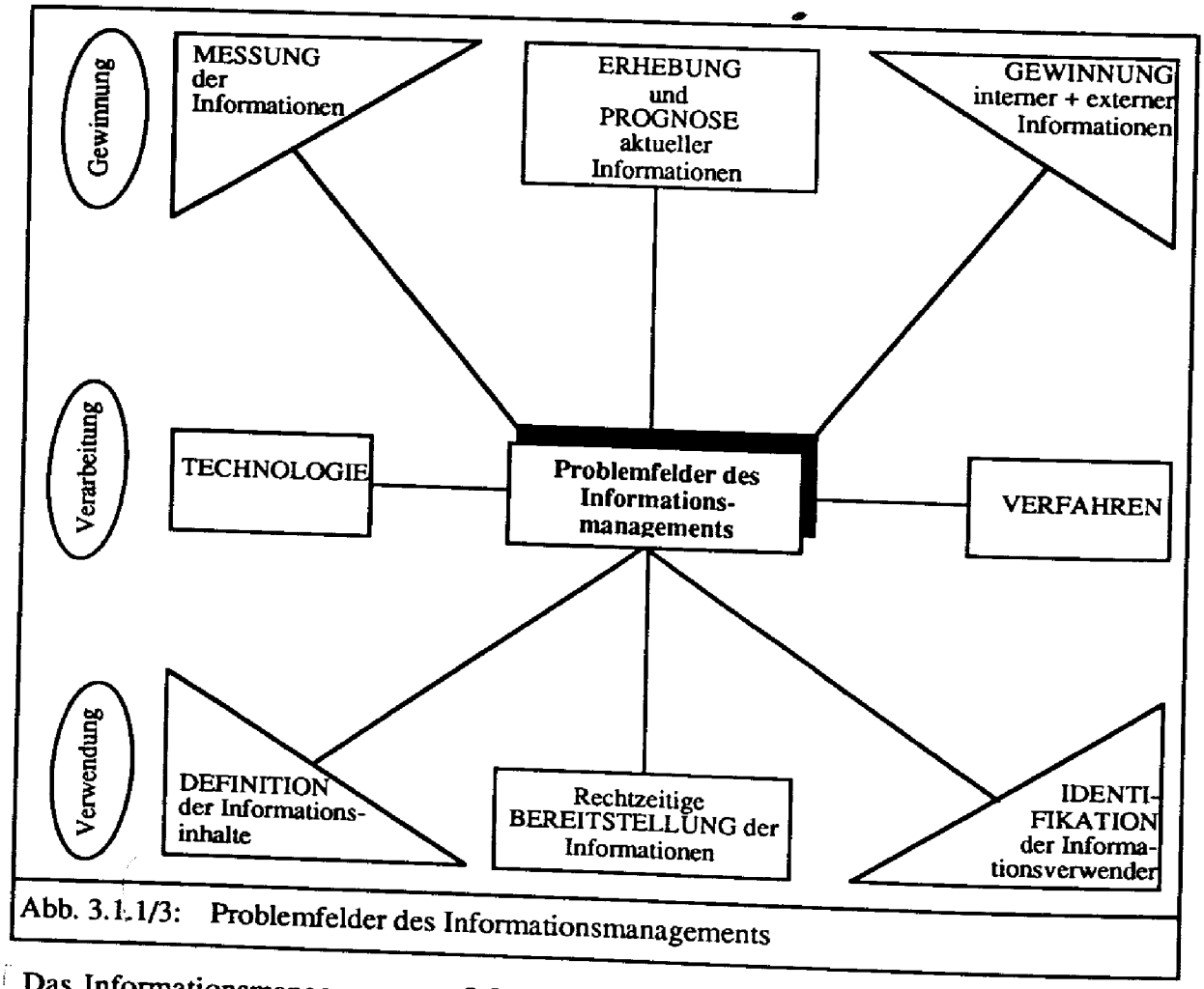


Abb. 3.1.1/3: Problemfelder des Informationsmanagements

Das Informationsmanagement muß Informationen als Wirtschaftsgut verstehen, das wie andere Ressourcen der Unternehmung bewertet werden muß, d. h. die Kosten der Gewinnung und Verarbeitung müssen durch den Nutzen der Verwendung der Informationen gedeckt sein.

Das Informationsmanagement hat unabhängig von seiner aufbauorganisatorischen Zuordnung folgende Aufgaben:

3.1.2. Teilaufgaben des Informationsmanagements

1. Ableitung der erforderlichen Informationssysteme aus strategischen Unternehmenszielen

Die strategischen Erfolgspositionen eines Unternehmens sollen letztlich das Handeln des Informationsmanagements bestimmen.
Sie bilden die Grundlage für die Gestaltung der Informationssysteme einer Unternehmung, zu denen maßgeblich die Datenbestände und Programmsysteme gehören. Strategische Erfolgspositionen können in den Produkten, in den Märkten oder in den Funktionen einer Unternehmung liegen. Beispielsweise kann ein PKW-Hersteller führend in der Produkt-

technologie sein (z. B. BMW), anderen Herstellern bei der Durchdringung der Märkte und im Vertriebsweg überlegen sein (z. B. VW) oder einzelne Unternehmensfunktionen besser beherrschen (z. B. kostengünstigere Produktion japanischer Hersteller).

Informationssysteme helfen mit, neue Produkte und Dienstleistungen zu schaffen, sie ermöglichen durch Automatisierung Kostenvorteile und sie eröffnen verbesserte Kommunikationsmöglichkeiten mit Kunden und Lieferanten. (vgl. Mertens/Schuhmann/Hohe (1988))

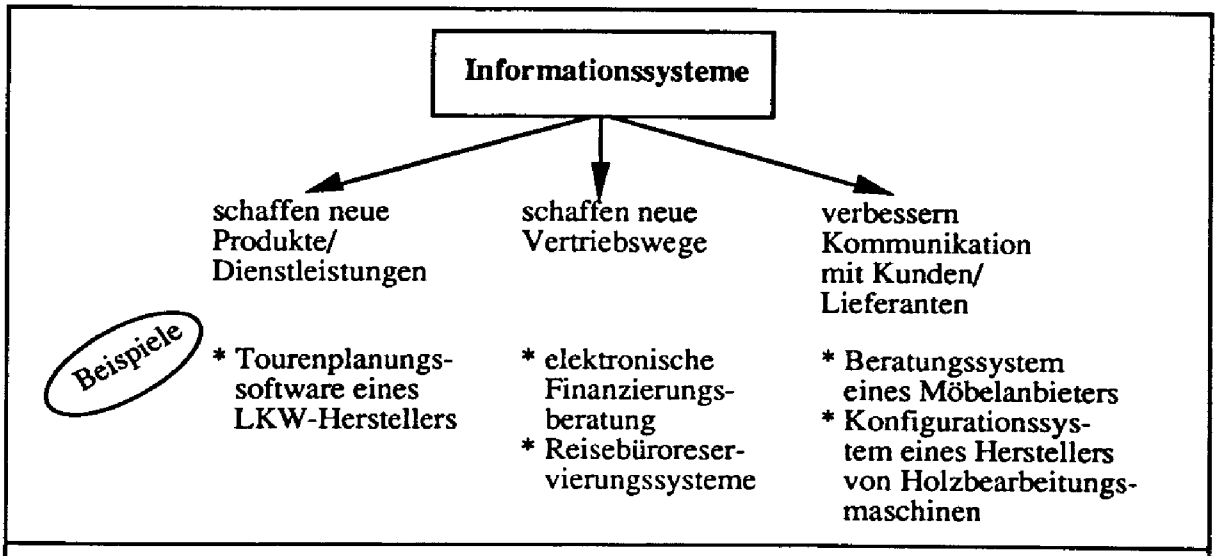


Abb. 3.1.2/1: Strategische Wirkungen von Informationssystemen

Informationssysteme müssen somit konsequent an den strategischen Erfolgspositionen einer Unternehmung orientiert werden. Informationen sind kein Wert an sich, sondern ein Produktionsfaktor zur besseren Erreichung der Unternehmensziele.

Dazu ist es für das Informationsmanagement erforderlich, sich bewußt mit den Unternehmenszielen und der Erfolgsposition der Unternehmung auseinanderzusetzen und aus diesen die Haupterfolgsfaktoren und die Hauptsteuerungsgrößen abzuleiten.

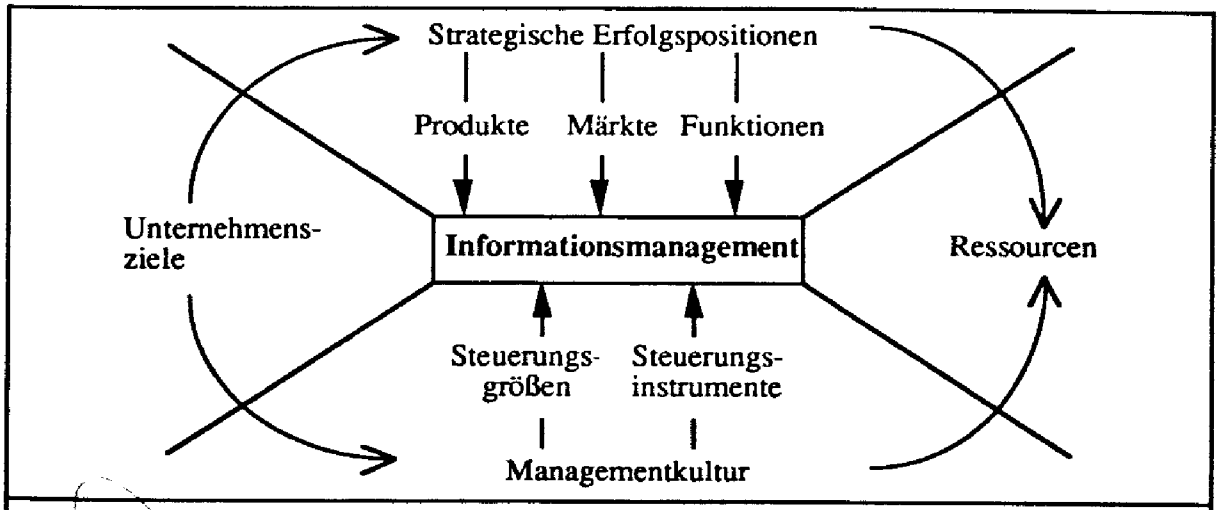


Abb. 3.1.2/2: Randbedingungen des Informationsmanagements

Üblicherweise sind dabei die Erfolgsfaktoren nach Funktionsbereichen und Geschäftsfeldern weiter zu differenzieren. Ergebnis ist eine Matrixbetrachtung der Haupterfolgsfaktoren.

		Geschäftsfelder (Beispiele)		
		PKW	LKW	Flugzeugbau
Funktions-bereiche	Forschung	Produkt-innovation	Erprobung umfassende Tests	Spezialisierung auf Kundenwünsche
	Produktion	Ausstattungs-vielfalt	Qualitäts-sicherung	Kosten
	Vertrieb	Kundendienst-verbreitung	weltweiter Kundendienst	Reaktionszeit des Kundendienstes

Abb. 3.1.2/3: Erfolgsfaktoren gegliedert nach Geschäftsfeldern und Funktionsbereichen

Die Ausrichtung der Informationssysteme an den strategischen Unternehmenszielen soll die Informationsversorgung der Managementebenen am Bedarf orientieren und dadurch die Informationsüberflutung vermeiden sowie DV-Fehlinvestitionen vermeiden.

Porter/Millar verwenden zur Systematisierung der Unternehmenszielwirkungen der Ressource "Information" eine Informations-Intensitäts-Matrix, die die Informationsdurchdringung der Wertschöpfungskette und den Informationsgehalt der Produkte bzw. Dienstleistungen abbildet.

		Informationsgehalt des Produktes		
		niedrig	**mittel**	**hoch**
Informations-durchdringung der Wertschöpfungskette	**niedrig**	Steinbruch	Steakhouse	
	mittel	Schokoladen-herstellung	Bierbrauerei	PC-Herstellung
	hoch	Ölraffinerie	PKW-Produktion	

Abb. 3.1.2/4: Informations-Intensitäts-Matrix nach Porter/Millar

Der Informationsgehalt der Produkte beschreibt deren Erklärungsbedürftigkeit vor und während der Nutzung eines Produktes, die Informationsdurchdringung der Wertschöpfungskette die erforderlichen Planungs- und Koordinationsprozesse im Leistungserstellungs- und Leistungsverwertungsprozeß.

Während in der Anfangsphase der betrieblichen Datenverarbeitung die Informationssysteme vornehmlich als Mittel zur Rationalisierung bisher manueller Abläufe verstanden wurden, setzt sich in den letzten Jahren zunehmend die Erkenntnis durch, daß Informationssystemen strategische Wirkungen auf den internen Leistungsprozeß der Unternehmung als auch auf den Markt und die erforderlichen Ressourcen haben.

	Marktwirkungen	**Wirkungen auf den internen Leistungsprozeß**	**Ressourcenwirkung**
Geldliche Wirkung	Umsatzwirkungen durch - Preisdifferenzierung - Produktdifferenzierung	Kostenwirkungen wie - Fertigungskostenersparnisse - Verwaltungskostenersparnisse	Investitionswirkungen durch steigenden Anteil von Informationssystemen im Vergleich zu anderen Investitionen
Zeitliche Wirkung	Zeitwirkungen wie - Verkürzung von Produktlebenszyklen - Verkürzung von Lieferzeiten	Zeitwirkungen wie - Verkürzung von Durchlaufzeiten - Senkung von Entwicklungszeiten	Hoher Zeitbedarf für die Entwicklung von Informationssystemen und entsprechende Bindung von Managementkapazitäten
Qualitative Wirkung	Qualitätswirkung wie - verbesserter Kundenservice - verbesserte Marktinformationen	Effizienzwirkungen wie - verbesserte Planungs-, Steuerungs- und Kontrollprozesse	Qualitätswirkung wie - veränderter Personalbedarf

Abb. 3.1.2/5: Zielwirkungen von Informationssystemen

Angesichts der erheblichen Ressourcen und der erforderlichen Zeiten zur Entwicklung von Informationssystemen sind diese konsequent an den wirtschaftlichen Zielen der Unternehmung auszurichten.

Beispiel:
Ein Unternehmen der Sanitärindustrie stellt ein sehr breites und tiefes Sortiment von Badewannen, Sanitärkeramik und -armaturen her. Seit einiger Zeit wird ein PC-gestütztes System zur Kommunikation mit den Kunden (Architekten/Planer, Installateure, Fachberater des Großhandels, Konsumenten) verwendet, das es diesen erlaubt

- *Badezimmer und andere Sanitärräume individuell mit graphischer Unterstützung am Rechner zu gestalten,*
- *diese dreidimensional in allen Farbkombinationen anzeigen zu lassen,*
- *die entsprechenden Produkte des Unternehmens aufgrund der Maße, Farben etc. zuordnen kann .*
- *eine Kalkulation der entstehenden Anschaffungs- und Installationskosten vornehmen kann,*
- *Anfragen und Bestellungen per elektronischer Kommunikation (über Telefon) an das Unternehmen absenden kann.*

Durch dieses System gelang es, das sehr beratungsintensive Angebot einer Vielzahl von Artikeln durch ein individuelles, virtuelles Gesamtprodukt "Neues Badezimmer" zu ersetzen. Als Sekundäreffekt reduzierten sich durch die elektronische Kommunikation die Liefer- und Durchlaufzeiten.

Solche Angebotssysteme mit Graphikkomponenten existieren in ähnlicher Form zum Beispiel bei Büromöbelherstellern oder Maschinenfabriken.

2. Schaffung stabiler, über lange Zeit existierender Informationsgrundstrukturen

Ziel ist die Schaffung über lange Zeit stabiler Strukturen, die jedoch die Möglichkeit der flexiblen, wenig aufwendigen Anpassung an dynamisch sich verändernde Anforderungen bieten.

Erreicht werden soll dadurch ein strategischer Investitionsschutz, da in den Datenbeständen und Programmsystemen erhebliche zeitliche, personelle und finanzielle Ressourcen des Unternehmens gebunden sind. Bekanntlich ist ein Hardware-Upgrade in wenigen Wochen realisierbar, ein Software-Upgrade dauert hingegen Monate und Jahre.

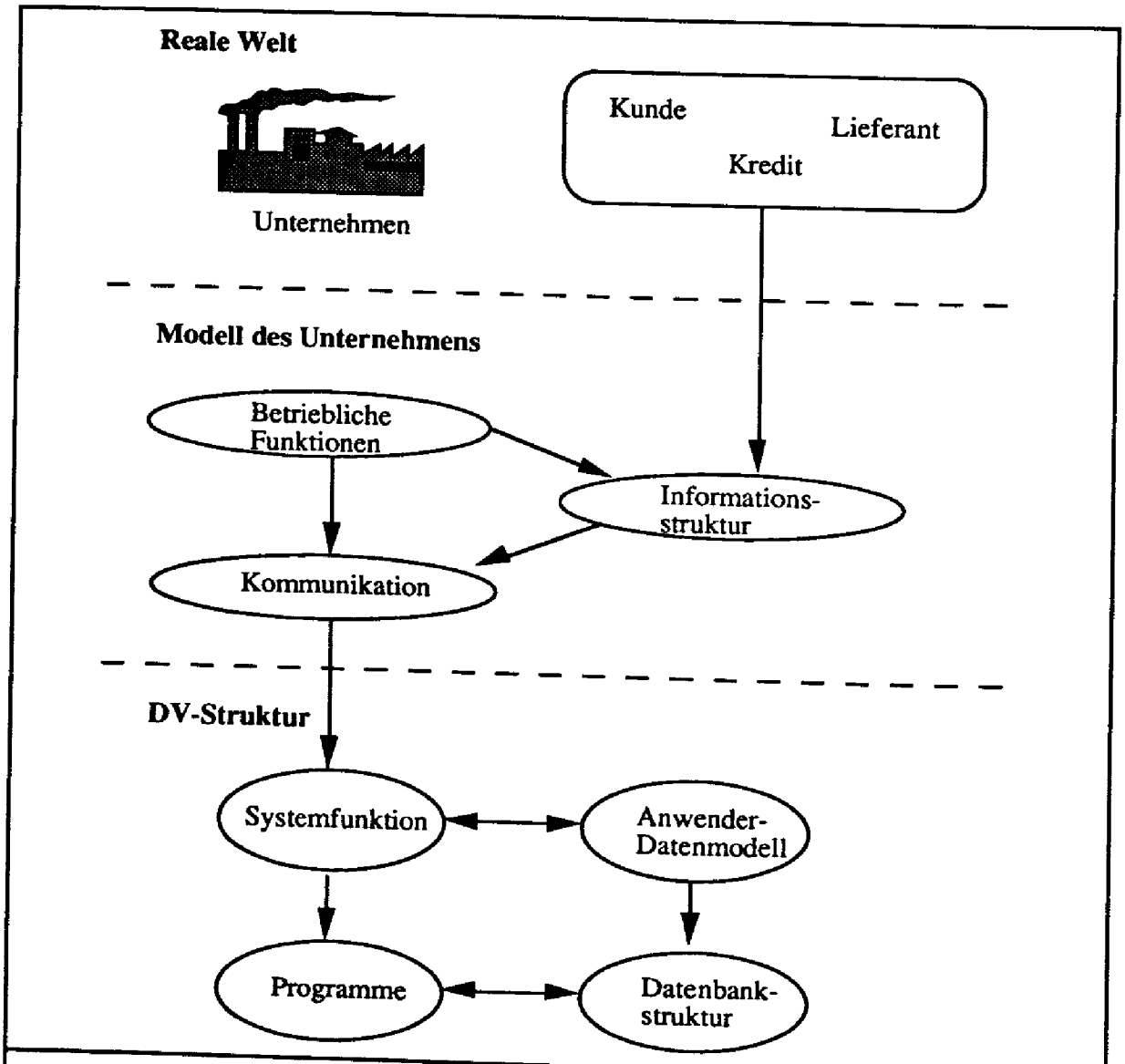


Abb. 3.1.2/6: Betriebliches Modell der Informationsverarbeitung (aus: Berner (1991))

Mit der Abbildung der Sachverhalte der realen Welt in einem Modell der erforderlichen Informationsstrukturen, der betrieblichen Funktionen und der Kommunikationsstränge sollen die logischen Zusammenhänge und fachlichen Strukturen eindeutig und unabhängig von den Gegebenheiten der DV-Struktur dargestellt werden.

Dazu sind die Informationsstrukturen im System und Umsystem der Unternehmung zu analysieren

Ziel ist die Identifizierung von Informationsproduzenten, von Informationsbeständen und von Informationsverwendern, die zentralen Einfluß auf die Erfüllung der Unternehmensziele haben.

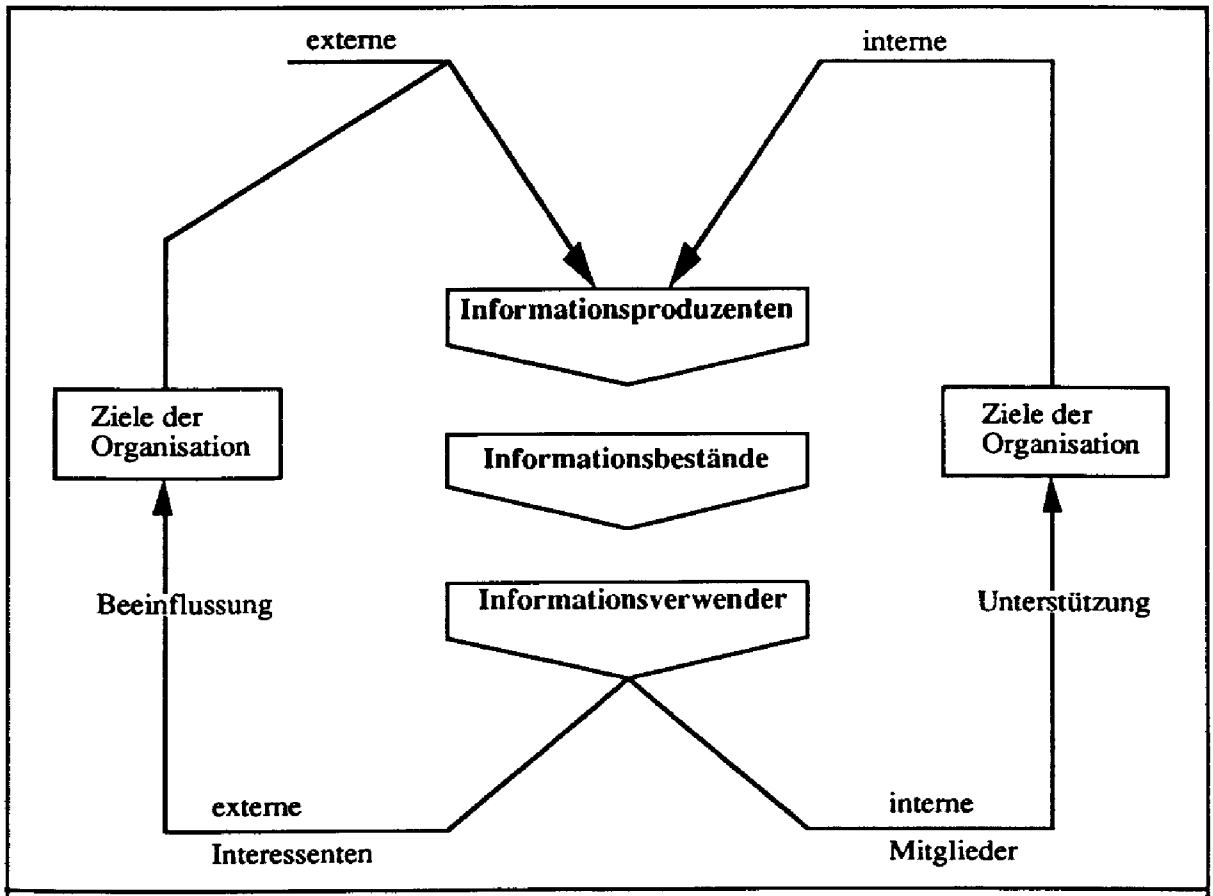


Abb. 3.1.2/7: Einfluß von Informationsproduzenten, -beständen und -verwendern auf die Erfüllung von Unternehmenszielen

Man spricht von Schlüssel-Informationsproduzenten, -beständen und -verwendern. Diese lassen sich generell in unternehmensinterne und -externe einteilen. Die unternehmensinternen Informationen werden von den DV-Anwendungssystemen geprägt.

Die Administrationssysteme leiten ihre umfangreichen Datenbestände von (manuellen oder automatisierten) Buchungsbelegen ab, zunehmend wichtiger werden EDI-Nachrichten. Die drei höheren Systemebenen bauen auf den dort gewonnenen Informationsbeständen auf, daher sind die Administrationssysteme bestimmend für Umfang und Qualität der lieferbaren Informationen.

	Informations-produzenten	**Informations-bestände**	**Informations-verwender**
Administrations-systeme	Externe Belege EDI-Nachrichten	Ist-Bezug quantitativ sehr detailliert Massendaten	Dispositions-systeme Informationssysteme
Dispositions-systeme	Administrations-systeme Prognoseverfahren	Prognosebezug quantitativ selektiert	Lower Management
Informations-systeme	Administrations-systeme Planungsprozeduren	Ist-/Planbezug quantitativ aggregiert selektiv	Middle Management
Planungs-systeme	Informationssysteme	Planbezug quantitativ/qualitativ	Top Management Planungs- u. Kontakt-Instanzen

Abb. 3.1.2/8: Produzenten und Verwender von Informationen nach den Arten von Anwendungssystemen.

Neben den internen DV-Anwendungssystemen treten als Informationsproduzenten noch externe Informationsquellen auf.

Externe Produzenten von Informationen können traditionelle (papiergebundene) (wie Bibliotheken, Marktforschungsinstitute) oder elektronische Informationsquellen wie unternehmensexterne Datenbanken oder über elektronische Kommunikation angeschlossene Unternehmen sein. sein.

Unternehmensexterne Datenbanken lassen sich in folgende Grundtypen einteilen:

-> Fakten-Datenbanken enthalten formatierte, numerische Daten z. B. Umsätze der Konkurrenz,
-> Volltext-Datenbanken enthalten neben der Referenzangabe auch den vollen Wortlaut des Dokuments,
-> Referenz-Datenbanken enthalten bibliographische Angaben über Informationsobjekte, z. T. auch noch eine kurze Inhaltsbeschreibung.

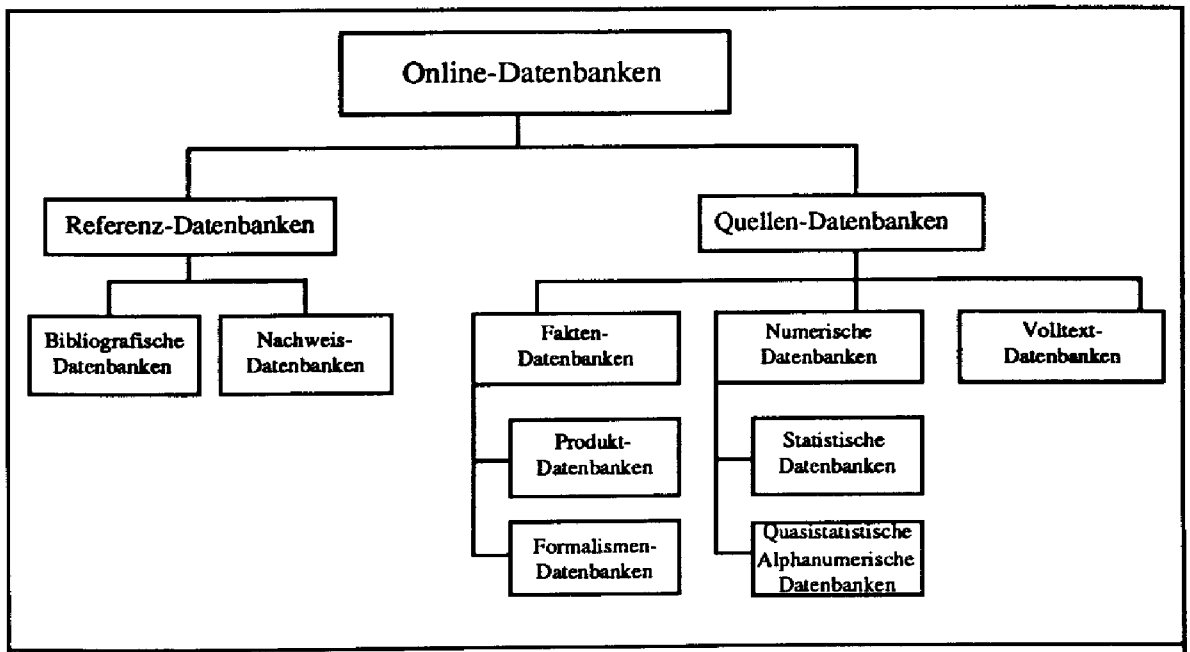


Abb. 3.1.2/9: Typen unternehmensexterner Datenbanken (aus: Detemple, Mack (1991))

Von zunehmender Bedeutung ist die elektronische Verknüpfung der internen Unternehmenssysteme mit den DV-Systemen von Lieferanten, Kunden etc. Zu differenzieren sind die elektronische Datenübertragung von der Datenintegration. Diese unterscheidet sich von der Datenübertragung dadurch, daß sich die einheitliche Struktur nicht nur auf die ausgetauschten Geschäftsnachrichten, sondern auch auf bestimmte unternehmensinterne Datenbestände erstreckt. (vgl. Schumann (1990)).

Integrationsgrad	Kennzeichen	Beispiele	Vorteile
Datenübertragung	Austausch von Geschäftsnachrichten per Datenfernübertragung	Anfrage Angebot Bestellung Bestellbestätigung Lieferavis Rechnung	Beschleunigung Fehlersenkung Geringere Bearbeitungskosten
Datenintegration	Gemeinsame Strukturierung und Nutzung von Geschäftsdaten	Artikelstammdaten	höhere Aktualität semantische Eindeutigkeit Informationsintegration Dispositionsintegration

Abb. 3.1.2/10: Datenübertragung und Datenintegration

Die Datenintegration kann sich technisch in gemeinsamen Datenbeständen ausdrücken, notwendig ist jedoch nur die unternehmensübergreifende Strukturierung der Daten.

In angloamerikanischen Literatur wird die elektronische Datenübertragung üblicherweise als EDI (Electronic Data Interchange) bezeichnet, während unternehmensübergreifende Integra-

tionsbemühungen der DV-Systeme unter den Begriff IOS (Inter Organizational Systems) subsumiert werden (vgl. Kaufman (1966), Cash/Konsynski (1985)).

Datenintegration soll die administrativen und dispositiven Tätigkeiten im Leistungsprozeß durch einen integrierten Informationsfluß besser verknüpfen. Ziel ist es, unabhängig von rechtlichen oder wirtschaftlichen Grenzen Informationen über den Leistungserstellungs- und Leistungsverwertungsprozeß am Ort ihrer Entstehung zu gewinnen und sie allen am Prozeß beteiligten Unternehmen zuzuleiten. Dazu muß eine vorgelagerte Einigung über die relevanten Auswertungsstrukturen erfolgen.

4. Ganzheitliche Abstimmung von Informationsstrukturen, Organisations- und Technologiestrukturen

Bei der Realisierung von Informationsstrukturen im Unternehmen konkurrieren DV-Informationstechnologien mit konventionellen organisatorischen Lösungen. Das Informationsmanagement hat somit eine geschäftsbezogene Dimension der Ableitung der erforderlichen Informationsstrukturen und der organisatorischen und technologischen Realisierung.

Die drei Aspekte müssen inhaltlich und zeitlich aufeinander abgestimmt werden: DV-technische Systeme ohne organisatorischen Über- und Unterbau sind ebenso zum Scheitern verurteilt, wie geschäftsbezogene Informationsstrukturen, die die Technologiepotentiale nicht nutzen.

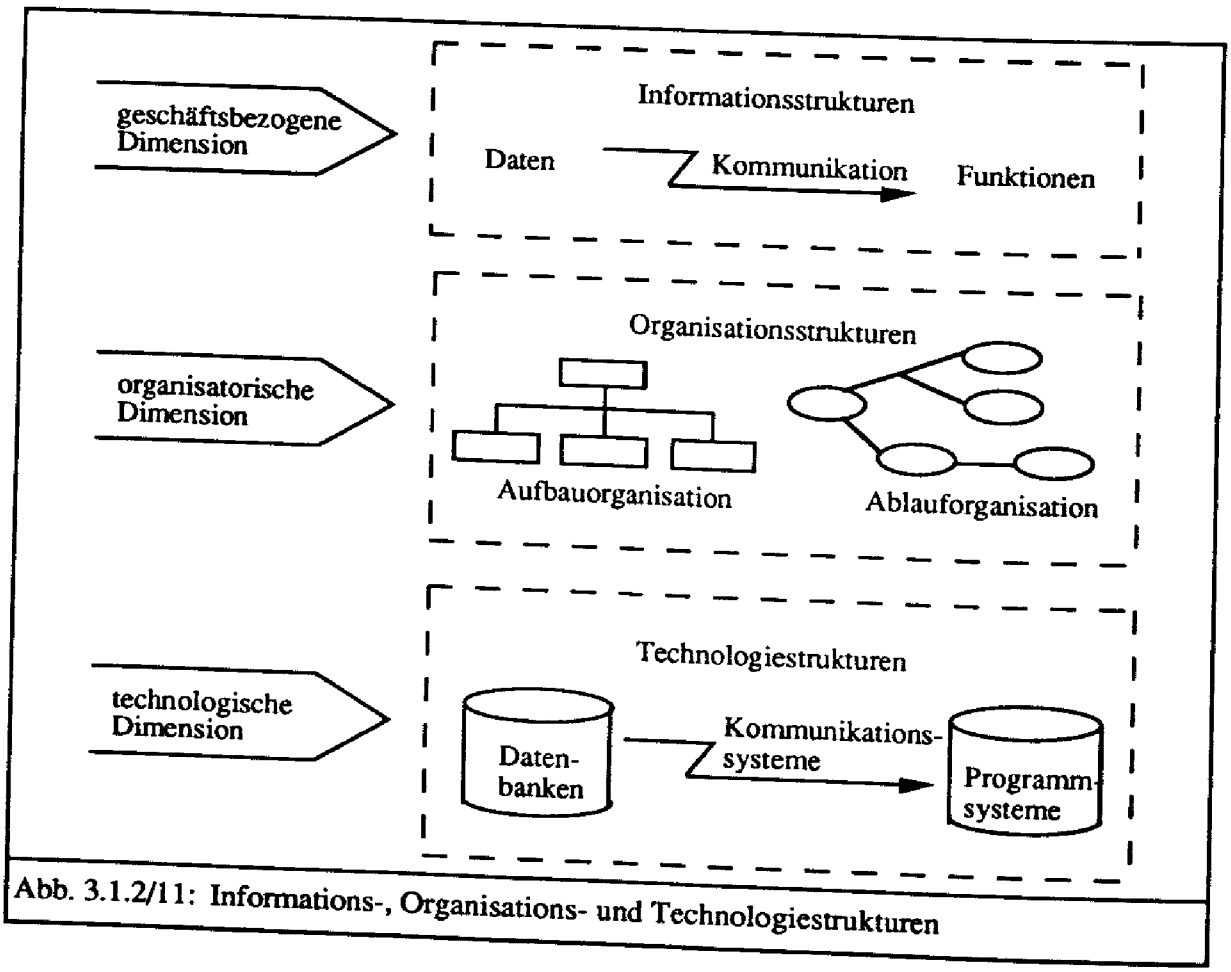


Abb. 3.1.2/11: Informations-, Organisations- und Technologiestrukturen

Die Technologiestrukturen sind dabei sowohl hinsichtlich der Hardware, der Basissoftware (z. B. Betriebssysteme, Kommunikationsprotokolle) als auch der Anwendungssoftware zu gestalten.

5. Organisatorische Verankerung einer Aufbau- und Ablauforganisation des Informationsmanagements

Das Informationsmanagement bildet eine Daueraufgabe der Unternehmensführung, die übergreifend zu den traditionell mit der Informationsbereitstellung und Informationsverarbeitung betrauten Organisationseinheiten (z.B. Rechnungswesen, Controlling, Datenverarbeitung) in der Aufbauorganisation eines Unternehmens zu verankern ist. Das Informationsmanagement integriert dabei Aufgaben, die traditionell auf mehrere Bereiche verteilt sind.

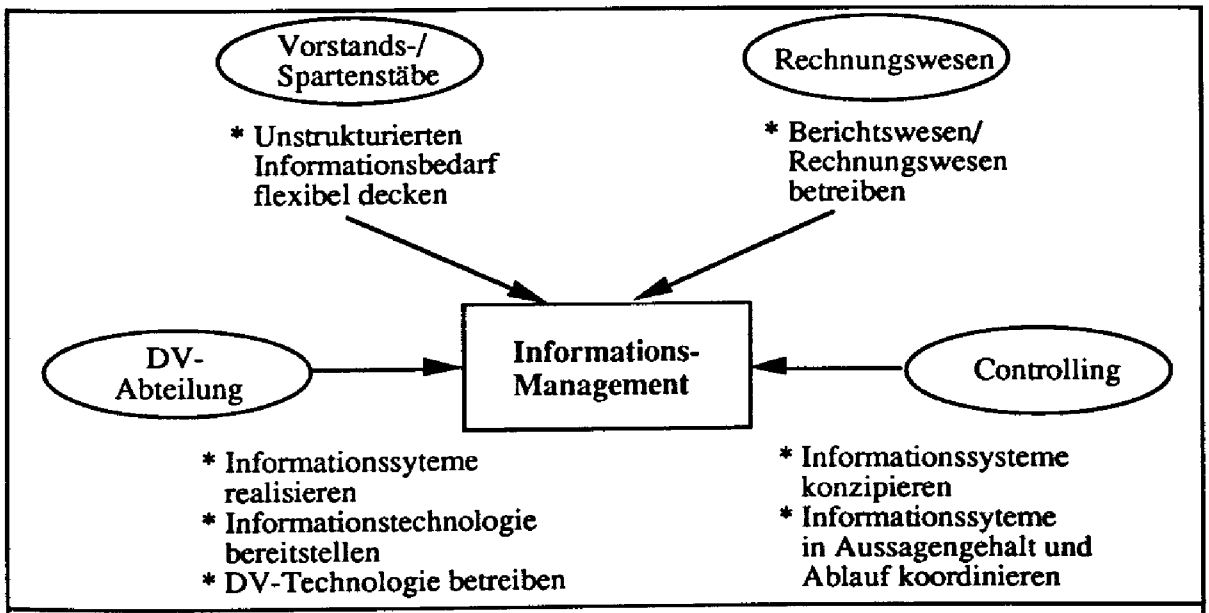


Abb. 3.1.2/12: Aufgaben des Informations-Managements

Die Aufgabenspannweite reicht dabei von der fachlichen Konzeption und systemtechnischen Realisierung bis hin zum Betrieb speziell der DV-Systeme. Eine solche weitgespannte Aufgabe der "Informationsproduktion" (Beschaffung, Verarbeitung, Versorgung) fordert die Verankerung in einer speziellen Organisationseinheit. In einigen Unternehmen übernimmt diese Aufgabe der Controller, dem die DV-Abteilung und das Rechnungswesen unterstellt wird. Andere Unternehmen bilden eigene Abteilungen "Informationsmanagement", die zumeist aus der DV-Abteilung hervorgehen.

Als zentrale originäre Aufgaben, die nicht von den traditionellen Abteilungen erfüllt werden, hat das Informationsmanagement

-> die Anforderungen der Fachabteilungen, der DV-Abteilung und der Unternehmensführung an Informationssysteme zu erfassen und in eine Systemplanung umzusetzen,
-> die Einhaltung übergreifender Standards bei der Systementwicklung, Systemdokumentation und -wartung durchzusetzen,
-> die Unterstützung der Endnutzer beim Personal Computing und Workgroup Computing zu sichern und die Integration der Endnutzer-Lösungen zum „Organizational Computing" zu leisten.

In der Praxis lassen sich 3 alternative Vorgehensweisen bei der Planung von Informationssystemen feststellen:

> das projektorientierte Vorgehen
> das strategieorientierte Vorgehen
> der Strategie-Architektur-Ansatz

Das projektorientierte Vorgehen sammelt DV-Projektanträge der Abteilungen, faßt diese in einem mehrjährigen Realisierungsprogramm (Projekt-Portfolio) zusammen und leitet daraus übergreifende DV-Grundsätze z.B. hinsichtlich der einzusetzenden Hard- und Software ab.

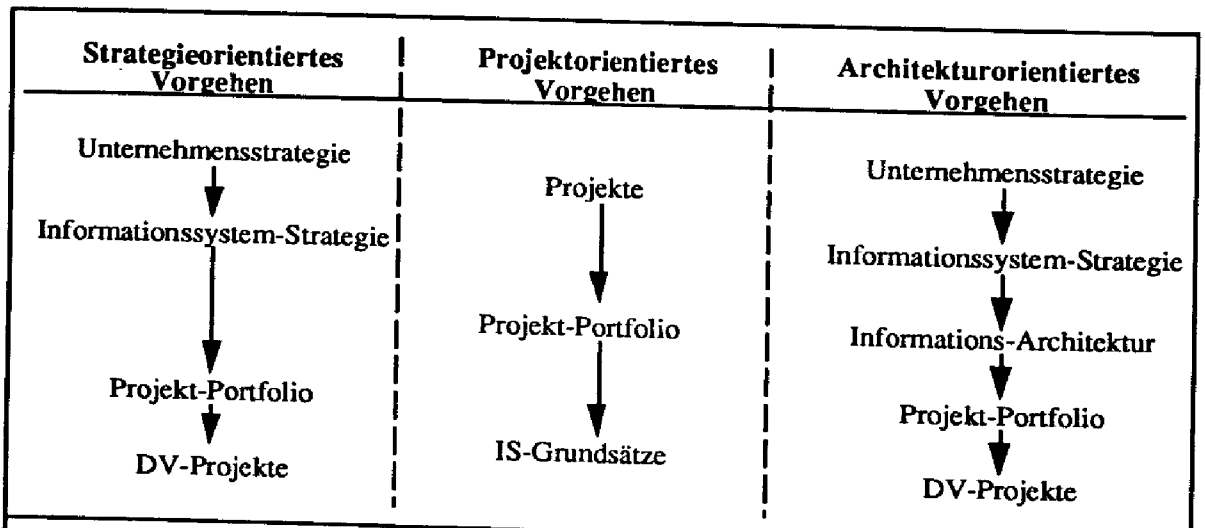


Abb. 3.1.2/13: Strategie-, projekt- und architekturorientiertem Vorgehen

Das strategieorientierte Vorgehen leitet aus der Unternehmensstrategie eine Informationssystem-Strategie ab. Daraus wird dann ein Projekt-Realisierungsprogramm (Projekt-Portfolio) entwickelt, aufgrund dessen die DV-Projekte realisiert werden.

Das architekturorientierte Vorgehen leitet aus der Unternehmensstrategie ebenfalls eine Informationssystem-Strategie ab; ergänzt diese jedoch um den Bedarf der Fachabteilungen. Daraus entsteht eine System-Architektur aus betriebswirtschaftlich notwendigen

-> Datenbeständen (Data Architecture)
-> Funktionen (Functional Architecture)
-> Kommunikationselementen (Communication Architecture)

für alle Fachabteilungen und Managementebenen. Diese betriebswirtschaftlichen Teil-Architekturen werden dann aus DV-Sicht verfeinert zu

-> der System Architektur (Systems Architecture), die aus Sicht der erforderlichen Software-Systeme die Datenbanken, Kommunikations- und Anwendungssysteme beschreibt
-> der Technologie Architektur (Technological Architecture), die die erforderlichen Hardware-Bestandteile kennzeichnet.

Aus dieser Architektur wird dann wiederum ein Realisierungsprogramm entwickelt. Dieses Vorgehen entspricht den Forderungen des Informationsmanagements, da es strategische und fachliche Überlegungen berücksichtigt und betriebswirtschaftliche mit DV-systemischen Aspekten integriert.

3.2. Informationssysteme als Instrumente des Informationsmanagements

3.2.1. Sichten von Informationssystemen

3.2.1.1. Nutzersystem, Objektsystem und Informationssystem

Bei der Entwicklung von Informationssystemen wird häufig zwischen dem Nutzersystem, dem Objektsystem und dem Informationssystem unterschieden.

Das Nutzersystem wird von der gesamten Unternehmung gebildet. Es faßt die Anforderungen der menschlichen und maschinellen Nutzer an die Inhalte, Zeiten und Formen der zu liefernden Informationen zusammen.

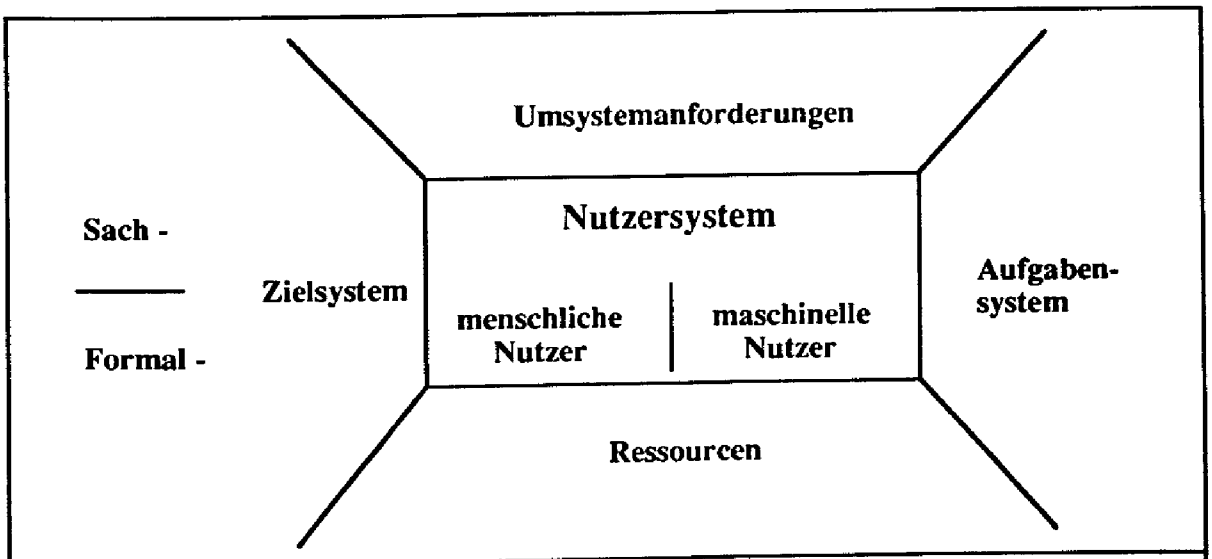


Abb. 3.2.1/1: Nutzersystem

Das Nutzersystem ergibt sich aus der Analyse der Gesamtaufgabe der Unternehmung vor dem Hintergrund der Formal- und Sachziele, der Ressourcen und der Umsystemanforderungen.

Das Objektsystem bildet den Ausschnitt der Unternehmung ab, der im Rahmen der DV-Systementwicklung relevant ist. Das Objektssystem wird gekennzeichnet durch die relevanten Ressourcen und die Aufbau- und Ablauforganisation. Zu den Ressourcen gehören unter anderem die Mitarbeiter, die maschinelle Ausstattung und die räumlichen Gegebenheiten dabei auch die vorhandene Informationstechnologie. Die Strukturen der Aufbau- und Ablauforganisation beschreiben nicht nur die Zuordnung der Teilaufgaben auf die Ressourcen, sondern auch die existierenden formellen Informationsflüsse.

Beispiel:
Für ein Informationssystem zur Verfolgung des Materialflusses in der Fertigung sind sämtliche Produktions-, Transport- und Lagereinrichtungen sowie deren räumliche Anordnung, die bei der Durchführung und Disposition beteiligten Mitarbeiter und deren Hilfsmittel (etwa Personal Computer) von Belang. Im Rahmen der Aufbauorganisation interessiert die Gliederung der Fertigung in Stellen, Gruppen und Abteilungen.

Das Informationssystem soll die Gegebenheiten des Objektsystems in der vorhandenen Informationstechnologie erfassen. Damit sind nicht notwendigerweise nur Technologien der elektronischen Datenverarbeitung gemeint, sondern durchaus auch andere Medien oder klassische Informationsinstrumente.

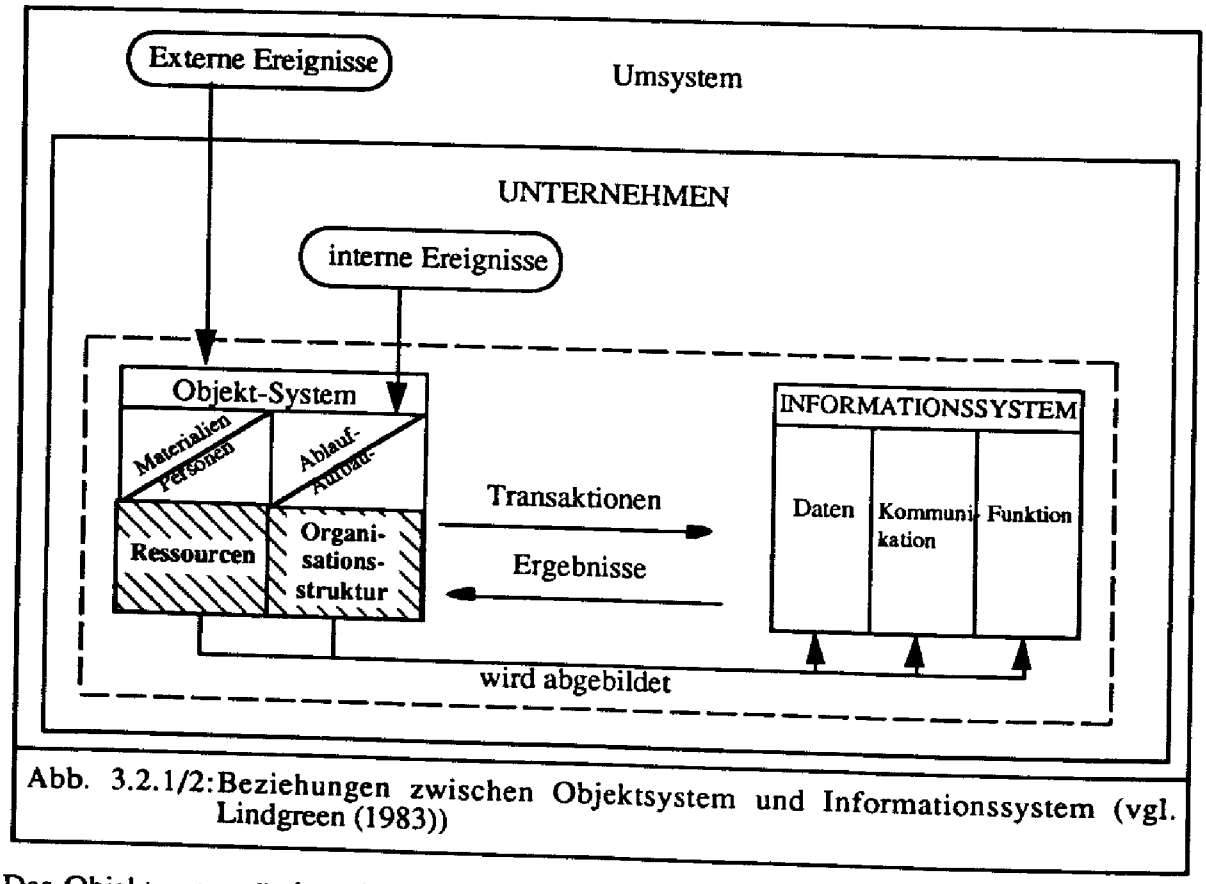


Abb. 3.2.1/2: Beziehungen zwischen Objektsystem und Informationssystem (vgl. Lindgreen (1983))

Das Objektsystem ändert sich im Rahmen der Systementwicklung. In der Realität ist es häufig so, daß der Entwicklungsaufwand zum größeren Teil durch Veränderungen des Objektsystems, z. B. der Aufbau- und Ablauforganisation oder Fertigungstechnologie, bewirkt wird und weniger durch das eigentliche Informationssystem.

Externe oder interne Ereignisse führen im Objektsystem zu bestimmten Ergebnissen oder bewirken dort wiederum Ereignisse. Sie schlagen sich in realen Objekten und in Informationsobjekten nieder.

Beispiel:
Das externe Ereignis "Wareneingang" führt zum einen zu den realen Objekten "Paletten auf der Lagerrampe" als auch zu dem Informationsobjekt "Lieferschein".

Die realen Objekte sowie die Informationsobjekte werden über Transaktionen in Daten des Informationssystems abgebildet; im Beispiel führt der Wareneingang etwa zu einer Eingabe der Merkmale des Lieferscheins und vielleicht ergänzend der Paletten in einem Materialwirtschaftssystem. Dadurch werden die Elemente des Objektsystems zu Daten im Informationssystem.

Aus diesen Datenobjekten produziert das Informationssystem aufgrund der dort verankerten Verarbeitungsfunktionen Ergebnisse. Die Steuerung der Verarbeitung kann dabei wiederum durch Transaktionen, durch interne Ereignisse oder durch Zeitablauf erfolgen.

Steuerungsform	Kennzeichen	Beispiel
Ereignisgesteuert	ein internes oder externes Ereignis führt zum Start der Verarbeitung	Der Verbrauch einer Mengeneinheit an einer Maschine führt zur Neuberechnung des Bestandes
Transaktionengesteuert	eine Transaktion vom Objektsystem an das Informationssystem startet die Verarbeitung	Die Verbuchung des Wareinganges führt zur Berechnung des Durchschnittswertes des Bestandes
Zeitgesteuert	der Ablauf eines bestimmten Zeitabschnitts oder der Eintritt eines bestimmten Zeitpunkts startet die Verarbeitung	Der Ablauf eines Monats führt zur Berechnung des Monatsdurchschnittswertes des Bestandes

Abb. 3.2.1/3: Steuerung der Verarbeitung im Informationssystem

Die Ergebnisse aus dem Informationssystem werden wiederum verwendet, um Entscheidungen innerhalb des Nutzersystems zu treffen.

3.2.1.2. Daten-, Kommunikations- und Funktionssicht

Nutzersystem, Objektsystem und Informationssystem werden im Zuge der Systementwicklung in Modellen abgebildet, die bestimmte verkürzte Sichten des Originals darstellen.

Scheer (1991) unterscheidet folgende Sichten eines vorgangskettenorientierten Informationsmodells:

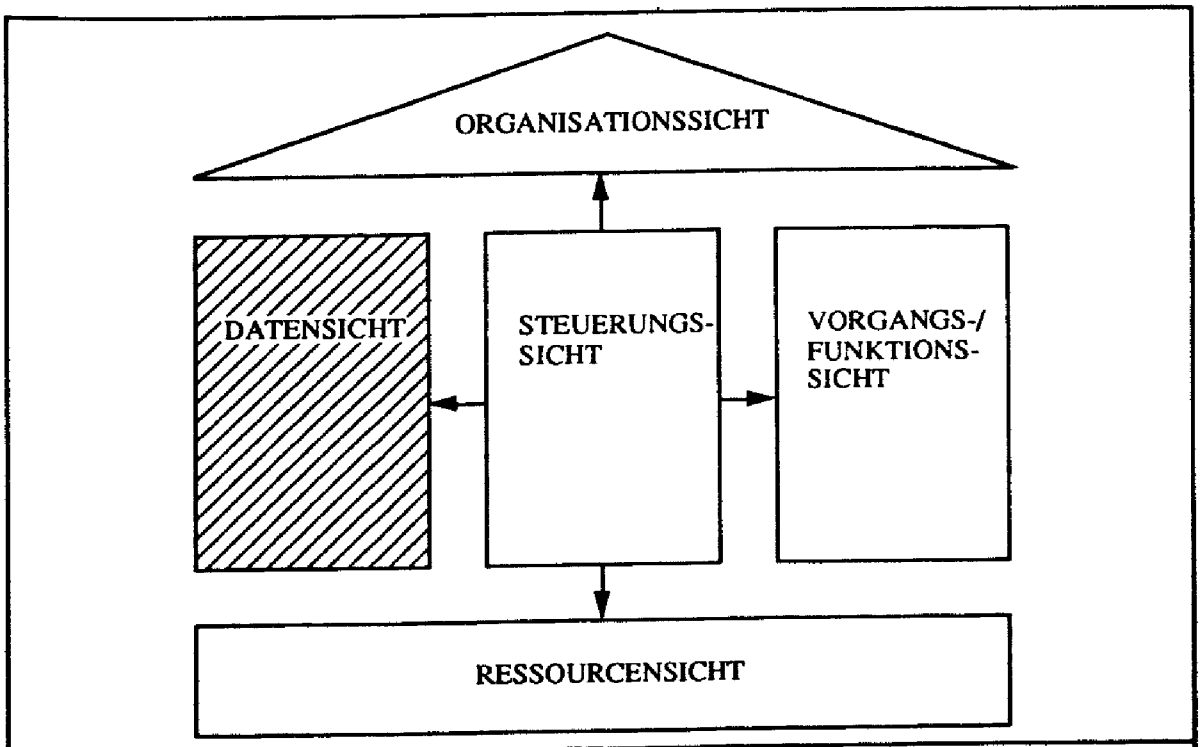


Abb. 3.2.1/4: Sichten eines Informationsmodells (in Anlehnung an Scheer (1991))

Eine umfassende Auflistung denkbarer Sichten auf das Original liefert Zachman (vgl. derselbe (1987)). Diese Sichten drücken bestimmte Fragen an das Original aus und münden in Modellierungs- und Abbildungsformen des zu entwerfenden Informationssystems.

Sicht	Datensicht	Kommunikationssicht	Funktionssicht	Personen	Zeitliche	Ziel
Frage	WOMIT wird gearbeitet?	WO wird gearbeitet?	WIE wird gearbeitet?	WER arbeitet woran?	WANN wird gearbeitet?	WARUM wird gearbeitet?
Ausrichtung	Struktur	Fluß	Verarbeitung	Aktoren		
Beschreibungsform	Objekt - Beziehung Attribute	Netzwerk	Input - Prozess - Output	Bericht	-Ereignis -Ergebnis -Ereignis	Ziel-Mittel-Beziehung
Analogie in industrieller Produktion	Stückliste	Arbeitsplatz	Arbeitsplan	Organisationsplan	Belegungsplanung	Arbeitsergebnis

Abb. 3.2.1/5: Sichten eines Informationssystems (vgl. Zachman (1987))

Zachman hebt die Datensicht, die Kommunikationssicht und die Funktionssicht als die für den Entwurf von Informationssystemen bedeutendsten Modellierungsgegenstände heraus.

In jeder dieser Sichten wird danach differenziert, wer diese Sichtweise verfolgt.

In der übergreifenden Unternehmensebene wird von der Unternehmensführung (z. B. Vorstand) der generelle Rahmen eines Informationssystems und sein Zusammenhang mit dem Objekt- und Nutzersystem beschrieben.

Auf der Fachebene hat der verantwortliche Manager eine konkrete Sicht von den zu beantwortenden Fragen: Standorte, Organisationseinheiten, geschäftliche Vorgänge sind die Kategorien seiner Sicht, die er in der jeweils spezifischen Fachsprache (z. B. der Produktion, des Rechnungswesens) beschreibt.

Der Systemdesigner betrachtet demgegenüber abstrakte logische Einheiten: Verteilte Software-Systeme, Programmsysteme und Datenbestände und verwendet die entsprechende DV-Fachsprache.

Der Systementwickler schließlich hat ein technisches Verständnis: Netze, Terminals, Programmzeilen und Datensätze auf Platteneinheiten.

Bei jeder dieser Sichten sollte eine angepaßte Beschreibungstechnik verwendet werden.

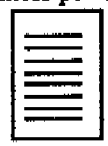
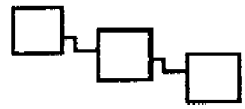
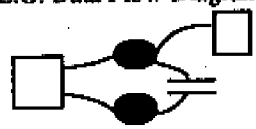
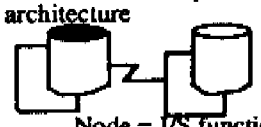
	Data description Entity relation	**Process description** Process Input / Output	**Network description** Node Line
Scope description (Ballpark view)	List of entities important to the business  Entity = class of business entity	List of processes the business performs  Process = class of business process	List of locations in which the business operates  Node = Business location
Model of the business (Owner`s view)	E.G. Entity /Relationship Diagramm  Entity = Business entity Relation=business rule	E.G. Functional Flow Diagramm  Process = Business process I/O = business resources	E.G. Logistic Network  Node = Business Uni Line = Business Relationship Flow
Model of the Information System (Designer`s view)	E.G. Data Model  Entity = Data Entity Relation = Data Relationship	E.G. Data Flow Diagramm  Process = application function I/O = User views (Set of data elements)	E.G. distributed systems architecture  Node = I/S function (processor, storage, access, etc.) Line = Line characteristics
Technology model (Builder`s view)	E.G. Data Design  Entity = Segment/Row Relation = Pointer/Key	E.G. Structure Chart  Process=Computer function I/O=Screen/device formats	E.G. System architecture  Node = Hardware/ System Software Line = Line specifications
Detailed description (Out-of-context view)	E.G. Data Base description  Entity = Fields Relation = Addresses	E.G. Program  Process = language statements I/O = Control blocks	E.G. Network architecture  Node = Addresses Line = Protocols
ACTUAL SYSTEM	DATA	FUNCTION	COMMUNICATIONS

Abb. 3.2.1/6: System-Architektur (vgl. Zachman (1987), S. 285)

Viele Probleme bei der Entwicklung von DV-Systemen resultieren daraus, daß die Sichtweisen der am Informationssystementwurf beteiligten Personen nicht eindeutig unterschieden werden.

Die datenorientierte Systementwicklung räumt dem Datenmodell die Priorität ein, verkennt aber nicht die wichtigen und strukturell prägenden Beziehungen zwischen den drei Modellen, so daß speziell auf der logischen und technologischen Ebene eine simultane Betrachtung der drei Modelle notwendig ist.

Alle drei Modelle existieren in einer statischen und in einer dynamischen Sicht.

3.2.1.3. Ebenen von Informationssystemen

Neben der Unterscheidung von Sichten bei der Modellierung von Informationssystemen lassen sich diese in 4 Ebenen einteilen:

1. die technologische Ebene, auf der die Hardware- und Software-Aspekte betrachtet werden,
2. die funktionale Ebene, die sich damit befaßt, ob die Informationssysteme die zur Aufgabenerfüllung richtigen Prozesse abbilden und über sinnvolle Datenbestände und Kommunikationsprozeduren verfügen,
3. die ökonomische Ebene, die speziell die betriebswirtschaftlichen Erfordernisse der Informationssysteme angesichts der Unternehmensziele verfolgt,
4. die organisatorisch-soziale Sicht, die die Auswirkungen auf die Aufbau- und Ablauforganisation sowie die Nutzer des Informationssystems betrachtet (vgl. Bubenko 1980), S. 395))

Die Gesamtheit aller Informationssysteme einer Unternehmung, die nach den drei Sichten und den 4 Ebenen systematisiert worden sind, sei in Anlehnung an die Literatur als Informationssystem-Architektur bezeichnet (vgl. für viele Scheer (1991)).

Teilweise wird der Begriff „Informationssystem-Architektur" in der Literatur auch enger gefaßt und auf die DV-Technologie beschränkt. Wir verwenden im folgenden die weite Begriffsfassung.

Danach besteht die Systemarchitektur auf allen 4 Ebenen aus den 3 Sichten DATEN - KOMMUNIKATION und PROZESS (vgl. Zachman (1987)).

INFORMATIONSSYSTEM-ARCHITEKTUR				
		Daten	**Kommunikation**	**Prozeß**
Technologische Ebene	Hardware	Speicher-medien	Rechnernetze	DV-Zentraleinheit
	Software	Datenbank-systeme	Netzwerkprotokolle	Anwendungs-programme
Funktionale Ebene		Daten-modell	Kommunikations-modell	Vorgangs-modell
Geschäftliche Ebene	Effektivität Effizienz	Erforderliche Grund-, Bestands-, Strukturobjekte	Kommunikations-partner, -inhalte, -strukturen, -frequenz	Geschäftsvor-gangstypen
Organisatorische Ebene	Aufbauor-ganisation	Datenad-ministrator	Netzwerkadmini-strator	
	Ablauf-organisa-tion	Datenpflege und Datenschutz	Kommunikations-abläufe und -rechte	Arbeitsab-läufe
	Benutzer	Informations-bedarf	Zeitpunkt und Form der Informationsbe-reitstellung	

Abb. 3.2.1/7: Ebenen und Sichten einer Informationssystem-Architektur

Danach hat letztlich jede Unternehmung eine (wie auch immer geartete) Informations-system-Architektur. Diese kann historisch gewachsen oder in Teilen neu zu gestalten sein.

Eine Informationssystem-Architektur unterscheidet sich von einer historisch-gewachsenen Systemlandschaft unter anderem durch folgende Punkte (vgl. Lockemann/Dittrich (1987)):

- die Komponenten sind arbeitsteilig ausgelegt,
- die Komponenten können weitgehend isoliert auf ihre Funktionsfähigkeit geprüft werden,
- die Komponenten lassen sich einfach gegen andere Module austauschen.

Die Modularisierung der Systemarchitektur betrifft alle vier Ebenen. Sie sichert die notwendige Flexibilität gegenüber technologischen und wirtschaftlichen Entwicklungen, die Erweiterbarkeit entsprechend geschäftspolitischer Entwicklungen und die Betriebssicherheit des Gesamtsystems durch eindeutige Schnittstellen.

3.2.2. Gestaltungsregeln für Informationssysteme

3.2.2.1. Integration der Informationssysteme

Unter Integration von Informationssystemen soll die ganzheitliche Abstimmung aller Komponenten verstanden werden, um die negativen Auswirkungen von arbeitsteiliger und dezentraler Auslegung von Systemkomponenten zu vermeiden (vgl. Hax /Majluf, S.73 ff).

Die Integration ist nach verschiedenen Kriterien zu unterscheiden.

Kriterium	Arten	Erläuterung
Integrationsrichtung	horizontale	entsprechend dem Leistungs- und Wertschöpfungsflusses
	vertikale	entsprechend der Hierarchie betrieblicher Anwendungssysteme
	zeitliche	entsprechend dem Plan-Ist-Regelkreis
Integrationsreichweite	funktionsintern	eine Unternehmensfunktion verfassend
	unternehmensintern	eine Unternehmung umfassend
	unternehmensübergreifend	eine gesamte Wertschöpfungskette umfassend
Integrationsintensität	Mengenintegration	Parallele und zeitlich synchrone Erfassung aller Komponenten des Mengenstroms
	Wertintegration	Parallele und zeitlich synchrone Erfassung aller Komponenten des Wertestroms
	Mengen-/Wertintegration	Parallele und zeitlich synchrone Erfassung des Mengen- und Wertestroms
	Operatoren/Objektintegration	Parallele und zeitlich synchrone Erfassung der mengen- und wertverbrauchenden Aktivitäten an Objekten (z. B. Produkten) durch Operatoren (Maschinen)

Abb. 3.2.2/1: Kriterien der Integration

Die Integration orientiert sich auf allen vier Ebenen am möglichst ungehinderten und effizienten Fluß der Informationen (vgl. Mertens u.a. (1991), S. 45).

Technologieintegration zielt darauf ab, daß die Informationen zwischen den verschiedenen Datenverarbeitungsanlagen innerhalb einer Unternehmung unabhängig von deren Standort und technischer Auslegung ohne manuelle Eingriffe weitergeleitet und weiterverarbeitet werden können. Im engen DV-technischen Sinne spricht man auch von Systemkopplung.

Funktionsintegration bedeutet, daß die zur Abwicklung eines Geschäftsprozesses notwendigen Informationen nur einmal erfaßt und gespeichert werden (Datenintegration) und damit alle Vorgänge des Geschäftsprozesses so gesteuert werden, daß sie reibungslos ineinander greifen (Vorgangsintegration).

Ökonomische Integration heißt in strategischer Hinsicht, daß die Informationssysteme folgerichtig aus den strategischen Erfolgsfaktoren des Unternehmens abgeleitet wurden und gesamthaft alle Steuerungserfordernisse des Managements abdecken (Effektivität der Informationssysteme). In operativer Hinsicht sollen die Informationssysteme effizient die technologischen und organisatorischen Möglichkeiten der Datenverarbeitung, -speicherung und -kommunikation nutzen, um die notwendigen Funktionen möglichst wirtschaftlich zu erfüllen (Effizienz der Informationssysteme).

Organisatorische Integration bedeutet, daß die Informationssysteme integraler Bestandteil der Aufbau- und Ablauforganisation der Unternehmung sind. Weder können Informationssysteme unabhängig von den Organisationsstrukturen im Unternehmen eingeführt werden noch umgekehrt. Zu der Organisationsintegration gehört auch die Einbeziehung der Fertigkeiten und Fähigkeiten der Benutzer bei der Gestaltung von Komponenten der Informationssysteme.

Organisatorische Integration bezüglich der Ablauforganisation bedeutet die Abstimmung der betrieblichen Abläufe auf die Planungs- und Steuerungskapazitäten der Informationssysteme. Dies führt unter anderem zu einer teilweisen Aufhebung der bisherigen Arbeitsteilung, zu einer Straffung und kontinuierlichen Gestaltung von Ablaufketten und zu deren Synchronisation.

Hinsichtlich der Aufbauorganisation führt die organisatorische Integration von Informationssystemen zu einer Abflachung von Managementhierarchien, da Anwendungssysteme die Managementebenen bei ihrer Aufgabenerfüllung unterstützen.

Ziele der Integration sind letztlich die Erhöhung der Produktivität und Wirtschaftlichkeit der Aktivitäten der Unternehmung. Primär reduziert die Integration die Anzahl und die Zeiten der Abläufe in der Organisation und speziell im Informationssystem der Unternehmung.

Produktivitätsziele	**Beispiele**	**Wirtschaftliche Auswirkungen**
Zeitziele	- Reduzierung von Durchlaufzeiten - Senkung von Entwicklungszeiten	- Vermeidung von Kapitalbindung in Zwischenlägern - Umsatzwirkung durch schnellere Markteinführung
Redundanzziele	- Vermeidung von doppelten Datenbeständen - Vermeidung von Mehrfacheingaben und Medienbrüchen	- Senkung von DV-Kosten - Senkung von administrativen Gemeinkosten

Abb. 3.2.2/2: Zielwirkungen der Integration

Prinzipiell existieren zwei Mittel zur Integration von Informationssystemen: Bei der Datenintegration greifen die verschiedenen Teilsysteme auf eine einheitliche Datenbasis zu, bei der Prozeßintegration übergeben die verschiedenen Teilsysteme nach einer bestimmten Steuerungslogik einander die Daten über genormte Schnittstellen.

	Formen	**Erläuterung**
Datenintegration	konzeptuelle	ein konzeptuell einheitliches Datenmodell
	physische	eine einheitliche Datenbank
Prozeßintegration	konzeptuell integriert, physisch heterogen	ein konzeptuell einheitliches Schnittstellenmodell, jedoch zwischen den Programmen heterogene Schnittstellenprogramme
	konzeptuell integriert, physisch integriert	ein physisch einheitliches Schnittstellenprogramm
	konvertergestützt integriert	ein Konvertersystem definiert und sichert das Schnittstellen-Austauschformat

Abb. 3.2.2/3: Mittel der Integration

Bei der Anwendung der Datenintegration ist zu unterscheiden

-> ein (physisch) einheitlicher Datenbestand
-> ein konzeptuell einheitliches Datenmodell

Beide Formen der vertikalen Datenintegration haben spezifische Vor- und Nachteile:

	Physisch integrierter Datenbestand	Konzeptuell integriertes Datenmodell
Vorteile	- Datenintegrität jederzeit gegeben - Selektions-/Aggregationsvorgänge bedarfsgerecht	- gutes Zugriffsverhalten durch vor-aggregierte/ vorselektierte Daten-bestände
Nachteile	- Aggregations-/Selektionsvor-gänge rechenzeitaufwendig, d. h. - schlechtes Zugriffsverhalten	- u. U. erhebliche Datenredundanz mit entsprechendem Speicher-platzbedarf - Datenintegrität nur zu vordefinier-ten Zeitpunkten oder nach be-stimmten Ereignissen

Abb. 3.2.2/4: Formen der Datenintegration

3.2.2.1.1. Vertikale Integration

Die vertikale Integration von Informationssystemen kann wiederum auf den verschiedenen Ebenen betrachtet werden. Technologisch ist das Ziel die Verbindung von Rechnern unterschiedlicher Leistungsklassen zu einer Rechnerhierarchie und die Verknüpfung von deren Softwaresystemen.

Unter vertikaler Funktionsintegration soll die hierarchische Verbindung zwischen den vier Ebenen von Anwendungssystemen (Administrations-, Dispositions-, Informations- und Planungssystemen) verstanden werden, um die Entscheidungen und den Informationsstand der verschiedenen Managementebenen miteinander zu koppeln. Diese Koppelung kann entsprechend der methodischen Ansätze zur Planungskoordination
-> einseitig in Form von Informationsflüssen von unten nach oben/oben nach unten oder
-> beidseitig in Form von (vermaschten) Regelkreisen erfolgen.

Ziel ist die Integration der Planungs- und Kontrollprozesse mit der Organisationsstruktur der Unternehmung, um
- entsprechend der Managementebene angepaßte Planungs- und Kontrollgrößen (Steuerungsgrößen) zu definieren,
- für jede Steuerungsgröße eindeutig Verantwortliche für die Zielerreichung durch zu ergreifende Maßnahmen zu benennen,
- letztlich die Ziele der strategischen und operativen Unternehmensplanung durch die Einbindung der Informationssysteme in die Führungsprozesse besser zu erreichen.

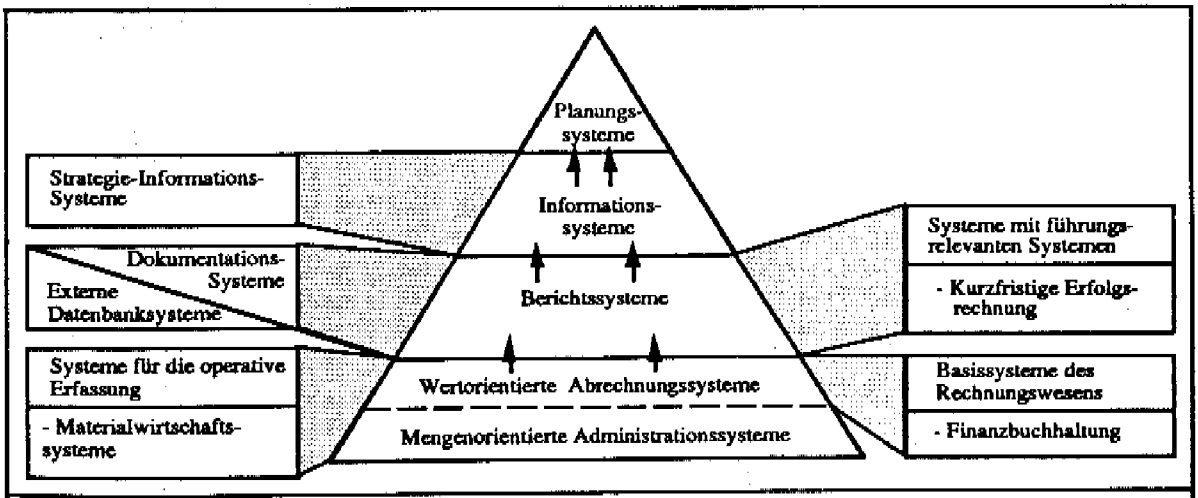


Abb. 3.2.2/5: Vertikale Integration von Anwendungssystemen (vgl. Scheer (1990))

Als Grundgerüst der Systempyramide dienen heute die traditionellen, DV-basierten Rechnungswesen- und Mengenabrechnungssysteme. Diese müssen so ergänzt werden, daß sie den Planungs- und Kontrollprozeß in operativer und strategischer Hinsicht unterstützen. Berichtssysteme verdichten die Daten des Rechnungswesens zu führungsrelevanten Informationen und ergänzen diese um externe Recherchen (z. B. für Markt- und Konkurrenzanalysen). Für die oberen Managementebenen wird versucht, strategisch relevante Informationen zu gewinnen und in die Systeme einzuspeisen.

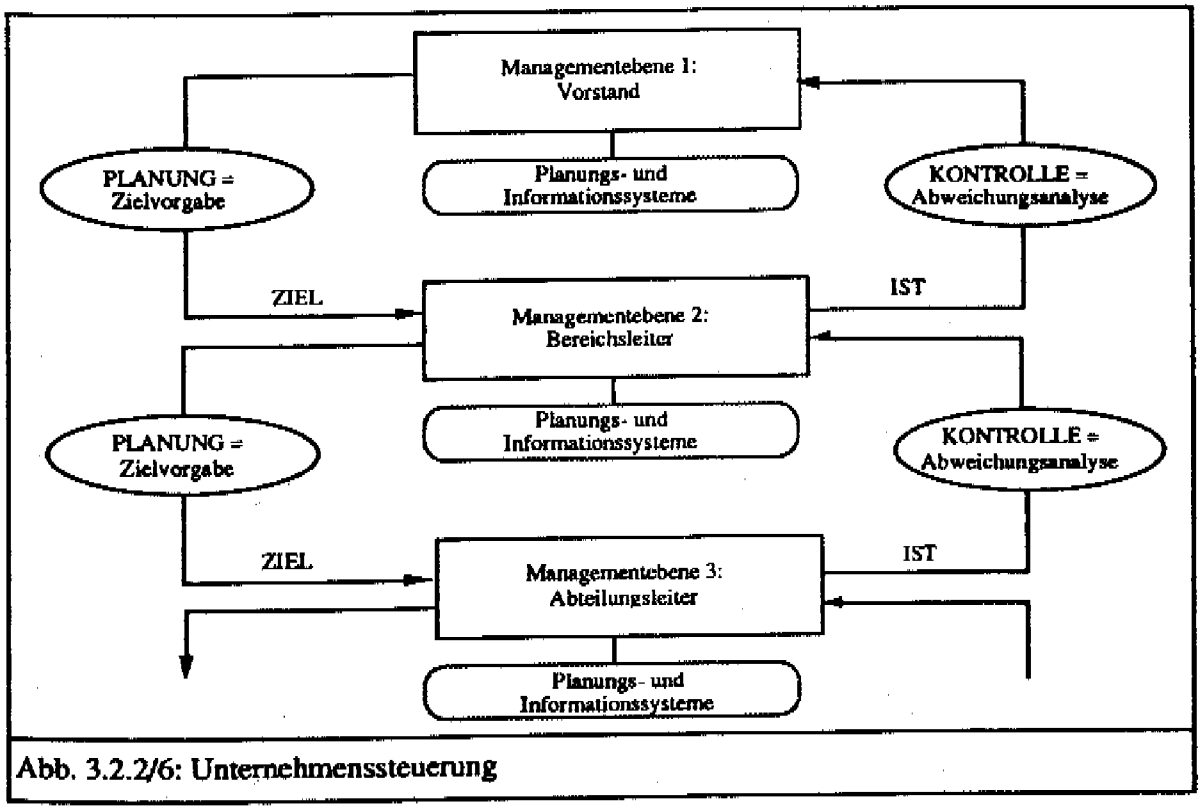


Abb. 3.2.2/6: Unternehmenssteuerung

Jede Managementebene verfügt in der Systempyramide über spezifische Planungs- und Informationssystemkomponenten.

Die vertikale Integration kann sowohl über die Verknüpfung von Vorgängen als auch über die Datenintegration erfolgen. Bei der Verknüpfung von Vorgängen werden die Programmsysteme innerhalb der Systempyramide zeit- oder ereignisgesteuert miteinander gekoppelt und übergeben über Schnittstellen den Folgeprogrammen die notwendigen Daten.

Bei der Datenintegration kann ein physisch integrierter Datenbestand existieren, z. B. in Form einer einheitlichen Datenbank, auf den alle Systeme der Pyramide zugreifen. In einem anderen Form existiert zwar ein konzeptuell integriertes Datenmodell, d. h. alle Anwendungssysteme verwalten ihre Daten in logisch abgestimmter Form, doch halten zumindest einige der Systeme physisch ihren eigenen Datenbestand. Nach bestimmten Ereignissen oder Zeiten übergeben die Systeme innerhalb der Pyramide einander die Daten, dann besteht für einen Moment ein integrierter Datenbestand.

3.2.2.1.2. Horizontale Integration

Im Rahmen der horizontalen Integration wird versucht, die Mengen- und Wertflüsse im Wertschöpfungsprozeß gesamthaft und durchgängig in Informationssystemen zu erfassen.

Ziele der horizontalen Integration sind

-> die zeitliche Straffung von Vorgangsketten (z. B. von Durchlaufzeiten in der Fertigung oder von Entwicklungszeiten),
-> die Synchronisation von Vorgängen und die Erhöhung der Gesamtproduktivität (z. B. durch Verringerung von Pufferlager in der Fertigung).

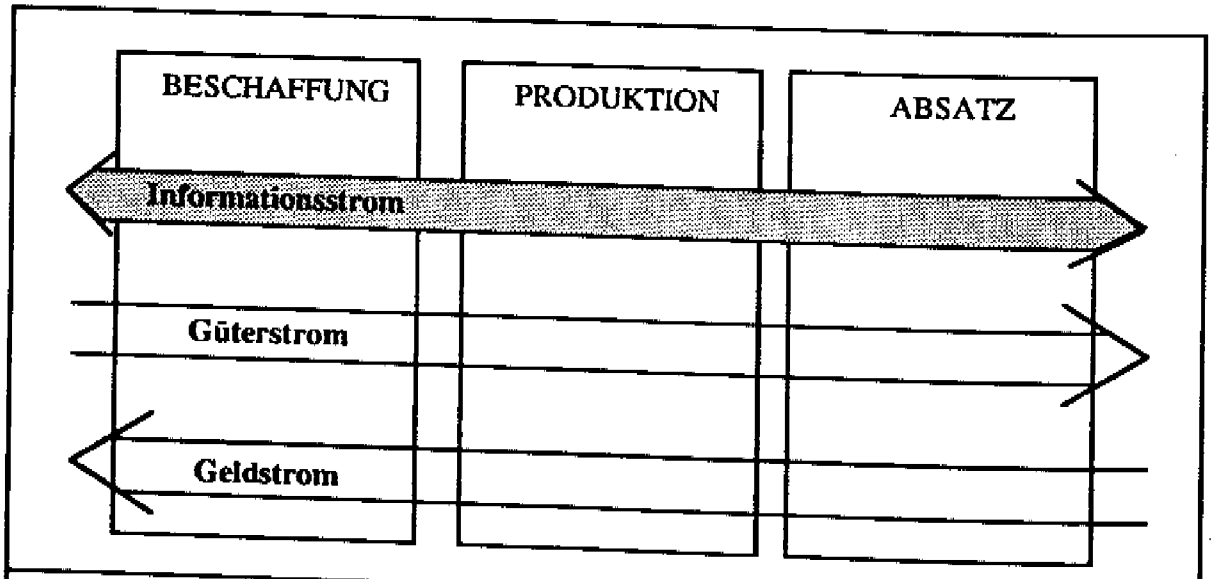


Abb. 3.2.2./7: Informations-, Güter- und Geldstrom

Beispielsweise sollen alle zur Abwicklung eines Kundenauftrages notwendigen Vorgänge im Leistungs- und Wertschöpfungsfluß in den Informationssystemen erfaßt und durch diese geregelt werden. Typische horizontale Wertschöpfungsketten sind

- die Auftragsbearbeitung vom Angebot über den Versand bis zum Zahlungseingang,
- der Produktlebenszyklus von der Produktidee über die Produktentwicklung bis zur Garantieleistung,
- der interne Materialfluß vom Wareneingang über die Einlagerung bis hin zum Verbrauch in der Produktion (vgl. Scholz-Reiter (1990),S.22).

Die horizontale Verknüpfung kann zeitlich vorwärts schreitend den Material- und Wertefluß durch Informationen begleiten oder von Planungsinformationen ausgehend rückwärts schreitend Maßnahmen des Mengen- und Werteflusses auslösen.

	Denkansatz	**Vorgangskette (Ausschnitt)**
vorwärts schreitend	reale Prozeßbegleitung	Materialbeschaffung Wareneingang Lagerentnahme Bearbeitung Stufe 1 ... Auslieferung Kundenübergabe Kundenzahlung
rückwärts schreitend	virtuelle Prozeß-Vorwegnahme	Kundenauftrag Produktionsplanung Materialbedarfsplanung Beschaffungsplanung

Abb. 3.2.2/8: Formen der horizontalen Integration

Zur vollständigen Abbildung des Mengen- und Werteflusses sind die Operationen an sämtlichen Fertigungsobjekten in sämtlichen Fertigungsstationen zu erfassen. Man spricht von der parallelen Erfassung von

-> Operatorenoperationen
-> Objektoperationen,

wenn die Informationssysteme jede Mengen- und jede Wertbewegung sowohl einmal dem Fertigungsobjekt zurechnen als auch der Fertigungsstation (= Operator).

	Objekt	**Operator**
Mengenfluß	- Produkt - Fertigungsauftrag	- Maschine - Lagerposition - Verladestation
Wertefluß	- Kostenträger	- Kostenstelle

Abb. 3.2.2/9: Objekte und Operatoren im Mengen- und Wertefluß

Die horizontale Integration kann entweder über Trigger-Konzepte oder mit Hilfe der Datenintegration erfolgen.

Trigger-Konzepte arbeiten mit den Start- und Ende-Ereignissen jedes Vorganges. Diese lösen bei parallelen oder nachgelagerten Vorgängen wiederum Ereignisse aus.
Die Ereignisse können Objekte oder Beziehungen zwischen Objekten betreffen und diese entweder verändern oder in ihrer Existenz treffen.

	Objekt	Beziehung zwischen Objekten
Existenz	- Aufnahme eines neuen Kunden - Löschung eines Produktes	- Aufnahme einer neuen Bestellung eines Kunden für Produkte
Veränderung	- Veränderung der Preiskonditionen	- Veränderung einer Bestellmenge

Abb. 3.2.2/10: Beispiele für von einem Vorgang ausgelöste Ereignisse

In Anlehnung an Scheer (1991) lassen sich Vorgänge des Güterstroms von solchen des Informationsstroms unterscheiden.

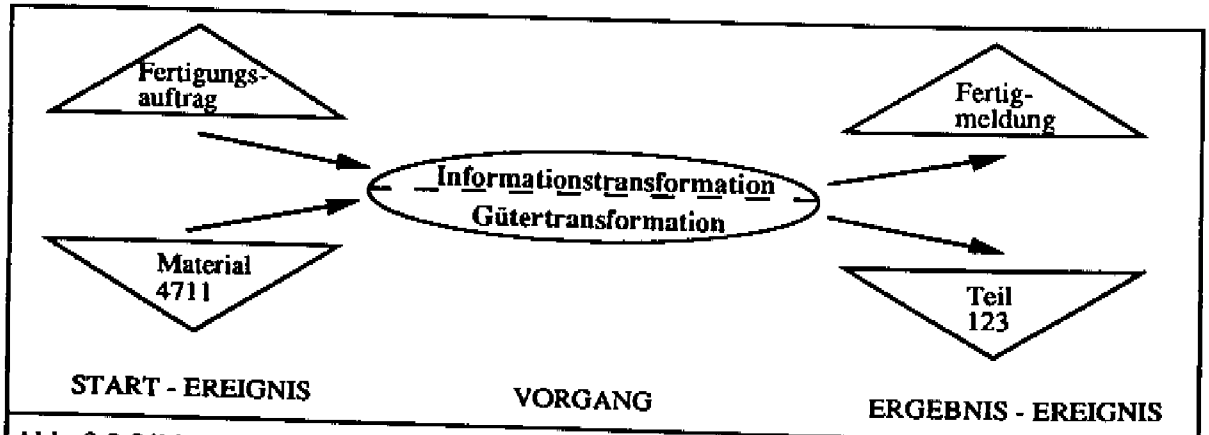


Abb. 3.2.2/11: Vorgänge des Güter- und Informationsstroms (vgl. Scheer (1991))

Daten-Integrationskonzepte arbeiten im Unterschied zu Trigger-Konzepten nicht ereignisorientiert, sondern auf der Basis eines gemeinsamen Datenbestandes.

Durch die Struktur und Attribute des Datenbestandes erkennen die in der Wertschöpfungskette integrierten Anwendungssysteme den Bearbeitungsstand.

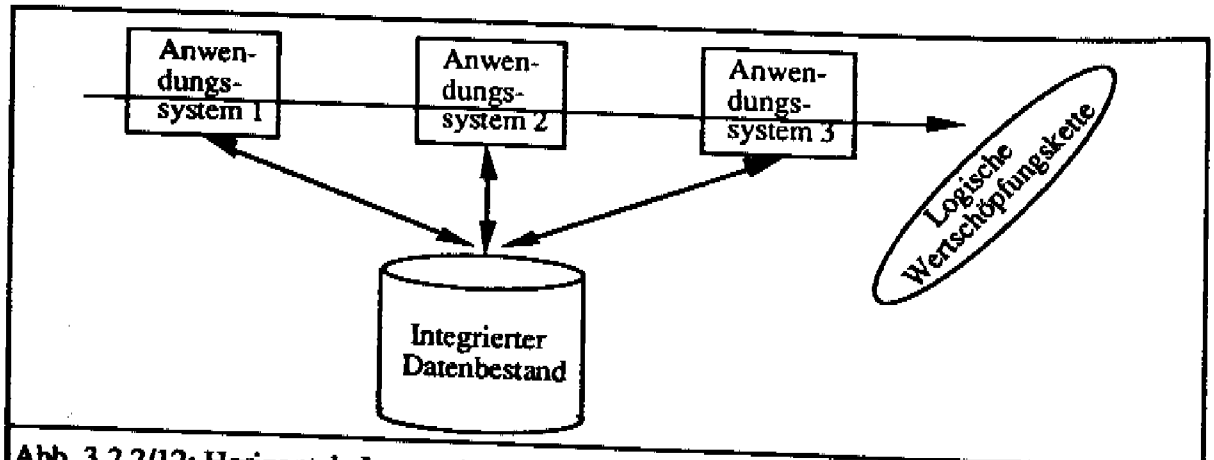


Abb. 3.2.2/12: Horizontale Integration auf Basis eines integrierten Datenbestandes

Bei reiner Datenintegration erfolgt die gesamte Steuerung des DV-Prozesses durch den Datenbestand; üblicherweise übernehmen jedoch spezielle Steuerungskomponenten die Gesamtablaufsteuerung.

Der Prozeß der Informationstransformation existiert dabei zum einen als Abbild des Gütertransformationsprozesses und wird dann als Bestandteil des Prozesses der Güterlogistik begriffen; zum anderen dient er der geistigen Prozeßvorwegnahme im Rahmen der Planung. Wir sprechen dann von Informationslogistik.

3.2.2.1.3. Zeitliche Integration

Unter zeitlicher Integration soll zum einen die Verknüpfung von Planungs-, Prognose- und Kontrollinformationen einer Periode und zum anderen der Vergleich von Informationen verschiedener Perioden verstanden werden.

Planungs-, Prognose- und Kontrollinformationen finden sich vor allem in den Anwendungssystemen mit controllingrelevanten Informationen, d. h. in Systemen, die die verschiedenen Managementebenen bei der Führung ihrer Organisationsbereiche unterstützen.

Die zeitliche Integration fordert von den DV-Systemen sowohl periodenübergreifend vergleichbare Auswertungsmethoden als auch Auswertungsstrukturen. So wäre es sinnlos, wenn die Planungsinformationen nach einer anderen Methode oder nach anderen Verdichtungsstrukturen ermittelt werden als die Kontrollinformationen.

Art	Erläuterung	Anforderungen
Zeitliche Methodenintegration	Das DV-System muß in der Lage sein, die Daten verschiedener Perioden mit einem flexibel bestimmbaren Verfahren auszuwerten.	Methodengenerationen
Zeitliche Strukturintegration	Zeitliche Strukturintegration bedeutet, daß das DV-System in der Lage sein muß, die Daten verschiedener Perioden mit einer flexibel bestimmbaren Struktur auszuwerten.	Strukturgeneration

Abb. 3.2.2/13: Anforderungen der zeitlichen Integration

Beispiel:
Ein Hersteller von Erfrischungsgetränken sollte sowohl in der Planung als auch im Ist die Absatzzahlen entweder in Litern oder in Verpackungseinheiten messen (Methodenintegration). Wird eine neue Verpackungsform eingeführt oder eine alte aus dem Absatzprogramm gestrichen, sollte das Absatzberichtssystem demnach für verschiedene Perioden vergleichbare Zahlen ermitteln können (Strukturintegration).

3.2.2.2. Verteilung von Informationssystemen

Die technologische Entwicklung bei der Hardware und der Software bietet zunehmende Möglichkeiten, wirtschaftlich verteilte DV-Lösungen zu realisieren. Im Großunternehmen sind heute teilweise schon über 70 % der Arbeitsplätze mit DV-Technologie ausgestattet. Ziel ist dabei die Steigerung der Leistungsfähigkeit und der Zugänglichkeit von Datenbeständen und Anwendungssystemen.

Beim Entwurf eines verteilten Systems sind folgende Fragen zu beantworten (vgl. Wedekind (1988)):

- Was wird verteilt? (Gegenstand der Verteilung)
- Wohin wird verteilt? (Ort der Verteilung)

- Wann wird verteilt? (Zeitpunkt der Verteilung)
- Warum wird verteilt? (Ziel der Verteilung)

Kriterium	Art	Erläuterung
Verteilungsgegenstand	Daten	Verteilung oder Verfügbarkeit von Datenbeständen an dezentralen DV-Anlagen
	Funktionen	Verteilung oder Verfügbarkeit von Programmsystemen
Verteilungsort	homogene Knoten	Rechnerknoten identischer Prozessorfamilien und Basissoftware
	heterogene Knoten	Rechnerknoten unterschiedlicher Prozessorfamilien und Basissoftware
Verteilungszeit	statisch-vorverteilt	statische Verteilung bei Installation des Systems
	dynamisch-betriebsverteilt	dynamische Verteilung während des Systembetriebs

Abb.3.2.2/14: Kriterien der Verteilung

Verteilt werden können Datenbestände oder Funktionen. Gesichtspunkte sind dabei zum einen die in jeder Organisationseinheit vorzufindenden Anwendungssituationen, zum anderen die technologische und wirtschaftliche Zuteilbarkeit von DV-Ressourcen auf die Organisationseinheiten.

Es ist dabei die physische Verteilung von der logischen Verfügbarkeit bestimmter Daten oder Programme zu unterscheiden. Es ist zwar logisch unbegrenzt möglich, allen Nutzern alle Programme und Daten zugänglich zu machen, eine physische Verteilung würde jedoch Kopien auf jedem Knoten und damit erhebliche Kapazitäten des Informationssystems voraussetzen.

Ziele der Verteilung können in der Steigerung der organisatorischen Leistungsfähigkeit liegen, indem Datenbestände und Programmsystem am Arbeitsplatz bereitgestellt werden Weiterhin soll dem Anwender die Möglichkeit eröffnet werden, seine eigenen DV-Lösungen auf seiner "Workstation" zu realisieren. Andere Ziele liegen auf der technischen Ebene, in dem die technische Leistung, Erweiterbarkeit oder Zuverlässigkeit (durch Redundanz der Systeme) des Informationssystems gesteigert werden. Wirtschaftliche Ziele betrachten die (häufig geringeren) Investitionskosten verteilter Systeme und deren Betriebskosten (speziell auch für die Kommunikation).

3.2.2.3. Verbreiterung des Informationsangebots

Traditionell basieren die DV-Systeme auf den Abrechnungssystemen des Rechnungswesens. Diese Systeme nutzen die Möglichkeiten moderner Informationsverarbeitungstechnologie nur selten voll aus.

Die erste Generation der Anwendungssysteme nutzte die Möglichkeiten der DV-Technologie allein zur quantitativen Leistungssteigerung der Informationsinhalte. Diese Systeme führten zu einer Informationsüberflutung der Managementebenen; es wurden zu viele und zu detaillierte Daten produziert, die häufig jedoch nicht den Anforderungen des Managements entsprachen. Als Reaktion auf diesen Mangel wurden in einer zweiten Generation die Formate der Berichte durch verbesserte Aggregationsverfahren und graphische Auswertungen gesteigert.

Leistungssteigerung	Beispiele	DV-Anforderungen
Berichtsquantität	- Mehr Berichtsinhalte - Umfassendere Berichtszeiträume - Häufigere Berichtsfrequenzen	- Speichermedien hoher Kapazität
Berichtskomfort	- Flexible Berichtsformen - Graphische Auswertungen	- Aggregationsverfahren - Arbeitsplatzstationen mit graphischen Fähigkeiten
Strukturflexibilität	- Anpassung an Veränderungen der Organisation und der Märkte	- Flexible Selektionsverfahren - Flexible Aggregationsverfahren - Leistungsfähige Datenbanksysteme - Vorverdichtungen der Berichtsinhalte
Informationsflexibilität	- Flexibles Informationsangebot aufgrund einer sich veränderten Informationsnachfrage	- Flexible Nutzung externer Datenbanken und Kommunikationssysteme - Einbindung der organisationsweiten Groupware-Konzepte
Berichtsqualität	- Erweiterung der Berichtsinhalte um qualitative Daten und Einschätzungen	- Datenbankstrukturen für Dokumente und Graphiken

Abb. 3.2.2/15: Erweiterungsfelder von Informationssystemen

Das grundsätzliche Manko dieser Systeme liegt darin, daß sie auf Fragestellungen beschränkt sind, die von vornherein strukturell verankert sind, während das Management zunehmend mit neuartigen Fragestellungen konfrontiert ist.

Beispiel:
Soll die Frage beantwortet werden, wieviel Aufwand ein Konzern in den letzten Jahren für Umweltschutz verwendet hat, müssen die Kostenrechnungssysteme aller Tochtergesellschaften eine entsprechende Kategorie vorhalten oder die Möglichkeit eröffnen, diese zu generieren.

Es ist theoretisch und praktisch nicht möglich, für alle nur denkbaren Informationsbedürfnisse des Managements die Daten zu erheben und für den Tag zu speichern, an dem diese benötigt werden. Daher muß dem Wandel der Informationsbedürfnisse mit flexiblen Informationssystemen begegnet werden, die einen Basis-Datenbestand durch situativ gewonnene Informationen ergänzen können. Ansätze dazu bieten

- der Zugang zu der Vielzahl von externen Datenbanken, die weltweit verfügbar sind,
- die Nutzung der organisationsweiten "Groupware"-Kommunikationsnetze, die den Sachverstand der Mitarbeiter allen schnell und umfassend verfügbar machen,
- die Verwendung flexibler Programmsysteme und komfortabler Arbeitsplatzrechner zur Selektion, Aggregation und Aufbereitung der Informationen.

Informationssysteme sollten Lernprozesse in einer sich stetig wandelnden Unternehmenswelt unterstützen und somit bedarfsorientiert arbeiten und nicht "stur vorgefertigte Datenmenüs" anbieten.

Ein weiterer Nachteil traditioneller Informationssysteme liegt darin, daß sie auf den quantitativen Daten des Rechnungswesens basieren und die Rolle qualitativer Daten und Einschätzungen für das Management zu wenig berücksichtigen. Solche weichen Faktoren werden beispielsweise im sogenannten 7-S-Modell der Unternehmensberatung McKinsey betont. Möglichkeiten dazu bieten weiterentwickelte Meßverfahren in den Sozialwissenschaften (z. B. proxy attributes), die Entwicklung von Verarbeitungsregeln für unscharfe Informationen (z. B. fuzzy set theory) sowie deren Verankerung in wissensbasierten Systemen (vgl. Fischer (1989) und Pohle (1990)).

Beispiel:
Die Existenz eines Unternehmens hängt in hohem Maße vom Personalpotential an risikobereiten Unternehmern, verwaltenden Managern und rational-analytischen Wissenschaftlern ab. Die Personalinformationssysteme sollten somit neben den traditionellen quantitativen Kenngrößen auch entsprechende Personenmerkmale halten, um die Nachwuchsrekrutierung und -planung zu erleichtern.

Qualitative Informationen sind speziell bei der Lösung schlecht-strukturierter und strategischer Problemstellungen des Managements zu erwarten. Strategische Chancen oder Risiken kündigen sich durch "schwache Signale (weak signals)" an, die sich einer Abbildung in den Meßgrößen des Rechnungswesens entziehen.

3.2.2.4. Analyse des Informationsbedarfs

Im Rahmen des strategischen Managements werden die Potentiale herauszufinden versucht, die über den langfristigen wirtschaftlichen Erfolg einer Unternehmung entscheiden. Potentiale sind die Differenzierungsmerkmale, die zeigen, was ein Unternehmen besser kann als andere. Sie kennzeichnen die Fähigkeiten, die es einem Unternehmen ermöglichen, spezifische Probleme für bestimmte Zielgruppen (= Kunden) nachhaltig besser zu lösen als der Wettbewerb. Neben materiellen und personellen Potentialen bilden Informationssysteme die dritte große unternehmerische Kraft.

1. Schritt: Analyse des Markt-Leistungszusammenhangs

Die Analyse des Informationsbedarfs kann ansetzen bei grundlegenden, stark vereinfachten Zusammenhängen zwischen Märkten und Betrieben.

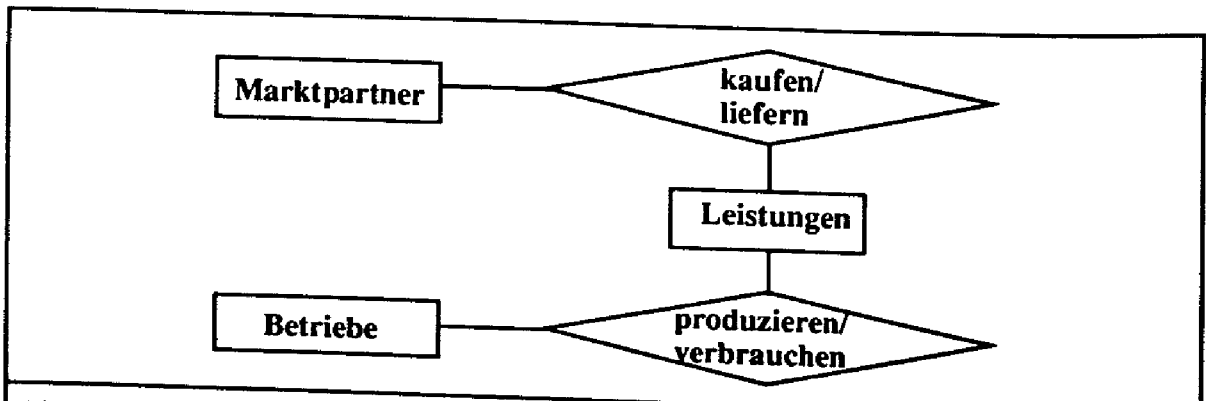


Abb. 3.2.2/16: Zusammenhang zwischen Märkten, Leistungen und Betrieben (vgl. Scheer (1990))

Die Beziehungen zwischen dem Betrieb und dem Marktpartner lassen sich auf den gegenläufigen Leistungs- und Geldfluß sowie den Informationsfluß zurückführen.

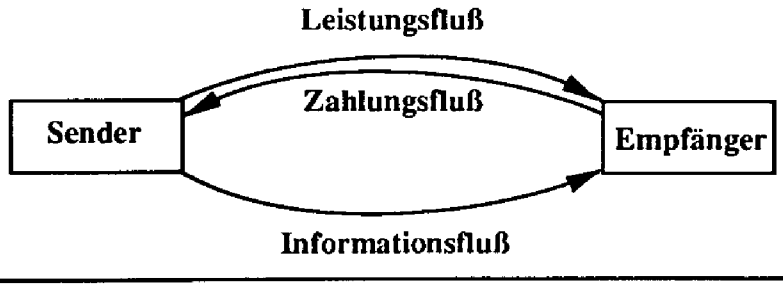


Abb. 3.2.2/17: Leistungs-, Geld- und Informationsfluß

Der Informationsfluß kann dem Leistungs-/Zahlungsfluß vorgeschaltet sein (d. h. ihn initiieren), er kann zeitlich parallel verlaufen oder nachgeschaltet sein. Bestimmte Elemente sind aus technischen, organisatorischen oder juristischen Gründen notwendig, damit der Leistungs- oder der Geldfluß ablaufen oder gesteuert werden kann. Andere Elemente des Informationsflusses begleiten zwar den Leistungs- bzw. den Geldfluß, sie sind aber nicht unbedingt notwendig und lassen sich daher ersatzlos streichen. Diese Elemente seien als hinreichend bezeichnet (vgl. Ferstl/Sinz (1991)).

	vorgeschaltet	**parallel**	**nachgeschaltet**
notwendig	Auftrag	Lieferanweisung Zahlungsanweisung	Empfangsbescheinigung
hinreichend	Anfrage Angebot	Lieferschein	Mahnung

Abb. 3.2.2/18: Systematisierung des Informationsflusses

Ziel der Informationsbedarfsanalyse ist es, die notwendigen und hinreichenden Elemente möglichst vollständig aufzulisten und jeweils zu erfassen, welche Informationskomponenten zur Erfüllung der Aufgabe aus Sicht des Marktes und des internen Leistungsprozesses notwendig sind.

Beispiel:
In einem Autohaus erkundigt sich ein potentieller Kunde nach den Konditionen bei dem Kauf eines Neuwagens (Preis, Rabattmöglichkeiten, Lieferzeit, Gebrauchtwageninzahlungnahme). Bei einer solchen Kundenanfrage fallen kaum notwendige Elemente an, wohl aber eine Reihe hinreichender Elemente.

	Marktsicht	*Leistungssicht*
notwendig		*Kundenadresse*
hinreichend	*Angebotskonditionen Gebrauchtwagen* *- Typ* *- Zustand*	*- Für Reservierung beim Hersteller Ziel-PKW* *- Typ* *- Ausstattungsmerkmale*
operativ sinnvoll	*Kundenmerkmale* *- Alter* *- Beruf*	*Finanzierungsmodelle* *- Leasing* *- Kreditmodelle*
strategisch sinnvoll	*- Denkbare Konkurrenzfabrikate* *- PKW-Typ* *- Ausstattungsmerkmale* *- Kaufbeeinflussende Faktoren*	

Abb. 3.2.2/19: Informationsbedarf bei einer Kundenanfrage in einem Autohaus

Viele Unternehmen werden bei einer solch unverbindlichen, persönlichen Anfrage nicht einmal die Kundenadresse speichern. Für eine erneute Ansprache des Kunden ist es jedoch sinnvoll, neben identifizierenden Merkmalen auch den Gegenstand und die Konditionen des Angebots zu speichern. Aus strategischer Sicht bietet es sich an, den Entscheidungsprozeß des Kunden (Was spricht für unser Angebot? Was spricht dagegen?) möglichst umfassend zu dokumentieren, um daraus Folgerungen für den Einsatz des absatzpolitischen Instrumentariums abzuleiten.

2. Schritt: Identifizierung der kritischen Erfolgsfaktoren

Bei der Ermittlung des Informationsbedarfs wird von den übergreifenden strategischen Unternehmenszielen (Business Goals) ausgegangen. Nach Porter unterscheidet man drei mögliche strategische Zielrichtungen eines Unternehmens: Die Kostenführerschaft, die Differenzierung im Leistungsangebot und die Fokussierung auf bestimmte Ziele (vgl. Porter (1980)).

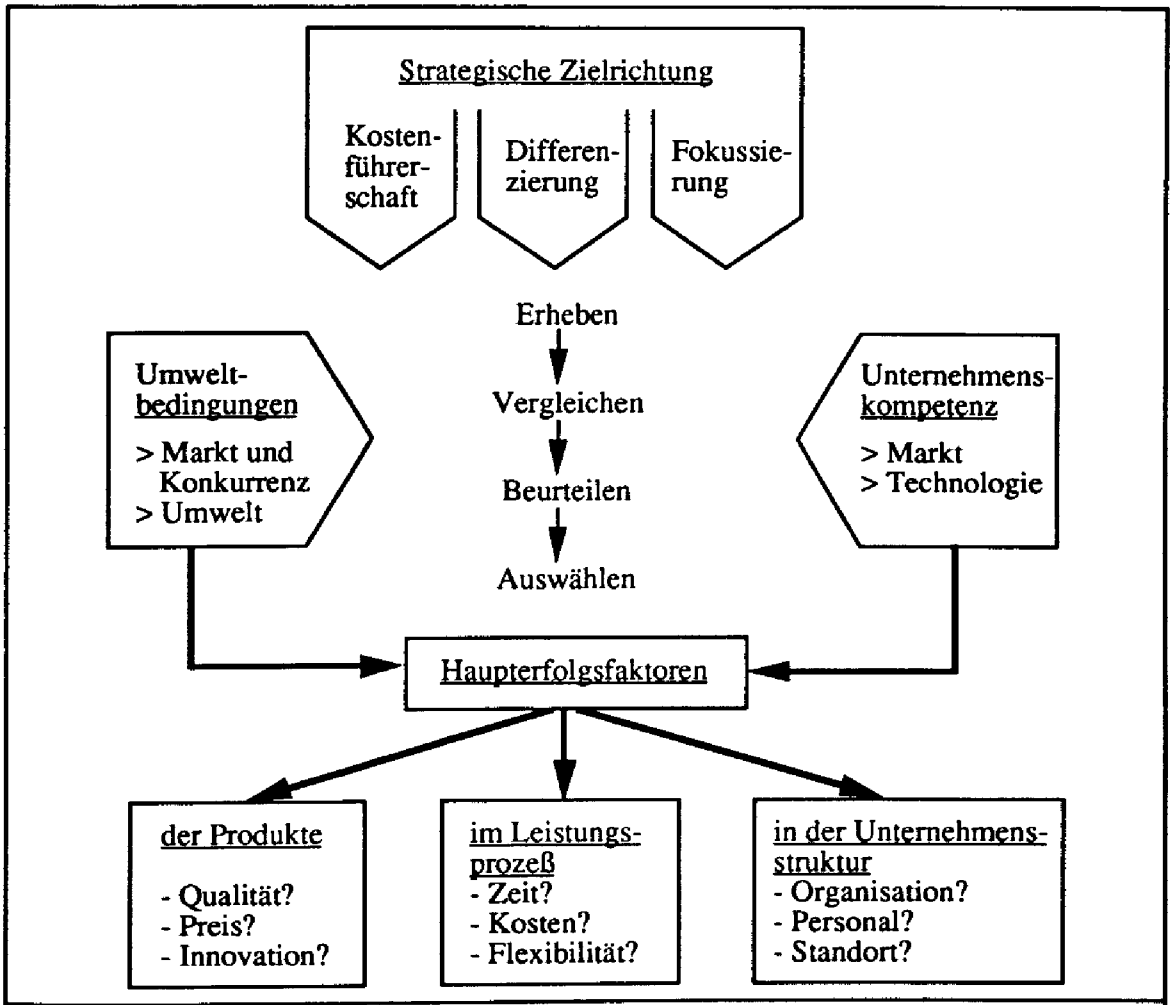


Abb. 3.2.2/20: Grundsätzlicher Ablauf bei der Identifizierung der kritischen Erfolgsfaktoren

Aus den strategischen Zielen werden durch eine detaillierte Analyse des Unternehmensunfeldes und der eigenen Stärken und Schwächen die denkbaren Erfolgsfaktoren der Unternehmung abgeleitet. Beim **analytischen Vorgehen** wird logisch nach Erfolgsfaktoren gesucht, beim statistischen Vorgehen werden empirische Daten über die Unternehmensstrukturen und deren Auswirkung auf die Zielerreichung und in Langzeitstudien ausgewertet. Ein Beispiel dafür ist die PIMS-Datenbank.

Die Critical Sucess Factor Methode (vgl. Rockart (1979), Daniel (1961)) geht davon aus, daß der Erfolg einer Unternehmung im Vergleich zu Wettbewerbern von einer begrenzten Anzahl kritischer Erfolgsfaktoren abhängig ist. Die Methode analysiert sowohl die Branchenstruktur als auch die eigene Erfolgsstrategie der Unternehmung. Ziel ist es, die Haupterfolgsfaktoren der jeweiligen Unternehmung zu spezifizieren, die entweder heute schon existieren oder die von der Konkurrenz noch nicht aufgegriffen worden sind und sich daher als Erfolgsfaktor eignen.

Beispiel 1:

Für einen Hersteller von kundenangepaßten Standardartikeln (etwa Einbauküchen, Sanitärobjekten) können folgende Eigenschaften den Erfolgsfaktor "Produkt" kennzeichnen:
- *Funktionale Eigenschaften wie Größe; Flexibilität hinsichtlich Raumanforderungen*
- *Geschmackliche Eigenschaften wie Design, Oberfläche und Farbe*
- *Preis (im Vergleich zu Konkurrenzangeboten)*
- *Termingerechte, schnelle und fachmännische Installation*
- *Erweiterbarkeit und Reparaturfähigkeit über lange Zeiträume*
- *Beratung des Absatzmittlers*

Beim Entwurf des Daten- und Funktionsmodells sollten diese Erfolgsfaktoren und mögliche Wettbewerbsvorteile aufgegriffen werden. So zeigen firmeninterne Studien, daß der Kunde sehr viel Wert auf eine optimale Ausnutzung seiner Raumgegebenheiten und auf eine schnelle und absolut termingerechte Installation legt. Um diese Faktoren durch die Informationssysteme zu unterstützen, sollten etwa folgende Attribute erfaßt werden:
:

Kundendaten	*Auftragsdaten*	*Produktdaten*
- *Lokale und technische Auslegung des Installationsortes* - *Kundenkonfiguration* -> *Adressen und persönliche Daten*	- *Liefertermin* - *Installationskennzeichen* - *Absatzmittler*	- *Variantenvielfalt* - *Ersatzteile*

Beispiel 2:

Für einen Hersteller von Konsumgütern mit hohem Individualwert (etwa Kosmetik, Bekleidung) sind folgende Eigenschaften des "Produkts" wichtig:
- *Funktionale Eigenschaften wie Farbechtheit, Wasserbeständigkeit*
- *Qualitative Eigenschaften wie Haltbarkeit; Hautverträglichkeit*
- *Geschmackliche Eigenschaften wie Farbe, Geruch, Design*
- *Preis (im Vergleich zu Konkurrenzangeboten)*
- *Ausrichtung auf individuelle Bedürfnisse (persönlicher Stil, Anpassung an persönliche Körpereigenschaften)*
- *Beratung des Absatzmittlers*

Marktstudien des Unternehmens haben gezeigt, daß die Kunden "maßgeschneiderte" Produkte präferieren würden (Maßkleidung; auf den individuellen Hauttyp abgestimmte Kosmetik o.ä.), darauf aber im Normalfall aus Preisgründen verzichten (Ausnahme: maßgeschneiderte Anzüge aus dem Ostasienurlaub o.ä.). Informationssysteme können durch die Speicherung individueller Kundenbedürfnisse diesen Erfolgsfaktor unterstützen:

Kundendaten	*Auftragsdaten*	*Produktdaten*
-> *Persönliche Merkmale (Größe, Hautkennzeichen)* -> *Psychologischer Typus des Kunden* -> *Adressen und persönliche Daten*	- *Absatzmittler*	-> *Produkteignung für bestimmte Kundenmerkmale*

Die Critical-Sucess-Factor-Methode ist konsequent auf die Wettbewerbsstärke der Unternehmen ausgerichtet. Es werden die Wettbewerbsfaktoren der Branche durchleuchtet (Rahmenbedingungen, Wettbewerber)und die für den Kunden wichtigen Faktoren identifiziert. Rockart (1979) sieht primär folgende Quellen von Haupterfolgsfaktoren: die Struktur der

jeweiligen Industrie, die Unternehmensstrategie, Umsystem-/Umwelteinflüsse und schließlich zeitweise Einflußfaktoren (wie Modeströmungen, zeitweise Marktbedürfnisse).

Bei der Critical-Sucess-Factor-Methode werden folgende Schritte durchlaufen:

1. Die strategischen und operativen Unternehmensziele sind zu analysieren und zu definieren.
2. Daraus sind die kritischen Haupterfolgsfaktoren für die einzelnen Unternehmensfunktionen abzuleiten, die für die strategischen Ziele bestimmend sind.
3. Für die Erfolgsfaktoren sind Meßgrößen zu definieren.
4. Für die resultierenden Hauptsteuerungsgrößen des Managements ist der Informationsbedarf zu ermitteln; es sind die Berichtsinhalte, -frequenzen und -formen der Informationssysteme festzulegen.

	Entwicklung	Beschaffung	Produktion	Vertrieb	Übergreifend
Haupterfolgsfaktoren	1. Entwicklung von Spezialitäten (Kundennutzen) 2. Anwendungstechnik	1. Niedrige Materialkosten 2. Lieferantendiversifizierung	1. Niedrige Herstellkosten	1. Technischer Service/ Kundenbetreuung 2. Innovatives Produktimage 3. Niedriger Verkaufpreis	1. Liquiditätssicherung 2. Kapitalbindung Umlaufvermögen
Gründe	1. Commodities, Fremdbezug preiswerter 2. Technische Problemlösung vor Ort	1. Preisdruck 2. Vermeidung von Abhängigkeiten	1. Preisdruck bei Commodities (von allen Herstellern angebotene Produkte)	1. Problemlöser für Behörden & Vereine 2. Produktimage für Vereine entscheidend	1. Vorfinanzierung von Projekten mit monatelanger Bauzeit
Steuerungsaufgaben	1. Konzentration der Entwicklungsaktivitäten auf innovative Produkte 2. Projektverfolgung	1. Optimale Beschaffungsdisposition 2. Lieferantenauswahl	1. Fertigungslosgröße 2. Verfahrensbeeinflussung in Kostenstelle	1. Kosten-/ Nutzen des technischen Service	1. Projektfinanzierung 2. Senkung von Vorräten und Forderungen
Steuerungsgrößen	1. Marktinformationen 2. Projektzeiten/-kosten + Produkterfolgsrechnung	1. Beschaffungsvolumina	1. Produktionskostenabweichungen	1. Umsatz + Kosten pro Projekt	1. Finanzstatus pro Periode 2. Differenzierter Ausweis von Vorräten und Forderungen

Abb. 3.2.2/21: Ableitung der Steuerungsgrößen aus Haupterfolgsfaktoren nach der Critical Success Factor Methode am Beispiel einer Unternehmung, die Sportplatzbeläge herstellt.

Der CSF-Ansatz betrachtet neben dem Gesamtunternehmen jeden Managementbereich und analysiert dort die spezifischen Erfolgsfaktoren.

Beim Entwurf von Informationssystemen führt dieses Vorgehen dann sowohl zu einem unternehmensweiten Funktions- und Datenmodell als auch zu bereichsspezifischen Modifikationen.

3. Schritt: Analyse der Geschäftsprozesse

Die Methode "Business System Planning" (BSP) ist ein von der IBM in den siebziger Jahren entwickelter Ansatz, die eine effiziente und effektive Lenkung der für die Informationsverarbeitung eingesetzten Ressourcen zur Unterstützung der Organisationsziele erlauben soll. Es handelt sich um einen analytischen Top Down-Ansatz, der auf

- eigenen Analysen des Managements
- strukturierten Interviews durch Systemanalytiker

beruht und das Ziel hat, die Hauptprodukte und die Schlüssel - Geschäfts- und Informationsprozesse sowie die hierfür benötigten Schlüssel-Daten zu identifizieren.

Bei einer BSP-Studie wird ein Team aus einem Mitglied der Geschäftsleitung, jeweils vier Mitgliedern der DV-Abteilung und der betroffenen Fachabteilung und zwei Methodentrainern (die über BSP-Erfahrungen verfügen) zusammengestellt. Eine Studie dauert etwa drei Monate. Sie umfaßt insgesamt 13 Phasen. Zentrale Phasen sind die Phasen 3: Definition von Geschäftsprozessen, 4: Definition von Datenklassen, 5: Analyse der Unternehmens-/Systembeziehungen und die 6. Phase: Definition der Informationssystemarchitektur.

Unter einem Geschäftsprozeß wird eine Gruppe von logisch zusammengehörigen Entscheidungen und Aktivitäten verstanden, die zur Steuerung und Nutzung der Ressourcen, der Produkte oder der Koalitionsteilnehmer einer Unternehmung notwendig sind. Prozesse durchlaufen einen Lebenszyklus von 4 Phasen (Planung, Beschaffung, Verbrauch/Einsatz/Verwaltung, Abschluß).

Prozeß-ansatz	**Objekt**	**Lebenszyklusstufen**			
		Planung	**Beschaffung**	**Verbrauch/ Erstellung/ Verwaltung**	**Abschluß und Kontrolle**
Ressourcen	**Geld**	Finanzplanung Umsatz- und Kostenprognose	Kapitalbeschaffung	Finanzanlagenplanung	Fremdkapitaltilgung Eigenkapitalbedienung
	Personal	Personalplanung	Personalrekrutierung, Aus- und Weiterbildung	Personaleinsatz Stellenbesetzungsplanung	Personalbeurteilung
	Material	Materialdisposition	Materialbeschaffung	Produktion	Entsorgung
	Anlagen	Investitionsplanung	Anlagenkauf	Instandhaltung	Desinvestition
Leistungen	**Produkt**	Marktforschung Produktplanung	Forschung/ Entwicklung	Produktion Logistik	Absatz Recycling
	Projekt	Vertrieb/ Projektplanung	Produkt-/ Personalbereitstellung	Material-/ Arbeitseinsatz Projektabrechnung	Projektcontrolling Projektfakturierung
Marktpartner	**Kunden**	Marktforschung	Werbung	Vertrieb	Fakturierung Debitorenbuchhaltung
	Lieferanten	Lieferantenwahl	Auftragserteilung		Kreditorenbuchhaltung

Abb. 3.2.2/22: Geschäftsprozesse in Anlehnung an eine BSP-Studie

Nach Anwendungserfahrungen (weltweit sind schon über 1500 Studien mit der BSP durchgeführt worden) existieren maximal etwa 60 Prozesse in einem Unternehmen (vgl. Riedl (1991), S.24). Es empfiehlt sich, die Analyse der Geschäftsprozesse nicht zu sehr zu verfeinern. Ziel ist zunächst nicht deren erschöpfende Auflistung, sondern die Selektion der strategisch relevanten Geschäftsprozesse (vgl. Vetter (1988), S.150f).

Üblicherweise basiert diese Analyse auf detaillierten Interviews mit Vertretern unterschiedlicher Managementebenen. Andere Ansätze studieren

-> die Budgets verschiedener Unternehmensebenen und schließen von deren Höhe auf die Bedeutung, die diesen Bereichen zugemessen wird (budget test),
-> die vorhandenen Informationssysteme, da diese den Problemdruck verdeutlichen (system test).

Die Prozesse können nach Funktions- oder Objektbereichen weiter gegliedert werden, doch geht dabei häufig die strategisch orientierte Gesamtsicht verloren.

4. Schritt: Einordnung der Geschäftsprozesse in die Wertschöpfungskette (Value Chain Analysis)

Die Wertschöpfungskette besteht zum einen aus den Mengenflüssen, zum anderen aus den Wertflüssen. In der „Value Chain Analysis" werden beide Ketten von der Beschaffung über die Produktion bis zum Absatz auf potentielle Wettbewerbsvorteile (Kostenvorteile, Produktdifferenzierung, Qualitätsvorteile) untersucht und es wird beurteilt, inwieweit diese durch den parallelen Informationsfluß und den Einsatz von Informationstechnologie verbessert werden können (vgl. Porter/Millar, 1985).

Zu unterscheiden sind

-> die Materiallogistik, die den Materialfluß von der Beschaffung bis zum Absatz verfolgt und dabei vorwärts gerichtet agiert (SCHIEBELOGIK).
-> die Vertriebslogistik, die den Informationsfluß von Angebot/Auftrag über die Produktionsdisposition bis hin zur Beschaffungsdisposition verfolgt und dabei im Wertschöpfungsfluß rückwärts gerichtet agiert (ZIEHLOGIK).

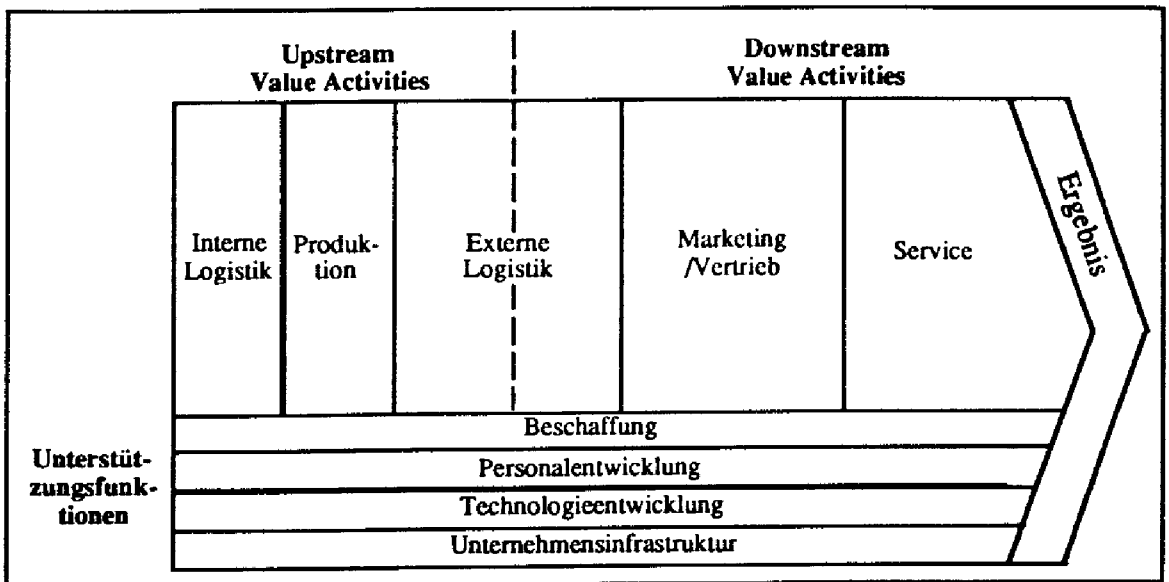


Abb. 3.2.2/23: Value Chain (vgl. Porter (1986) S. 24)

Ziel ist es zum einen, die Geschäftsprozesse in die Wertschöpfungskette einzubinden und dort nach den jeweiligen Erfolgfaktoren zu suchen. Wettbewerbsvorteile können sich ergeben aus Konfigurations- oder aus Koordinationsaufgaben der Prozesse. Konfigurationsaufgaben betreffen Grundsatzentscheidungen hinsichtlich der Ausstattung und regionalen Verteilung

der Produktionseinheiten. Beispielsweise können die Fertigungsstandorte, Fertigungsanlagen und das Fertigungsprogramm so ausgelegt werden, daß möglichst stückkostenoptimal gefertigt werden kann.

Koordinationsaufgaben befassen sich mit der Abstimmung der Aktivitäten in den verschiedenen Wertschöpfungsstufen und Bereichen der Unternehmung. Beispielsweise können bestimmte Forschungsergebnisse bereichsübergreifend verwendet werden, die Abstimmung zwischen den Forschungsabteilungen kann das Ziel der Durchlaufzeitenminimierung verfolgen etc.

Ein weiteres Ziel ist die Analyse der Schnittstellen zwischen den Geschäftsprozessen in der Wertschöpfungskette, die
-> horizontal zwischen den Wertschöpfungsstufen verlaufen,
-> vertikal zwischen ausführender, dispositiver etc. Tätigkeit liegen,
-> lateral verschiedene Geschäftseinheiten verbinden.
Diese Schnittstellen können innerbetriebliche oder auch zwischenbetriebliche Abläufe betreffen und werden analysiert, um den Informationsbedarf im Wertschöpfungsfluß abzustimmen.

5. Schritt: Definition von Datenklassen

Es werden Kategorien von logisch zusammengehörigen Informationen gekennzeichnet, die notwendig sind, um die Objekte in den zugehörigen Geschäftsprozessen zu bearbeiten.
Beispiel:
Nach der CSF-Methode ergab sich für den Sportplatz-Belaghersteller, daß ein wesentlicher Erfolgsfaktor in dem technischen Service und der Problemlösung für Behörden und Vereine liegt. Üblicherweise wenden sich die Kunden an mehrere Anbieter und lassen sich ein Angebot für die Ausstattung des jeweiligen Sportplatzes erarbeiten. Entscheidend für die Auftragsvergabe ist weniger der Preis als vielmehr die Güte der technischen Projektplanung und die termingerechte Projektfertigstellung.

Geschäftsprozeß „Kundenauftrag für Projekt erreichen"				
Datentypus	Plandaten	Beschaffungs-daten	Transaktions-daten	Kontroll-daten
Erforderliche Daten	Baustellen-topographie Technische Anforderungen	Ist-Produkt-lagerbestand Produktions-plan Personaleinsatz	Ist-Material-einsatz Ist-Personal-einsatz	Ist-Projekt-abschluß

Abb. 3.2.2/24: Datentypen und Datenausprägungen für den Geschäftsprozeß "Kundenauftrag über Produkt ausführen"

Bei der Verfeinerung der Datenklassen wird doppelgleisig vorgegangen: Zum einen wird von den Organisationseinheiten ausgegangen, die innerhalb oder außerhalb des Unternehmens existieren. Zum anderen werden die im vorgelagerten Schritt abgeleiteten Geschäftsprozesse verwendet. Die aus beiden Sichten resultierenden Datenkategorien werden hinsichtlich Redundanzen und Inkonsistenzen abgeglichen. Ergebnis der Prozeß- und Datenanalyse ist eine tabellarische Übersicht: Welche Prozesse nutzen für welche Objekte welche Daten?

Objekt "Projekt"	Datentypen			
PROZESSE	**Plandaten**	**Beschaffungs-daten**	**Transaktions-daten**	**Kontrolldaten**
Vertrieb	Baustellentopo-graphie		Absatzpreis Konditionen	
Projektplanung	Technische Anforderungen			Technische Abweichungen
Material-/ Personalbereit-stellung	Plan-Material/ Arbeitseinsatz Plan-Projektend-termin	Ist-Produkt-lagerbestand Produktionsplan Personaleinsatz		
Projekt-abrechnung			Ist-Material-mengen Ist-Arbeits-stunden	Ist-Projektend-termin
Projekt-fakturierung				
Projekt-controlling	Plan-Material/ Arbeitseinsatz Plan-Projektend-termin			Plan-/Ist-Abweichungen Umsatz-/Kostendaten Vergleichsdaten

Abb. 3.2.2/25: Informationen für ein Projekt in den betroffenen Geschäftsprozessen

In der anschließenden Analyse der Unternehmens-/Systembeziehungen wird untersucht, inwieweit die Informationsverarbeitung die Prozeß- und Datensicht bisher unterstützt. Die Beziehungen zwischen DV-Systemen, Organisationseinheiten, Daten und Prozessen machen die DV-Abdeckung und DV-Lücken deutlich. Daraus erfolgt die Definition der Systemarchitektur, die die natürlichen Cluster zwischen Organisation, Daten, Prozessen und DV-Systemen beschreibt und daraus ableitet, welche DV-Projekte für das Unternehmen in Zukunft notwendig sind.

3.2.3. Datenmodelle als Teil von Informationssystemen

Das Informationssystem besteht grundsätzlich
(1) aus dem Funktionsmodell
(2) aus dem Datenmodell
(3) aus dem Kommunikationsmodell

3.2.3.1. Informationssystem-Entwurf

Das Informationssystem ist als Modell ein Abbild des Objektsystems. Es ist zu entscheiden ,
- welche Teile der realen Welt betrachtet werden sollen (Abgrenzungsentscheidung)?
- welche Schnittstellen zwischen dem betrachteten Teil und dem nicht betrachteten Umsystem einbezogen werden (Schnittstellenentscheidung)?
- welche Objekte und Beziehungen in der realen Welt näher betrachtet und welche vernachlässigt werden sollen (Selektionsentscheidung)?
- wie detailliert die Elemente des realen Objektsystems zu betrachten sind (Aggregationsentscheidung)?

Das Informationssystem soll dabei möglichst semantisch gehaltvoll sein. Modelle, die so aufgebaut sind, beschreiben vollständig und strukturell richtig die Elemente des realen Objektsystems. Der Begriff „Semantik" bedeutet dabei die Abbildung der realen Objekte in (sprachliche) Ausdrücke, die später Grundlage der Datenverarbeitung sind.

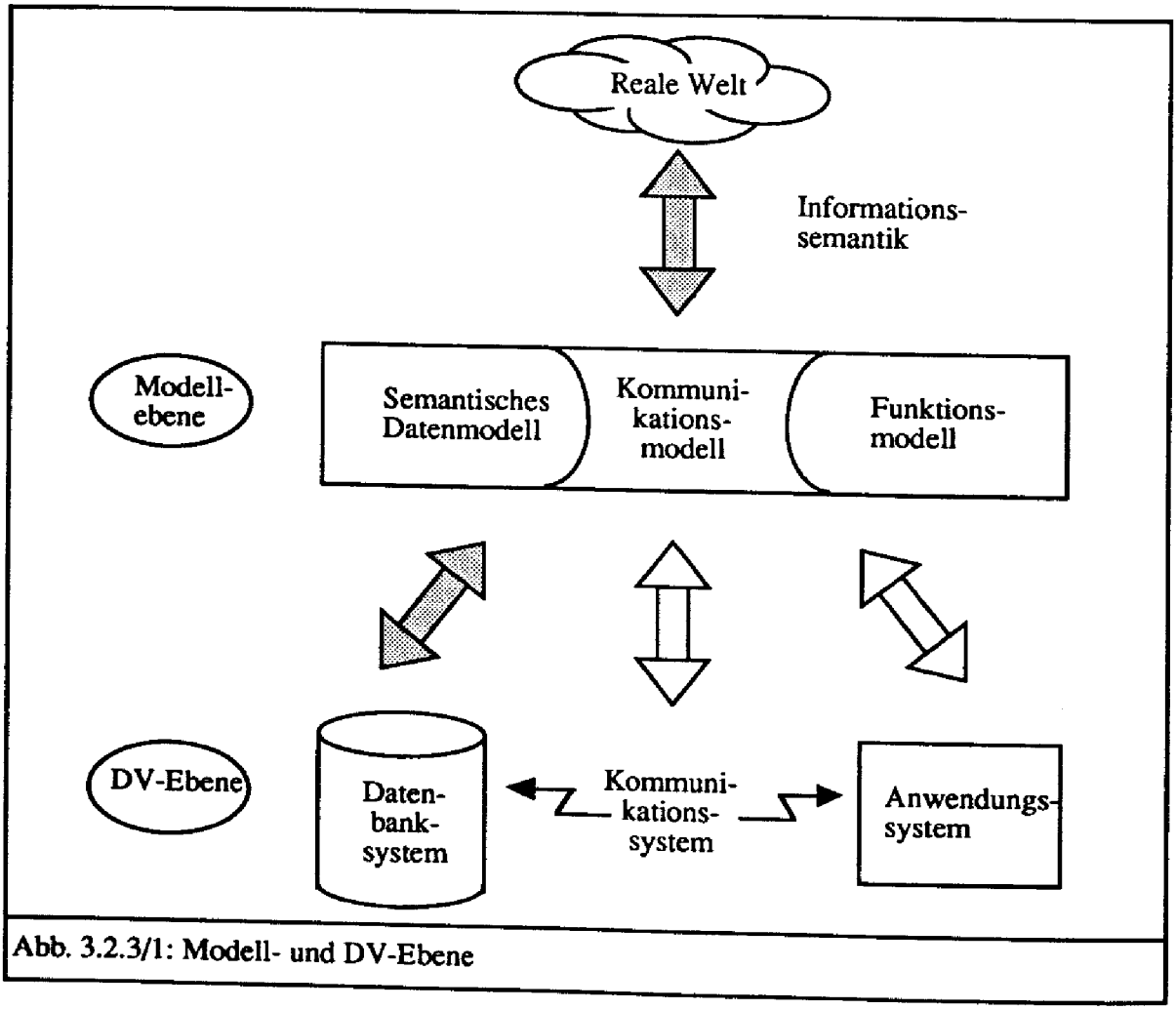


Abb. 3.2.3/1: Modell- und DV-Ebene

Informationssemantik besagt, daß das Informationsmodell die Struktur, die Randbedingungen und das Verhalten des Objektsystems (homomorph) strukturrichtig wiedergibt.

Für die Ableitung eines Datenverarbeitungssystems aus dem Informationsmodell müssen dessen Elemente eindeutig bestimmbar sein und für ihre Eigenschaften Meßprozeduren vorliegen. Man spricht dann von Datensemantik (vgl. Dampney (1988)).

Informationssemantik	**Datensemantik**
- strukturgleich - verhaltensgleich	- zählbare, identifizierbare Datenelemente - Meßoperationen programmierbar
* nur subjektiv beurteilbar	* mathematisch formal ableitbar

Abb. 3.2.3/2: Unterscheidung Informationssemantik und Datensemantik (vgl. Dampney (1988))

Ziel des Informationsmodells ist es, die Gegebenheiten des Objektsystems richtig widerzuspiegeln. Anliegen ist eine formal eindeutige und inhaltlich vollständige Beschreibung aller relevanten Elemente des Objektsystems, unabhängig davon, ob sich diese Elemente hinterher in der gleichen Struktur im DV-System abbilden lassen. Da das Informationsmodell für lange Zeit eine stabile Basis der DV-Systementwicklung bilden soll, wird darauf vertraut, daß sich die technischen Möglichkeiten im Laufe der Zeit den inhaltlichen Erfordernissen anpassen.

Auf der DV-Ebene soll das Informationsmodell unter Beachtung technologischer Möglichkeiten, ökonomischer Randbedingungen und betriebswirtschaftlicher Meß- und Bewertungskonzepte in ein funktionsfähiges DV-System überführt werden. Um die Elemente des Informationssystems auf den Komponenten des DV-Systems abzubilden, werden in der Literatur zur Systemanalyse folgende Prinzipien genannt:

Prinzip	Beschreibung	Vor-/Nachteile
der logischen Konsistenz	Die Input-, Prozeß- und Outputstrukturen des Informationssystems sind logisch konsistent auf den Elementen des DV-Systems abzubilden.	+ Überprüfbarkeit durch DV- und Fachverantwortliche
der Lokalität	Zusammengehörende Elemente des Informationssystems sind physisch lokal den Elementen des DV-Systems zuzuordnen.	+ Vermeidung von langen Kommunikationswegen + Vermeidung von Schnittstellen
der Vollständigkeit	Die Elemente des Informationssystems sind vollständig auf den Elementen des DV-Systems abzubilden.	+ Vollständige Abdeckung der Nutzeranforderungen
der Modularisierung	Modulare Elemente des Informationssystems sollen auf modularen Elementen des DV-Systems abgebildet werden.	+ Modulare Austauschbarkeit in vertikaler Hinsicht + Arbeitsteilige Entwicklung
der Verifizierbarkeit	Modulare und gleichzeitige Testbarkeit von Elementen des Informationssystems und des DV-Systems.	+ Schnellere Entwicklung

Abb. 3.2.3/3: Prinzipien der Abbildung des Informationssystems auf das DV-System

3.2.3.2. Kennzeichen des Datenmodells

Ein Datenmodell ist eine sprachliche Beschreibung der in einem Unternehmen zu verwendenden Datenelemente. Es wird versucht, diese strukturiert und vollständig zu beschreiben, um

- den Leistungsprozeß von DV-Systemen unabhängig von den beteiligten Personen zu vereinheitlichen (personelle Transparenz),
- wichtige Zusammenhänge zwischen den Teilsystemen zu identifizieren (interfunktionelle Transparenz).

Je nach den Besonderheiten der Geschäftsprozesse können drei Stufen der Spezifizierung des Datenmodells unterschieden werden

	Kennzeichen	Freiheitsgrade der Modellierung	Beispiele
Unternehmens-spezifiziertes Datenmodell	-> Datenmodell bildet Wissen ab, das andere nicht besitzen -> Datenmodell eröffnet Wettbewerbsvorteile	Nur durch Eigenschaften verfügbarer Software-Entwicklungswerkzeuge eingeschränkt	Marktinformations-systeme Controlling-Systeme Warenwirtschafts-systeme
Branchen-spezifiziertes Datenmodell	-> Datenmodell bildet Wissen ab, das im wesentlichen alle Wettbewerber besitzen -> Organisationsanpas-sung ist effizienter als DV-Entwicklung	Kaufmännische oder rechtliche Gepflogen-heiten engen ein	Verkaufsabwicklung Personalabrechnung
Branchen-übergreifendes Datenmodell	-> Datenmodell bildet Wissen ab, das alle Unternehmen einer Volkswirtschaft besitzen	Rechtliche Rahmenbe-dingungen oder kauf-männische Gepflogen-heiten schränken ein.	Finanzbuchhaltung

Abb. 3.2.3/4: Spezifizierungsstufen des Datenmodells (vgl. Beha/Huy, (1990))

Nur in dem Fall, daß sich im Datenmodell technologisches oder organisatorisches Wissen widerspiegelt, das andere Wettbewerber nicht besitzen, bietet sich an, das Modell individuell zu entwickeln. In den anderen beiden Fällen ist zu prüfen, ob die Modellierung nicht mit dem Ziel des Einsatzes einer Standardsoftware betrieben werden sollte, da dieses oft der wirtschaftlichere Weg ist. Dann ist der Entwurfsprozeß an den Strukturen der Standard-software auszurichten.

Ähnlich wie in der klassischen Organisationslehre wird versucht, zunächst die Gesamt-aufgabe des Unternehmens durch ein globales Datenmodell zu beschreiben.

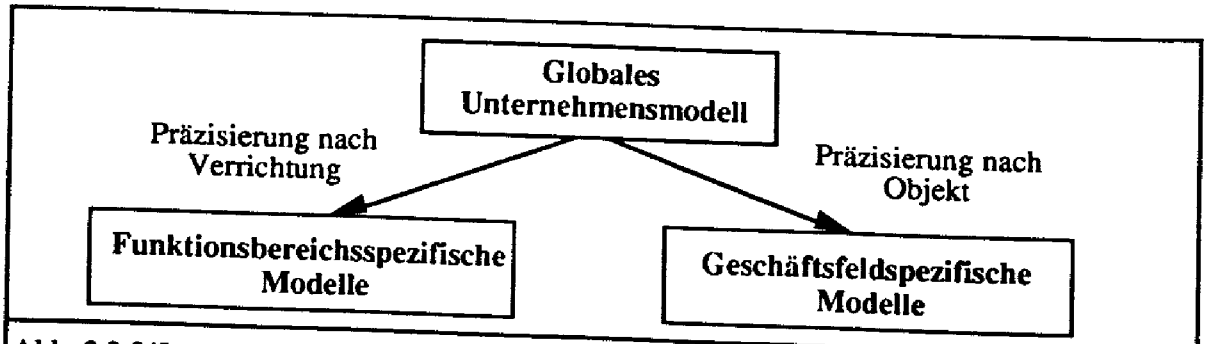


Abb. 3.2.3/5: Präzisierung des globalen Unternehmensmodells

Dieses grobe und globale Unternehmensmodell wird dann stufenweise präzisiert. Verbreitet ist eine Präzisierung nach Objekten mit dem Ziel der Entwicklung geschäftsfeldspezifischer Datenmodelle und eine Präzisierung nach Verrichtungen mit dem Ziel funktionsbereichs-spezifischer Modelle. Wesentlich für den Entwurfsprozeß ist, die vorzufindenden Strukturen des Unternehmens zu analysieren und entsprechend der Gestaltungsziele neu zu ordnen.

Der Entwurf des Datenmodells wird dabei an den generellen und den prozeß- und funktionsbezogenen Erfolgsfaktoren des Unternehmens orientiert.

	Kennzeichen	Differenzierungsbeispiele
Unternehmens-Datenmodell	- Zusammenstellung der wichtigsten Entity- und Beziehungstypen des Leistungsprozesses - Verfeinerung um die unternehmensspezifischen Haupterfolgsfaktoren innerhalb der Branche	- Kundentypen und -merkmale - Kundennutzenbestimmende Produktmerkmale - Marktstrukturen - Organisationsstrukturen
Prozeß-Datenmodell	- Differenzierung der Geschäftsprozeßstrukturen - Verfeinerung um die prozeßspezifischen Haupterfolgsfaktoren	- Output-Typen (Einzel-, Serien-, Massenfertigung) - Input-Typen (rohstofforientiert) - Prozeß-Typen (Fertigungstiefe)
Funktions-Datenmodell	- Unterteilung entsprechend der Aufbauorganisation - Verfeinerung um die jeweils funktionsspezifischen Haupterfolgsfaktoren	- Dominierende Funktionsziele: Kosten, Zeiten, Qualität, Innovation

Abb. 3.2.3/6: Unterscheidung Unternehmens-, Prozeß- und Funktions-Datenmodell

Kapitel 2: Datenmodellierung und Datenbank-Entwurf

1. Probleme des Datenbankentwurfs

Aufgabe des Datenbankentwurfs ist es, ausgehend von den Sachverhalten der Realität formalisierte, vom Rechner und von der Systemsoftware handhabbare Beschreibungen der gewünschten Daten zu schaffen (vgl. Dittrich (1990), S. 229).

Beim Datenbankentwurf stellen sich Fragen des methodischen, systemtechnischen und organisatorischen Vorgehens, die im folgenden behandelt werden sollen (vgl. Lockemann/ Rademacher (1990)).

1.1. Wie ist die Wirklichkeit im Datenmodell zu beschreiben?

Bei der Erfassung der Wirklichkeit des Unternehmens in einem Modell lassen sich drei Sichtweisen unterscheiden:

Mit der strukturellen Sicht wird die im Unternehmen existierende Aufbau- und Ablauforganisation zu beschreiben versucht.

Die materielle Sicht beschreibt den Mengen- und Wertefluß im Unternehmen unabhängig von deren Aufteilung auf Organisationseinheiten und Organisationsabläufe.

Die personelle Sicht erfaßt die Mitarbeiter, deren Fertigkeiten und Fähigkeiten sowie deren Nutzbarkeit für den Zweck des Datenbank-Entwurfs.

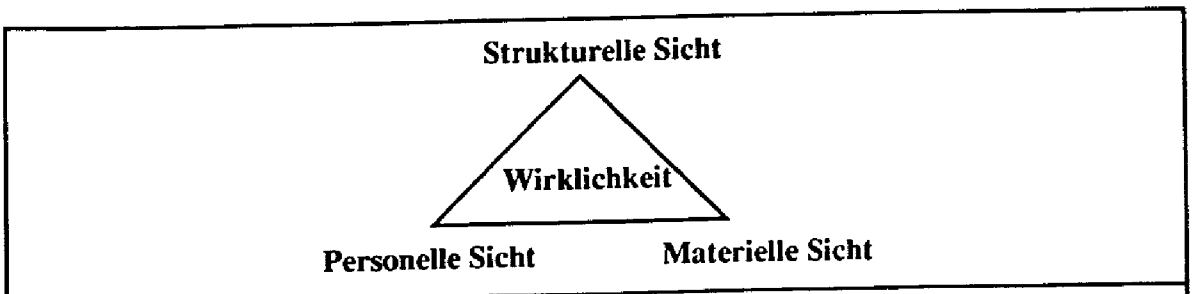


Abb. 1.1/1: Dimensionen der Beschreibung der Wirklichkeit

(1) Strukturelle Sicht

Mit der strukturellen Sicht wird die Organisationsstruktur des Unternehmens beschrieben. Verwendet werden dazu Analyse- und Darstellungsinstrumente, die aus der Organisationslehre abgeleitet und für Zwecke der Systemanalyse verfeinert werden.

Von der IBM wird beispielsweise ein Instrument unter dem Begriff „Information System Study" (ISS) vorgeschlagen, das als Dimensionen die ORGANISATION, (d.h. die funktionelle und räumliche Gliederung der Unternehmung), GESCHÄFTSPROZESSE und deren AUSLÖSER sowie DATEN (von Datenobjekten und Datenbeziehungen) unterscheidet (vgl. Vetter (1988), S. 148 f.)

Strukturelement	Subelement	Erläuterung
ORGANISATION	LOKATION	Räumliche Verteilung der Organisationseinheiten eines Unternehmens
	FUNKTION	Funktionen der Organisationseinheiten eines Unternehmens (z. B. in Form von Aufgabenbeschreibungen)
PROZESSE	ZWECK	Gruppe von logisch zusammenhängenden Entscheidungen und Aktivitäten, die zur Steuerung und Verwaltung von - Ressourcen - Produkten (Dienstleistungen/marktgängige Leistungen) - Umweltfaktoren des direkten und indirekten Umsystems der Unternehmung erforderlich sind.
	AUSLÖSER	Ereignisse, die das Ergebnis zeitlich vorgelagerter Prozesse sind und wiederum Prozesse auslösen.
DATEN	OBJEKTE	Datenobjekte sind individuelle und identifizierbare Exemplare von Dingen oder Begriffen der realen oder der Vorstellungswelt, für welche Informationen festzuhalten sind. Als Datenobjekte kommen grundsätzlich in Frage - ein Individuum (Mitarbeiter, Kunde, Lieferant etc.) - ein reales Objekt (z. B. eine Maschine, ein Gebäude, ein Produkt) - ein abstraktes Konzept (z. B. ein Fachgebiet, eine Kostenstelle) - ein Ereignis (z. B. Verkauf, Verbuchung)
	BEZIEHUNGEN	Beziehungen sind identifizierbare und individuelle Aktivitäten zwischen Objekten, die in der realen oder der Vorstellungswelt existieren und für die Informationen festzuhalten sind. Als Beziehungen kommen grundsätzlich in Frage - reale Aktivitäten, die beobachtbar sind und sich daher durch Verben (besser „Tuwörter") kennzeichnen lassen (z. B. kaufen, liefern, bearbeiten) - abstrakte Verhältnisse, die beschreibbar sind und sich durch Verben kennzeichnen lassen (z. B. gehört zu, vertritt die)

Abb. 1.1/2: Dimensionen der strategischen Anwendungs-/Datenplanung (vgl. Vetter (1988), S. 148ff.)

Geschäftsprozesse sind die Aktivitäten, die innerhalb einer Organisation durchzuführen sind und die durch die Ablauforganisation geregelt werden. Mehrere Geschäftsprozesse können zu PROZESSGRUPPEN und diese werden wiederum zu FÜHRUNGSSYSTEMEN zusammengefaßt.

(2) Materielle Sicht

Die Betriebswirtschaftslehre bietet eine Reihe von Gedankengebäuden an, die zur Analyse und Strukturierung des Leistungs- und Wertschöpfungsprozesses in einer Unternehmung geeignet und nützlich sind. Die folgende Tabelle stellt einige dieser üblichen betriebswirtschaftlichen Kategorien zusammen.

	Referenzmodell	**Beispiele für Kategorien**
Informationsfluß	Buchhaltung	Konto Geschäftsstelle Kontenrahmen
	Kosten-/ Leistungsrechnung	Kostenart Kostenstelle Kostenträger
	Organisationslehre	Aufgabe - Gesamtaufgabe/Teilaufgaben/Elementaraufgaben - Aufgabensynthese - Verrichtung/Objekt/Phase/Raum/Zeit/Zweck/Rang Aufbauorganisation - Stelle/Instanzen/Abteilung - Zentralisation/Dezentralisation Ablauforganisation - Arbeitsanalyse/Arbeitssynthese
	Organisationstheorie	Rolle Organisationsmitglieder Organisationsteilnehmer Gruppen, formelle und informelle Delegation, Partizipation, Kommunikation
Zahlungsfluß	Finanzierungslehre	Güterstrom Geldstrom Eigen-/Fremdfinanzierung Innen-/Außenfinanzierung Auszahlung/Einzahlung Ausgabe/Einnahme Finanzphase
Leistungsfluß	Produktionstheorie	Input, Prozess, Output

Abb. 1.1/3: Beispiele nützlicher Kategorien zur materiellen Analyse im Rahmen der Datenmodellierung

Viele Instrumente der Systemanalyse und Datenmodellierung verwenden ähnliche Kategorien, die dann jedoch auf einem höheren Abstraktionsniveau definiert werden, da sie nicht nur für betriebswirtschaftliche Anwendungssysteme konzipiert wurden.

Die Verwendung der üblichen betriebswirtschaftlichen Kategorien erleichtert die Kommunikation mit den Fachabteilungen in den Unternehmen und ist daher abstrakten Begriffssystemen vorzuziehen.

(3) Personelle Sicht

Grundlegendes Ziel jeder Systementwicklung ist die Deckung des Informationsbedarfs spezifizierter Systemnutzer. Dabei sind folgende Unterscheidungen wichtig:

1. Welche Typen von Systemnutzern existieren?
2. Welchen Informationsbedarf haben diese Systemnutzer?

Der Datenbank-Entwurf ist eine Aufgabe, die gemeinsam von Systemanalytikern, den Datenbankspezialisten und den Vertretern der Fachabteilungen zu bewältigen ist. Die Kenntnisse der Fachabteilung über die Problemstrukturen und Abläufe sind das Ergebnis einer langjährigen intensiven Beschäftigung mit der Materie und verkörpern das Fachwissen der Unternehmung. Im Rahmen des Datenbank-Entwurfs ist dieses Fachwissen, das sich häufig nur in den Köpfen der Mitarbeiter befindet, für das DV-System nutzbar zu machen. Eine intensive Zusammenarbeit mit der Fachabteilung erhebt somit nicht nur deren Anforderungen an das zukünftige System, sondern sie beschleunigt auch den Entwurf und führt zu stabilen und akzeptierten Ergebnissen.

Auch beim Betrieb des DV-Systems sind die Fachnutzer und die DV-Nutzer zu unterscheiden. Die Fachnutzer sind am materiellen Gehalt der gelieferten Informationen interessiert, während die DV-Nutzer an der Wartbarkeit, Elastizität und Flexibilität des Systems interessiert sind.

Der Informationsbedarf läßt sich in folgenden Punkten unterscheiden:

Dimension	Fragen	Beispiele
Informationsumfang	Welche Sachverhalte der Realität sind für die Nutzer relevant? Nach welchen Kategorien sollen die Informationen gegliedert werden?	Umsatz pro Produkt pro Kunde pro Region pro Verantwortungsbereich
Informationszeit	Wann sind die Informationen zu liefern? In welcher Frequenz sind die Informationen zu liefern?	- feste Zeitpunkte - in Abhängigkeit von bestimmten Ereignissen
Informationsform	Wie sind die Informationen aufzubereiten?	- graphisch - standardisierte Kommentare

Abb. 1.1/4: Merkmale des Informationsbedarfs

1.2. Wie soll beim Entwurf vorgegangen werden ?

Der Entwurf des Datenmodells kann entweder induktiv oder deduktiv erfolgen. Beim induktiven Vorgehen (Bottom Up-Struktur) werden zunächst die Sachverhalte der realen Welt erfaßt und in den gewählten Kategorien systematisiert. Dabei wird in mehreren Phasen zunehmend von den Einzelausprägungen abstrahiert; meist werden zunächst betriebswirtschaftliche oder technische Gattungsbegriffe (wie Auftrag, Kunde, Maschine, DV-Anlage) und dann informatorische Gattungsbegriffe (wie Objekt, Operator) verwendet.

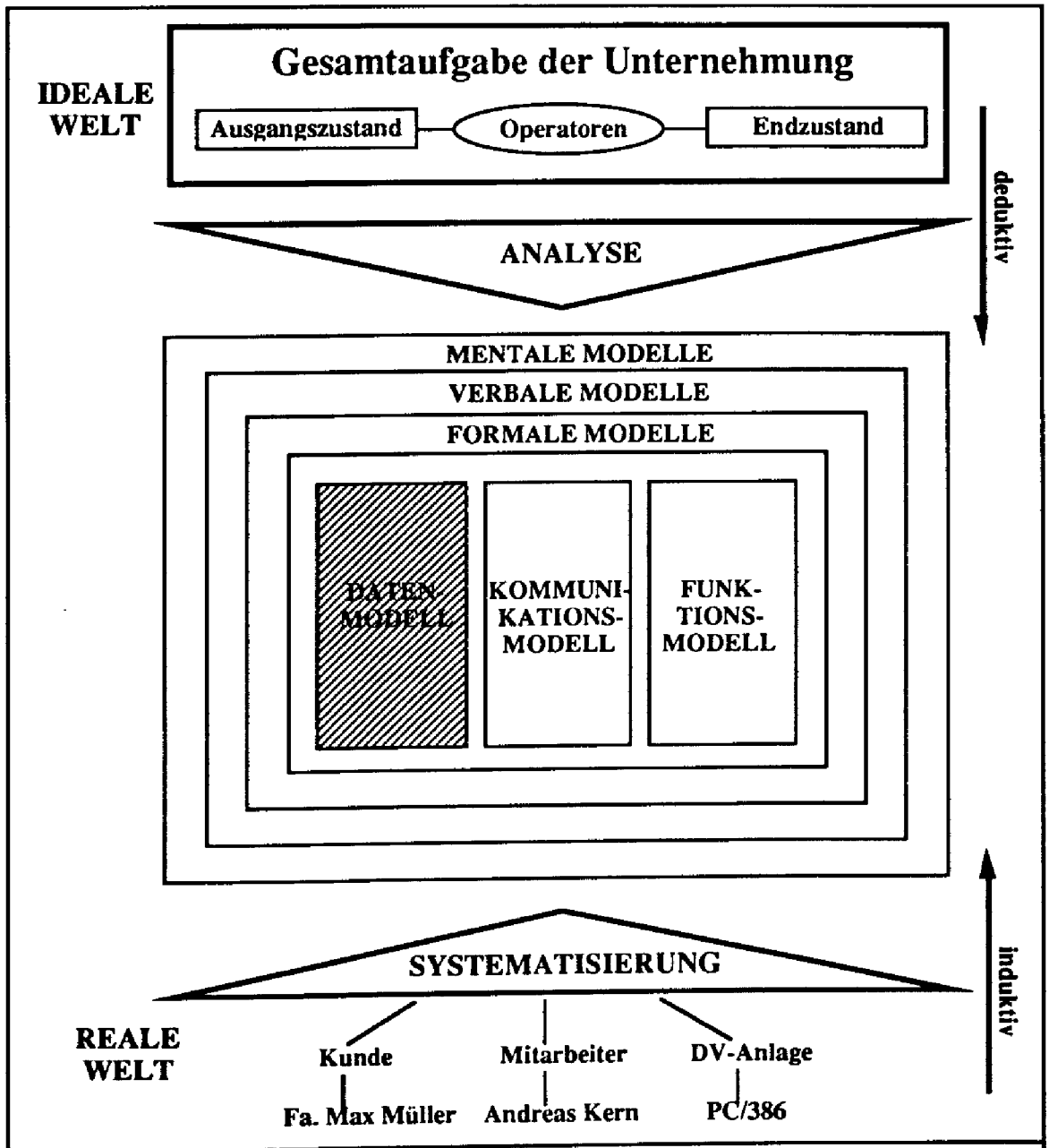


Abb. 1.2/1: Induktives und deduktives Vorgehen beim Entwurf betrieblicher Anwendungssysteme

Meistens werden die Sachverhalte der Realität nach der mentalen Analyse zunächst verbal beschrieben, bevor sie mit Hilfe einer formalen Beschreibungstechnik explizit in einem Modell abgebildet werden.

Beim deduktiven Vorgehen (Top Down-Struktur) wird von der Gesamtaufgabe der Unternehmung ausgegangen und diese wird in einer Art Aufgabenanalyse nach bestimmten Kriterien (klassisch: Verrichtung, Sachmittel, Zweckbeziehung Rang, Phase, Ort) zerlegt (vgl. Kosiol(1962)). Statt der von "klassischen", von Kosiol geprägten Kriterien schlägt Zehnder vor, den betrieblich - organisatorischen Sachverhalt nach den fünf Kategorien

- Personen
- Materialien
- Objekte
- Informationen
- Energien

zu systematisieren (vgl. Zehnder (1989)).

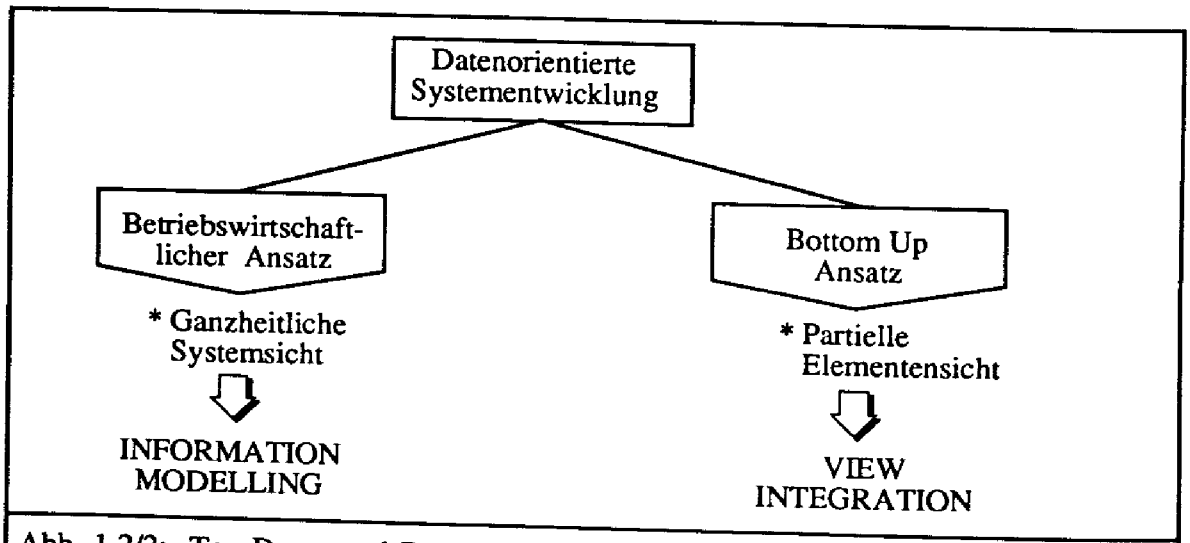


Abb. 1.2/2: Top-Down und Bottom Up-Ansatz der Datenorientierten Systementwicklung im Überblick (vgl. Zimmermann (1983))

Ausgangspunkt der Abstraktions-Entscheidungen sind die Unternehmensziele, die mit den zu entwerfenden Informationssystemen zu erfüllen sind und die ökonomischen Anforderungen an diese Systeme. Das Prinzip der Abstraktion fordert, sich bei jedem Entwurfsschritt auf das, für die ökonomischen Anforderungen Wesentliche zu konzentrieren und unwesentliche Details zu vernachlässigen.

1.3. Wann ist das Datenmodell zu entwickeln?

Da das Datenmodell eine über lange Zeit stabile Basis für die darauf aufbauenden DV-Prozesse (Programmsysteme) bilden soll, hat der Datenbankentwurf idealtypisch natürlich zeitlich vorgelagert zur Entwicklung der Anwendungsprogramme zu erfolgen.

Diese idealtypische Vorstellung ist in der Unternehmenspraxis nur selten einzuhalten, da in aller Regel schon eine Systemarchitektur besteht. Häufig ist diese über viele Jahre gewachsen und stellt eine erhebliche Investition für das Unternehmen dar.

Neuentwurf und Re-Design

Der Entwurf von Informationssystemen geschieht nur selten von Grund auf neu, da in den Unternehmen in der Regel eine Vielzahl von erprobten DV-Systemen existieren. Zu unterscheiden sind der totale Neuentwurf eines unternehmensweiten Informationssystems vom partiellen Neuentwurf eines Teils und vom Redesign/Re-Engineering Informationssystemteile.

	Totaler Neuentwurf	**Partieller Neuentwurf**	**Re-Design vorhandener Systeme**
Kennzeichen	- umfassende betriebswirtschaftliche Grob-/Feinkonzeption notwendig - Parallel häufig Auswahl von Hardware- und Softwarekomponenten	- Häufig schlecht dokumentierte Schnittstellen - Nicht integrierte semantische Strukturen	- Häufig schlecht dokumentierte Datenstrukturen - Nicht integrierte, heterogene semantische (Begriffs-)Strukturen - Primitive Dateistrukturen
Aufgaben	- Strategisch orientierte Anwendungssystementwicklung - Unternehmensdatenmodellentwicklung	- Abstimmung Funktions-/Kommunikations-/-Datenmodell - Integration in vorhandene Systemlandschaft	- Nachdokumentation der Datenstrukturen - Schaffung einheitlicher Datenstrukturen - Sicherstellung der Nutzbarkeit der alten Datenstrukturen
Vorgehensweise	- Top Down	- Bottom Up	- Gegenstrom-Verfahren

Abb. 1.3/1: Neuentwurf und Re-Design von Informationssystemen

Der langwierige Prozeß des schrittweisen Übergangs von einer an Einzelanwendungen ausgerichteten Datenorganisation zu einer anwendungsübergreifenden, strategisch und unternehmensweit ausgerichteten Organisation der Informationsressourcen muß durch Methoden in der strategischen Unternehmensplanung, der DV-Systementwicklung und beim Redesign unterstützt werden.

Ein sinnvoller Ansatz ist die Datenmodellierung mit den in der Folge beschriebenen Methoden und Instrumenten. Das Datenmodell dient als Bezugsrahmen sowohl der Neuplanung, des partiellen Neuentwurfs als auch der Überarbeitung von Altsystemen.

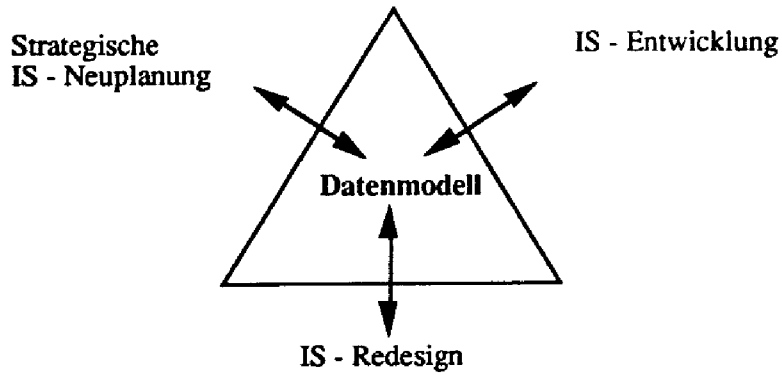


Abb. 1.3/2: Das Datenmodell als zentraler Bezugspunkt des Informationssytem-Entwurfs

Je nach Entwurfssituation ist die Rolle des Datenmodells unterschiedlich. Bei einer strategischen System-Neuplanung wird das Datenmodell im Top-Down-Ansatz schrittweise verfeinert und die Datenmodellierung erfolgt zeitlich vorgelagert. Bei einer Überarbeitung vorhandener DV-Altsysteme wird eine zusätzliche Analysephase (Bottom Up) durchgeführt, um die Datenstrukturen der Altsysteme zu durchdringen. Dabei wird deren Informationsbedarf bis auf die Attributebene aufgedeckt.

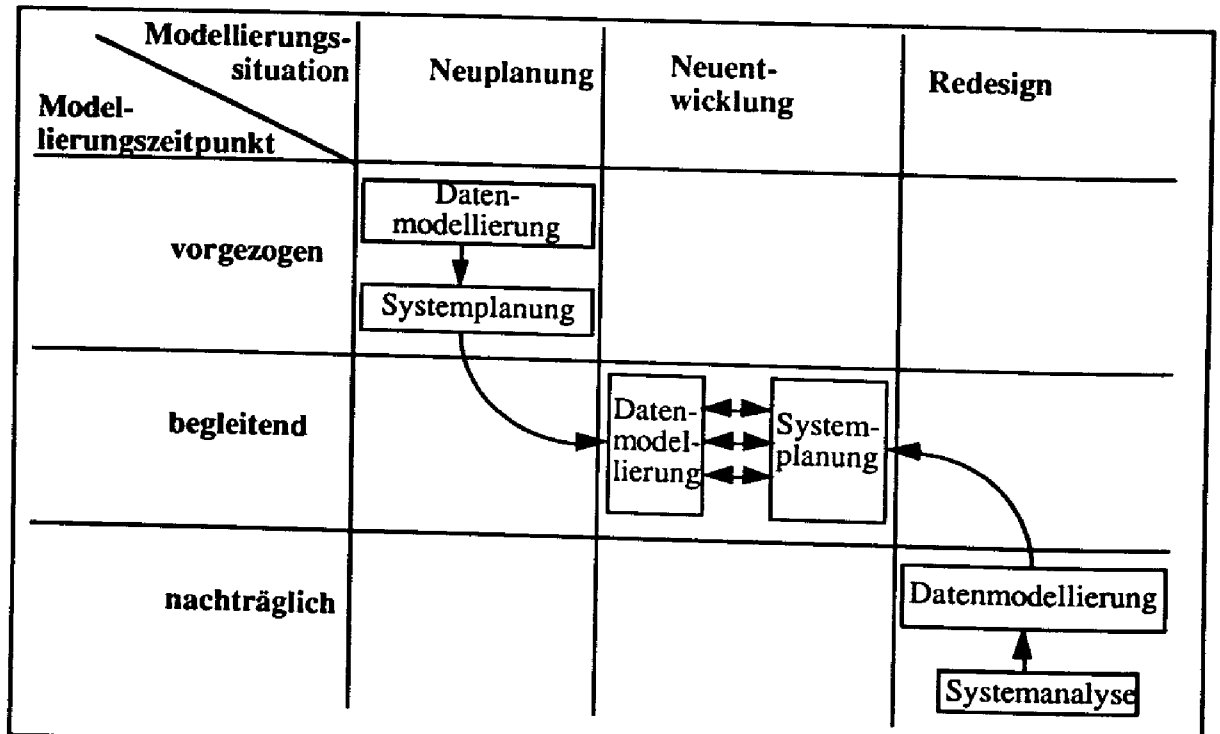


Abb. 1.3/3: Vorgehensvarianten der Datenmodellierung (aus: Ortner, E. (1991), S. 277)

Im Anschluß an diese Analysephase (Reverse-Engineering) sind die Datenelemente neu in der gewählten Methodik zu beschreiben und unternehmenseinheitlich zu definieren (Nachmodellierung). Im dritten Schritt sind schließlich die Altsysteme durch Systeme abzulösen, die die Strukturen des Datenmodells umfassend abbilden (Forward Engineering) (vgl. Ortner (1991), S.277 f.)

1.4. Wie ist die Datenmodellierung organisatorisch zu verankern?

Die Integration von betriebswirtschaftlichen Anwendungssystemen mittels Datenintegration ist nur zu einem kleineren Teil eine technische, sondern vielmehr entscheidend eine organisatorische Aufgabe, die entsprechende aufbau- und ablauforganisatorische Regelungen erfordert. Grundsätzlich bieten sich drei Organisationsalternativen zur Verankerung der Datenmodellierung an (vgl. Ortner (1991), S. 275):

1. Es wird eine zentrale Stelle geschaffen, die alle Modellierungsaufgaben eigenverantwortlich wahrnimmt.

2. Die Datenmodellierung bildet eine zentrale Service-Stelle, die primär koordinierende Aufgaben wahrnimmt und ihre Mitarbeiter in die Entwicklungsprojekte delegiert, in denen die eigentliche Modellierung vorgenommen wird. Die zentrale Service-Stelle konzentriert sich auf die Einhaltung von Datenstandards und auf die Integration in das übergreifende Unternehmensdatenmodell.

3. Die Datenmodellierung ist nicht als separate Stelle institutionalisiert, sondern die Aufgabe der Datenmodellierung findet in den einzelnen Entwicklungsprojekten oder in den einzelnen Software-Verantwortungsbereichen statt.

Organisatorische Alternativen	Vorteile	Nachteile
Zentrale Datenmodellierung	- Einheitlichkeit und Integration - Zentrales Datenlexikon wird gepflegt	- Trennung von eigentlichen Entwicklungsprojekten (mangelndes Problemverständnis) - Nadelöhr Datenmodellierung denkbar
Zentrale Service-Stelle	- Unterstützung der Entwicklungsprojekte - Steigerung des Problemverständnisses in der Datenmodellierung	- Unternehmensweite Integration gefährdet, muß durch Personen und Instrumente gesichert werden. - Service-Stelle muß ausreichende Kapazitäten vorhalten, da sonst Wartezeiten in den Entwicklungsprojekten entstehen.
Verteilte Datenmodellierung	- Problemverständnis und Problemdurchdringung in Entwicklungsprojekten am größten	- Datenmodellierung und Datendokumentation wird zur (oft vernachlässigten) Nebenaufgabe

Abb. 1.4/1: Aufbauorganisatorische Alternativen zur Verankerung der Datenmodellierung

Die Probleme bei der organisatorischen Etablierung der Datenmodellierung liegen darin, daß der DV-Sachverstand und die betriebswirtschaftliche Problemdurchdringung in den einzelnen DV-Projekten am größten ist, eine dezentrale Datenmodellierung aufgrund der unterschiedlichen Verständnisse und der geringen Priorität die erforderliche Einheitlichkeit und die Integration des Datenmodells gefährdet.

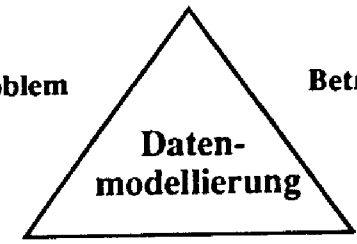


Abb. 1.4/2: Spannungsfeld der organisatorischen Verankerung

Die Integration ist üblicherweise nur zu sichern, wenn eine zentrale Stelle verantwortlich ist für die Dokumentation der Datenelemente und für die Konsistenz des Datenmodells.

Zusätzlich ist es erforderlich, daß die betroffene Fachabteilung sich für die Datenobjekte nach Abschluß eines DV-Projektes verantwortlich fühlt.

Neben der aufbauorganisatorischen Etablierung der Datenadministration ist daher auch eine über längere Zeit stabile ablauforganisatorische Regelung zu finden, die die Interessen des Informationsmanagements, der DV-Abteilung und der Fachabteilungen berücksichtigt. Die folgende Abbildung zeigt die aufbau- und ablauforganisatorische Stufung der Datenmodellierung und Datenadministration in der Hoesch AG (vgl. Mertes/Klonki (1991)).

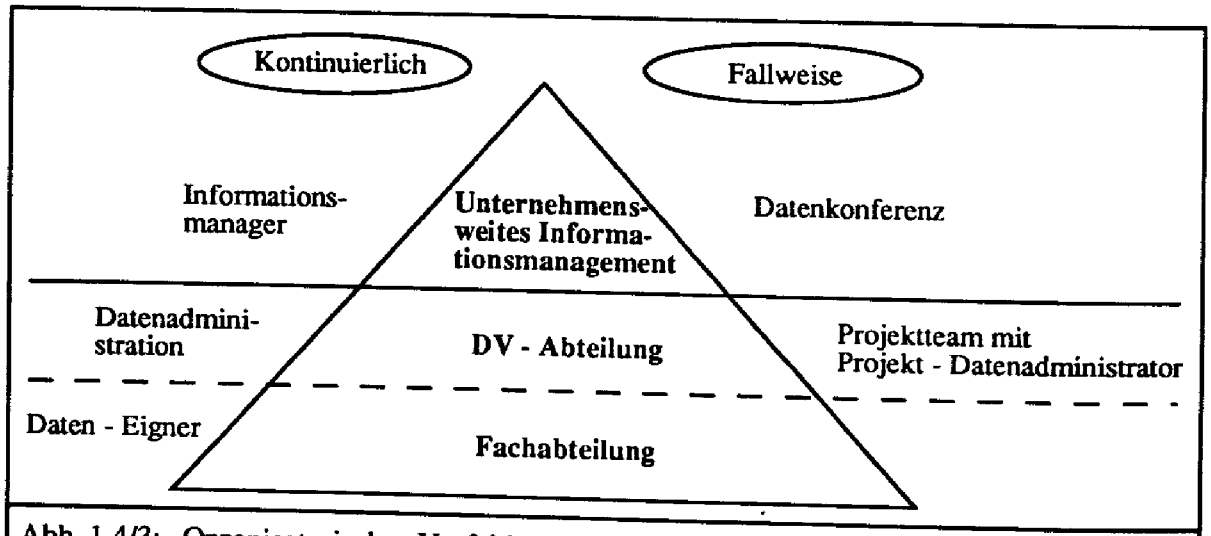


Abb. 1.4/3: Organisatorisches Umfeld der Datenmodellierung am Beispiel der Hoesch AG (vgl. Mertes/Klonki (1991))

2. Gliederung des Entwurfsprozesse

2.1. 3-Schema-Architektur nach ANSI/SPARC

Die 3-Schema-Architektur wurde bis 1975 von der US-amerikanischen Organisation ANSI/X3/SPARC Study Group on Data Base Management Systems unter maßgeblicher Mitwirkung der IBM entwickelt, um die Datenbanksystem-Architekturen der Zukunft zu konzipieren.

Zentrales Anliegen des Entwurfs ist Datenunabhängigkeit; neue Anwendungen und neue Benutzersichten sollen sich genauso wenig wie neue Hardware und Speicherzugriffsorganisationen auf die Konzeption vorhandener Anwendungen auswirken. Daher wird zwischen Anwender- und Realisierungsebene eine Zwischenebene eingezogen, das sogenannte konzeptionelle Schema.

Konzeptionelles Schema (conceptual scheme)

Das konzeptionelle Schema soll für die Gesamtorganisation ein hard- und softwareunabhängiges Modell der Informationswelt darstellen und nach bestimmten Regeln in einem spezifischen Formalismus in einem Datenmodell organisieren.

"Jede Informationseinheit der Unternehmung ist (a) aus Unternehmensgesamtsicht (semantische Dimension), (b) aus der Sicht der Datenspeicherung (syntaktische Dimension) und (c) aus der Sicht der Anwendungen bzw. Benutzer (pragmatische Dimension) zu verwalten." (Ortner/Söllner (1989), S.86).

Das konzeptionelle Schema ist das Verzeichnis der relevanten Informationsobjekte eines Unternehmens und der Beziehungen zwischen den Objekten. Es ist zum einen datenneutral zum anderen datenunabhängig. Datenneutral heißt neutral gegenüber der Sicht einzelner Anwendungssysteme und hat unternehmensweite Gültigkeit. Es wird datenunabhängig aufgebaut, das heißt unabhängig von der Verteilung der Daten auf den Rechnern des Unternehmens, von der eingesetzten Datenverwaltungssoftware, den Betriebs- und Kommunikationssystemen etc. (vgl. Ortner/ Söllner (1989),S.85). Durch seine Datenneutralität und Datenunabhängigkeit stellt das konzeptuelle Schema den ruhenden Pol der DV-Systemarchitektur

eines Unternehmens dar, der sich nur ändert, wenn sich das Umsystem oder die strategische Zielsetzung einer Unternehmung entscheidend verändert.

Ergänzend zur ursprünglichen ANSI-SPARC-Architektur wird das konzeptionelle Schema in der Literatur zunehmend in zwei Subschemata unterteilt (vgl. Lockemann/Rademacher (1990); Schlageter/Stucky (1983), S. 42f.))

Konzeptuelles Schema	**Semantisches Subschema** - unabhängig von konkreten Datenmodellen
	Logisches Subschema - mit Bezug zu bestimmten Datenmodellen

Abb. 2.1/1: Unterteilung des konzeptuellen Schemas

Das semantische Subschema (oft auch semantisches Datenmodell genannt) beschreibt in einem geeigneten Formalismus den zu modellierenden Bereich der Wirklichkeit unabhängig von den Eigenschaften spezifischer Datenmodelle. Daraus wird das logische Subschema abgeleitet (oft auch logisches Datenmodell genannt), das auf die Anwendung eines konkreten Datenmodells und teilweise eines speziellen Datenbanksystems abstellt. Zwischen semantischen, formalisierten Subschema und logischem Subschema bestehen Abbildungsregeln, die der Entwickler mehr oder weniger mechanisch verwenden kann.

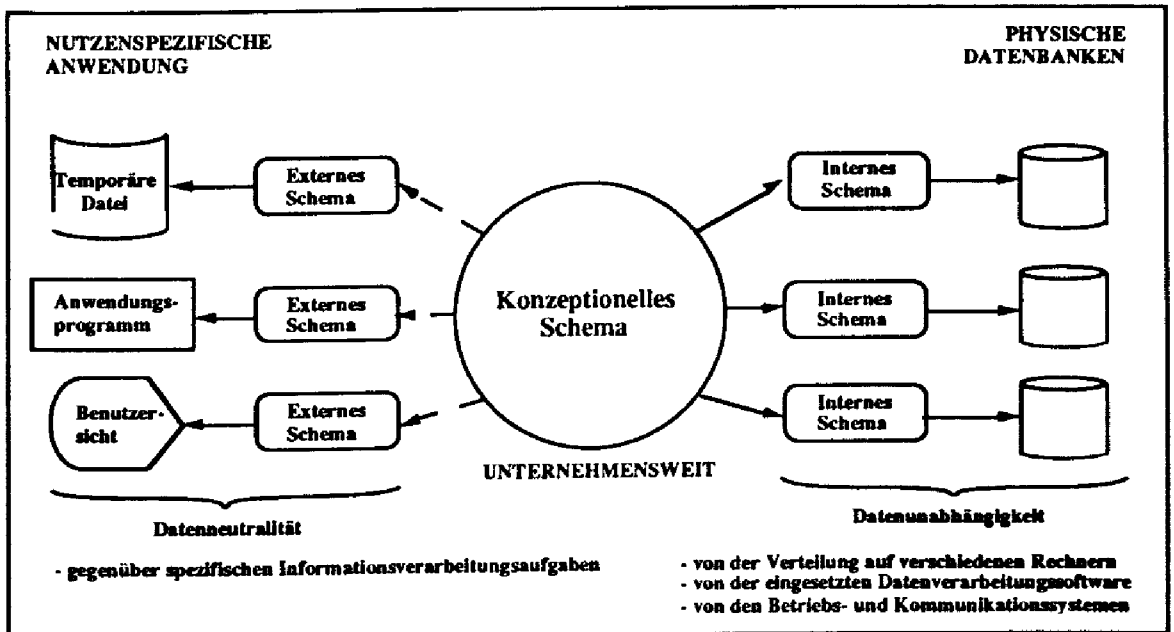


Abb. 2.1/2: 3-Schema Architektur nach ANSI/ SPARC (vgl. Ortner/Söllner (1989))

Internes Schema (internal scheme)

Das interne Schema beschreibt die logische und physische Speicherungsstruktur der Daten des konzeptuellen Schemas mit Hilfe der eingesetzten Datenbanksysteme und Speichermedien. Es enthält zum Beispiel Betriebssystem-Dateibeschreibungen, Schemabeschreibungen einer speziellen Datenbank, Angaben über Zugriffspfade und Abspeicherungsmodalitäten.

Während des Lebenszyklusses eines konzeptuellen Schemas können im Zuge der technologischen Entwicklung durchaus mehrere interne Schemata existieren; zu einem Zeitpunkt existiert jedoch nur ein internes Schema.

Externes Schema (external scheme)

Das externe Schema dient der Präsentation der Daten in einer logischen (externen) Datenstruktur gegenüber einem Nutzer. Seine Darstellung ist zum einen inhaltlich, zum anderen in der Darstellung abhängig vom jeweiligen Informationsbedarf der in der Regel vielen und heterogenen Nutzer (Menschen und Programme). Das externe Schema kann entsprechend der Kategorien der Informationssystem-Architektur

-> technologisch
-> funktional
-> organisatorisch

betrachtet werden.

Technologisch sind die Daten nur temporär, d.h. zur Ausführungszeit der Programme oder zu Zeiten des Zugriffs menschlicher Nutzer in dieser Struktur vorhanden.

Während das konzeptionelle Schema die Objekte und Beziehungen der zu entwerfenden Datenbank spezifiziert, beschreibt das externe Schema in funktionaler Hinsicht die Sichten nach Funktionen und nach Hierarchien abgestufter Benutzergruppen, um

- Subschema zu isolieren, die unabhängig vom globalen konzeptionellen Schema geändert und modifiziert werden können sowie

- spezielle Beschreibungs- und Handhabungssprachen für die Subschemata zu verwenden.

Organisatorisch verknüpft das externe Schema das unabhängig von bestimmten aufbau- und ablauforganisatorischen Gesichtspunkten gestaltete (konzeptuelle und interne) Datenmodell mit der Aufbau- und Ablauforganisation des Unternehmens.

Grundsätzlich haben Programme und Benutzer nur über das externe Schema Zugang zur Datenbasis. Dabei können Programm- bzw. Nutzergruppen durchaus identische externe Schemata verwenden.

2.2. Phasengliederung

Phasenmodelle ergänzen die 3-Schema-Sicht um Schritte, die beim Entwurf einer Datenbank zu durchlaufen sind. Scheer (1988) unterteilt den Entwurf der sachlogischen Datenstrukturen in die Konstruktions- und Modellierungsphase. Die so entstandenen Datenstrukturen werden dann in Datenmodelle (z. B. Relationenmodell, Netzwerkmodell) überführt und in Datenbanksysteme implementiert.

Schritte			
Konstruktion	**Modellierung**	**Schemabildung**	**Implementierung**
(Semantische Datenmodellierung)	(Logische Datenmodellierung)	(Logische Datenimplementierung)	(Physische Datenmodellierung)
Instrumente			
Konstruktions-hilfsmittel z. B. E-R-M	Datenmodelle z.B. Netzwerkmodelle	Normalformenlehre nach Codd	Datenbanksystem
Ergebnis			
Semantisches Datenmodell z. B. - Entity Relationship-Modell	Logisches Datenmodell z. B. - Relationales Modell	Datenbankschema z.B. - normalisiertes Datenmodell	Datenbank-implementierng z.B. - DB2-Datenbank

Abb. 2.2/1: Phasen des Datenbankentwurfs

Der Entwurfsprozeß soll in vier Phasen eingeteilt werden:

In der Konstruktionsphase werden die Datenstrukturen aufgrund der unternehmerischen Erfordernisse aus betriebswirtschaftlicher Sicht entworfen und mit Hilfe einer formalen Beschreibungssprache dokumentiert (fachliche Konstruktion).

Im Modellierungsprozeß werden die Datenstrukturen aus Sicht der Datenverarbeitung eindeutig

- formal in ihren semantischen und syntaktischen Strukturen
- materiell in ihren quantitativen (Mengengerüst) und qualitativen (regionalen, organisatorischen) Anforderungen gekennzeichnet,

um die Erfordernisse an die einzusetzende Hard- und Software abzuschätzen. Ziel ist dabei besonders, das logische Datenmodell zu beschreiben und die grundsätzlichen technologischen Anforderungen an das einzusetzende Datenbanksystem herauszuarbeiten.

Der Prozeß der Schemabildung hat dann das Ziel, die technische oder organisatorische Performance des zu implementierenden Datenbanksystems im Hinblick auf

-> die Speicherplatzausnutzung
-> das Zugriffsverhalten
-> das Kommunikationsverhalten

unter Beachtung formal-logischer und ökonomischer Gesichtspunkte zu optimieren.

Im letzten Schritt wird dann das logische Schema in eine implementierbare und effiziente technische Struktur vor dem Hintergrund eines spezifischen Datenbanksystems umgesetzt.

Die folgende Abbildung zeigt, wie ein Phasenmodell und das ANSI-SPARC-3-Schema-Modell zu verbinden sind:

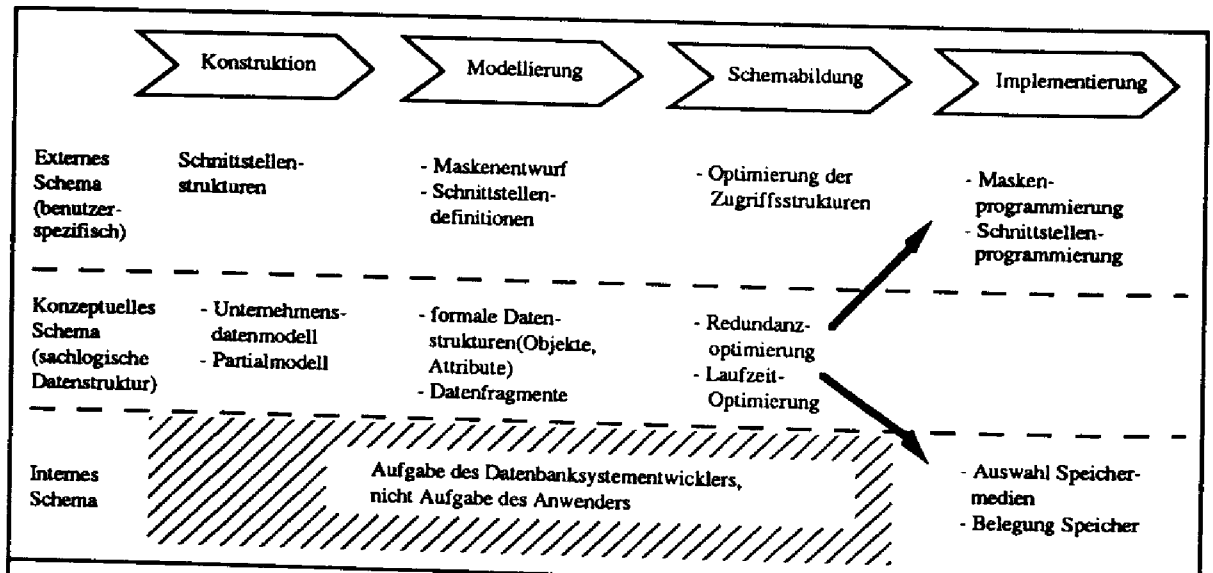


Abb. 2.2/2: Verbindung des Phasenmodells und des ANSI-SPARC-3-Schema-Modells

Während der Konstruktionsphase ist zunächst das konzeptuelle Schema der Datenbank zu entwerfen. Bei einem totalen System-Neuentwurf mündet dieses in einem umfassenden Unternehmensdatenmodell, das sämtliche Datenelemente, deren Beziehungen und die zulässigen sachlogischen Transaktionen kennzeichnet. Bei den üblichen Überarbeitungen oder Ergänzungen vorhandener DV-Systeme entsteht ein Partial-Datenmodell für den aktualisierten Bereich, dessen Schnittstellen zur Systemumgebung im externen Schema spezifiziert werden. In der Konstruktionsphase arbeiten vornehmlich Systemanalytiker (mit Datenmodellierungskenntnissen) mit den Mitarbeitern der Fachabteilungen zusammen, die Einschaltung von Datenbankfachleuten ist nur bei Spezialfragen notwendig. Die Datenbankspezialisten setzen dann anschließend in der Modellierungsphase das semantische in ein logisches Datenmodell um. Beispielsweise entwickeln sie auf der Grundlage des relationalen Datenmodells (vgl. Abschnitt 2.4) jetzt eine Tabellen-Datenstruktur, spezifizieren den Aufbau der Datenobjekte und die zulässigen Datenoperatoren.

Sowohl das konzeptuelle als auch das externe Schema sind aus statischer wie auch aus dynamischer Sicht zu betrachten. Die statische Sicht beschreibt die Datenobjekte und die Datenbeziehungen mit ihren jeweiligen Attributen. Die dynamische Sicht wird gekennzeichnet durch die Datenflüsse und die aufgrund der Datenstrukturen und -flüsse möglichen Systemzustände.

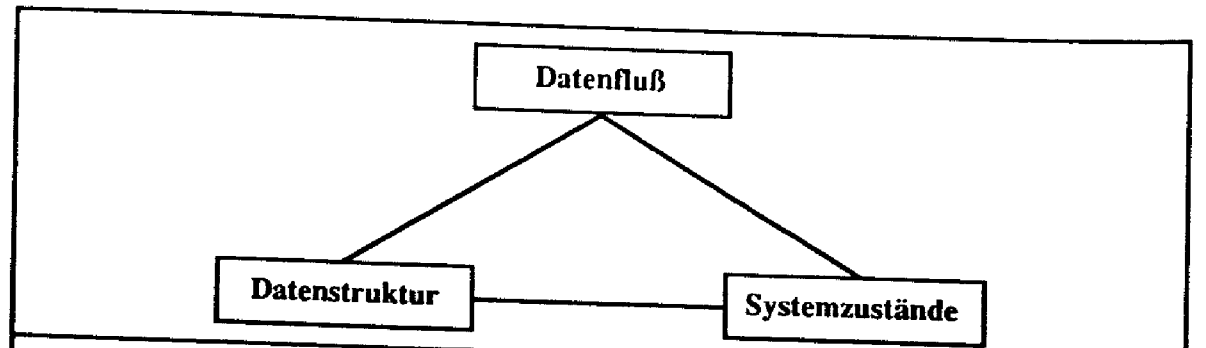


Abb. 2.2/3: Elemente eines Datenmodells (nach Marco/Yourdon(1986))

Datenflüsse werden durch Transaktionen beschrieben. Im externen Schema werden die strukturellen und verhaltensmäßigen Anforderungen an Abfragen und Reports definiert, die auf die Datenbank zugreifen. Auf der konzeptuellen Ebene werden die Anforderungen der

Transaktionen verallgemeinert und die Struktur und die Aktionen jedes Datenobjekts des Anwendungssystems definiert.

	statisch	**dynamisch**
Konzeptuelles Schema	sachlogische Datenstruktur	sachlogische Transaktionen
Externes Schema	sachlogische Nutzerschnitt-stellen/-oberflächen	strukturelle Transaktionen
Internes Schema	Speicherstrukturen, Zugriffspfade	Zugriffstransaktionen, Integritätsprozeduren

Abb. 2.2/4: Statische und dynamische Sicht des konzeptuellen, externen und internen Schemas

Auf der Datenbankebene werden schließlich die Speicherungsstrukturen, die Zugriffspfade und die zulässigen Transaktionen und ihre Realisierung im System beschrieben.

3. Datenkonstruktion (= semantische Datenmodellierung)

3.1. Kennzeichnung des Konstruktionsprozesses

Der Konstruktionsprozeß, häufig semantische Datenmodellierung genannt, soll den relevanten Ausschnitt der realen, empirisch beobachtbaren Objektwelt mit einem system-analytischen Vorgehen durchleuchten und in einem abstrakten Modell abbilden. Dabei wird grundsätzlich ausgegangen von

- der Unternehmenspolitik, deren Zielen und Aufgaben,
- der Analyse der güter-, finanz- und informationswirtschaftlichen Leistungsprozesse,
- den Interdependenzen mit der Unternehmensumwelt und den Koalitionsteilnehmern der Unternehmen,
- den heutigen und zukünftigen Informationsbedürfnissen der verschiedenen Management-ebenen,
- den soziopsychologischen Randbedingungen der Informationsgenerierung,
- der angestrebten Leistungsfähigkeit des DV-gestützten Informationssystems und des daraus resultierenden Ressourcenbedarfs.

Der Konstruktionsprozeß soll bewußt unabhängig von dem Möglichkeiten bestimmter Datenmodelle oder Datenbanksysteme gestaltet werden. Ziel ist die umfassende Analyse des betriebswirtschaftlichen Problems und dessen Beschreibung in den Kategorien eines Konstruktionshilfsmittels. Ein gebräuchliches Hilfsmittel ist das Entity-Relationship-Modell, das als Kategorien Datenobjekte, Datenbeziehungen und Attribute unterscheidet.

Die Konstruktionshilfsmittel stellen sicher, daß sich das Ergebnis des Konstruktions-prozesses ohne Bruch in ein logisches Datenmodell überführen und letztlich in einem Daten-banksystem implementieren läßt. Sie schlagen somit eine Brücke zwischen der Analyse des Realitätsausschnittes mit den betriebswirtschaftlichen Fachbegriffen und der Realisierung in einem logischen Datenmodell mit den Kategorien der Informatik.

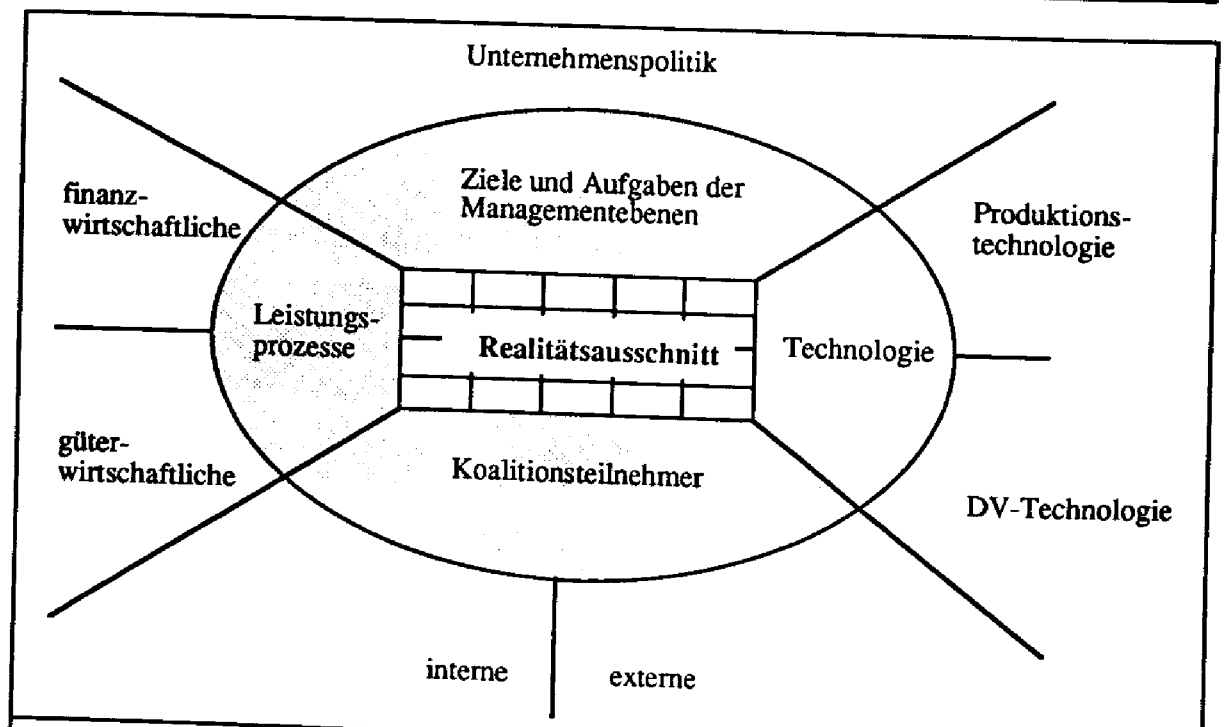


Abb. 3.1/1: Analyseraster bei der Durchleuchtung und Abgrenzung des Realitätsausschnittes

Ziel der Konstruktion ist es, die verwirrende Vielfalt der Objektwelt in einem einheitlich und stringent gestalteten "abstrakten Modell" zu erfassen. Geleitet wird der Konstruktionsprozeß von einer "Weltsicht" des Konstrukteurs, unterstützt wird er durch formale Konstruktionsoperatoren, begleitet wird er von korrespondierenden Konstruktionshilfsmitteln.

Im Konstruktionsprozeß werden die Datenstrukturen aus den betriebswirtschaftlichen Zusammenhängen und Erfordernissen heraus entworfen. Dabei wirkt der Datenbankkonstrukteur mit seinem „Weltbild" mit dem fachlichen Bild der Fachabteilungen zusammen. Dieser Schritt ist entscheidend für den Erfolg des zu entwerfenden Daten- und Anwendungssystems (vgl. Scheer (1988b), S. 15).

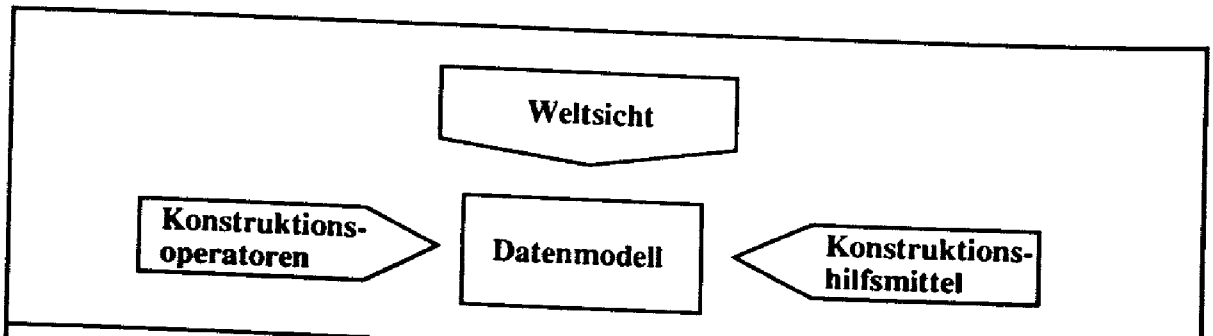


Abb. 3.1/2: Konstruktions-Weltsicht als Ausgangspunkt für den Konstruktionsprozeß

3.1.1. Konstruktionsweltsicht und Sprachebenen

Die übliche Konstruktionsweltsicht unterscheidet folgende Kategorien (vgl. Bubenko, (1977))

(1) Objekte (entities)
(2) Eigenschaften (properties) der Objekte
(3) Beziehungen zwischen Objekten (associations)
(4) Eigenschaften der Beziehungen
(5) Ereignisse (events) systeminterner oder externer Art
(6) Eigenschaften der Ereignisse (attributes)
(7) Gültigkeiten (assertion)
(8) Folgerungen (conclusions)
(9) Zeit (time)

Objekte kennzeichnen reale Objekte der Wirklichkeit, beispielsweise Personen, Gebäude, Maschinen etc. Diese Objekte lassen sich konkret identifizieren und in aller Regel durch „Substantive" eindeutig bezeichnen.

Beziehungen zwischen Objekten kennzeichnen Aktivitäten, die zwischen den realen Objekten bestehen. Diese lassen sich in der Regel durch Verben kennzeichnen.

Beispiele sind *„Auftrag 4711 belegt Maschine Extruder A"* oder *„BALSAM AG kauft Plastikgranulat, rot"*.

„Objekte" und „Beziehungen" haben bestimmte Eigenschaften, die diese identifizieren und ihre Struktur kennzeichnen. So hat beispielsweise das Objekt „Extruder A" neben der Bezeichnung sicher auch noch eine identifizierende Nummer und wird charakterisiert durch seine wirtschaftliche und technische Kapazität. Die Beziehung „belegt" durch einen bestimmten Auftrag wird durch Eigenschaften, wie beispielsweise den Zeitraum der Belegung, spezifiziert.

Wichtig ist die Verwendung einer sehr konkreten „Sprachebene". Im Laufe des Konstruktionsprozesses werden die Erkenntnisse verallgemeinert, es wird ein höheres sprachliches Abstraktionsniveau gewählt. Die heute verbreiteten Konstruktionshilfsmittel setzen dabei auf der Typebene ein.

Ereignisse mit bestimmten Attributen wirken im Zeitablauf auf Objekte oder Beziehungen ein und verändern deren Eigenschaften. Die Gültigkeit gibt an, ob eine bestimmte Beziehung zu einem spezifizierten Zeitpunkt existiert oder nicht. Folgerungen beschreiben logische Regeln, die aus bestimmten Ereignissen spezifische Gültigkeiten ableiten.

Die Zeit spielt eine zentrale Rolle bei der Konstruktion. Eine entsprechende Modellierungsvorschrift wird benötig, um

- Zeitintervalle kennzeichnen zu können,
- das Auftreten von Ereignissen beschreiben zu können,
- die Existenzbedingungen für Objekte definieren zu können.

Üblicherweise werden die ersten vier Kategorien in der statischen Datenkonstruktion betrachtet, die restlichen drei Kategorien unterstützen dann die Konstruktion des dynamischen Datenmodells.

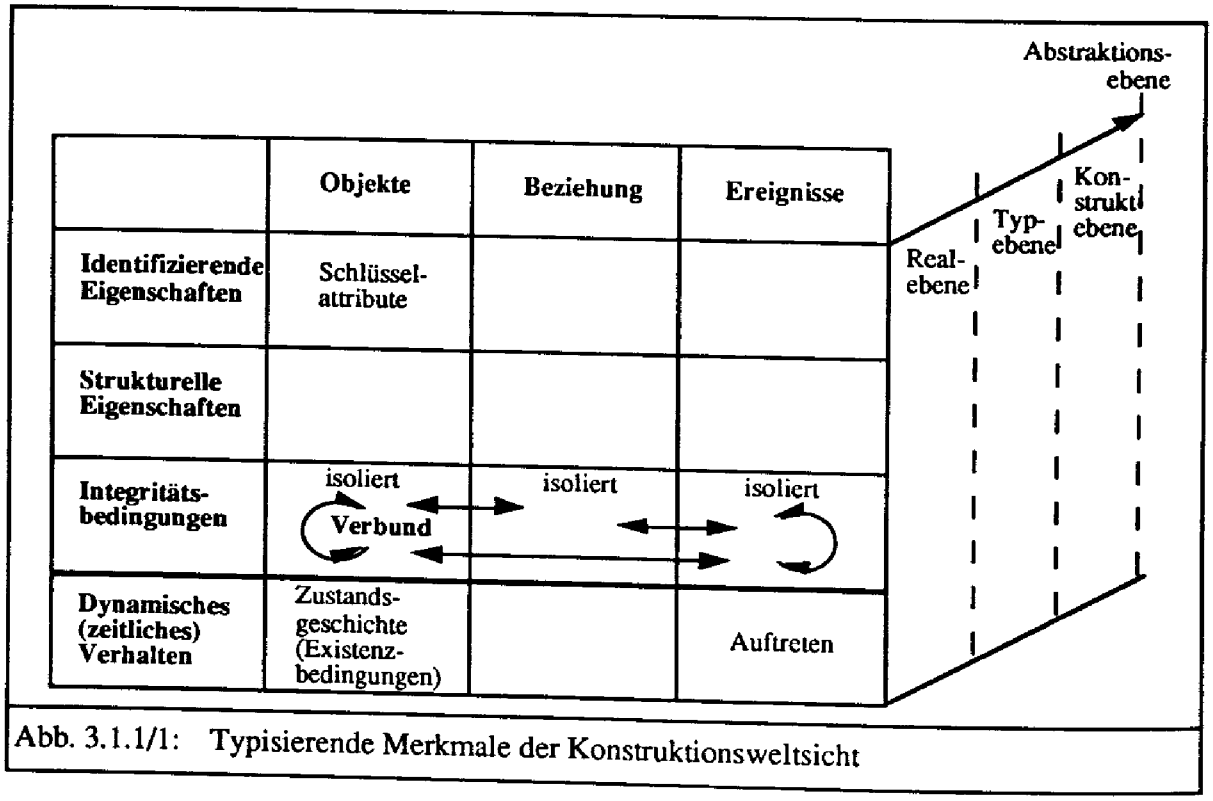


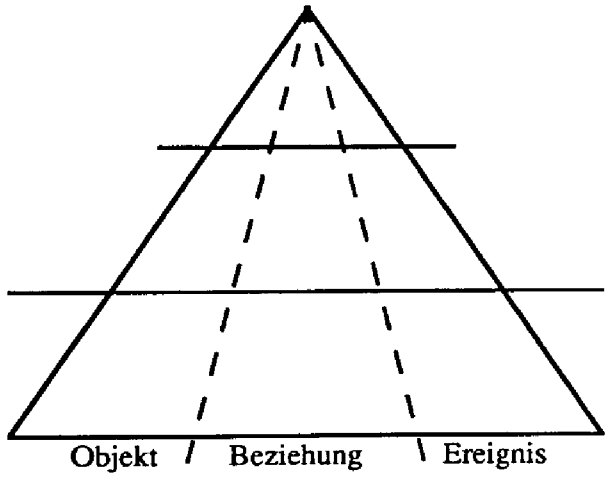
Abb. 3.1.1/1: Typisierende Merkmale der Konstruktionsweltsicht

Abstraktionsebenen

Zum Konstruktionsprozeß gehören bestimmte Abstraktionsebenen. Wedekind/Ortner ((1980), S.10) schreiben dazu : "Zum Wesen des systematischen Konstruierens gehört die Einführung von Abstraktionsebenen. Eine Abstraktionsebene ist die Zusammenfassung von genau festgelegten Aufgaben. Da die verschiedenen Aufgaben sich gegenseitig bedingen, müssen die Abstraktionsebenen in eine Ordnung gebracht werden."

Die Abstraktion ermöglicht es dem Konstrukteur, Gemeinsamkeiten und strukturelle Unstimmigkeiten in der Realitätsbeobachtung zu entdecken und letztlich einfachere und stimmigere Systeme zu entwickeln.

Zu unterscheiden sind Begriffe, die die Struktur der Konstruktionsweltsicht kennzeichnen von Begriffen, die das Abstraktionsniveau kennzeichnen. Üblicherweise wird dabei die Realebene von der Typebene und Konstruktebene unterschieden.



Konstruktebene
= abstrakte "metasprachliche" Objekte der betrachteten Wirklichkeit

Typebene (Klasse)
= Gruppe gleichartiger realer Sachverhalte der betrachteten Wirklichkeit

Realebene
= reale Tatbestände der betrachteten Wirklichkeit

Abb. 3.1.1/2: Strukturelemente und Abstraktionsebenen der semantischen Datenmodellierung

Reale Elemente sind in der Wirklichkeit zu beobachten und durch den Konstrukteur den Strukturen seiner Weltsicht zuzuweisen. Der Konstrukteur entscheidet zum Beispiel, ob ein „Auftragsschreiben" als juristischer Begriff und ablauforganisatorische Aufgabe als Datenobjekt (d.h. als Sache) oder als Datenbeziehung (d. h. als Aktivität „kaufen") oder vielleicht auch als Ereignis im Datenmodell erfaßt wird. Typelemente werden durch Gruppierung realer Elemente mit gleichen Eigenschaften gebildet, während Konstruktelemente durch Zusammenfassung (Kombination) von Typelementen gebildet werden.

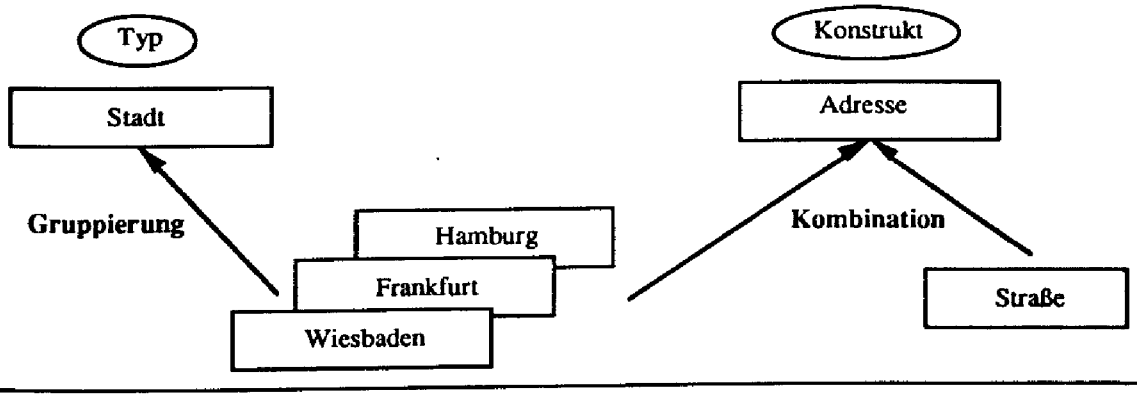
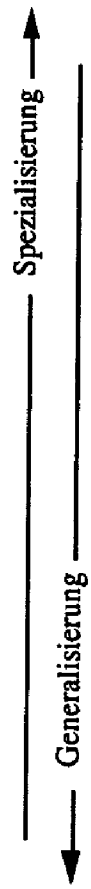


Abb. 3.1.1/3: Gruppierung und Kombination von Elementen

Betriebswirtschaftlich sind bei der Abstraktion jeweils die Mengen- und die Wertbetrachtung zu unterscheiden.



	Mengenbetrachtung	Wertbetrachtung
Realebene	1. "Fa. Max Müller" 2. Packung Persil 3. Verpackungsmachine Füllo 1000	1. Zahlung an Max Müller über 1000,- DM
Typebene	1. Kunde, Lieferant 2. Artikel, Einkaufsteil 3. Betriebsmittel, Werkzeuge, Mitarbeiter	1. Konto
Konstruktebene	1. Marktpartner 2. Stoff 3. Ressourcen	1. Kontenklasse 2. Kostenträger 3. Kostenstelle

Abb. 3.1.1/4: Verdeutlichung der Real-, Typ- und Konstruktebene

Sprachebenen

Nach Abgrenzung und Analyse des Realitätsausschnittes sind möglichst gehaltvolle und relevante Aussagen aus dem Anwendungsbereich zu gewinnen. Dabei muß es das Ziel sein, die oft vagen Aussagen der Führungskräfte und Mitarbeiter der Fachabteilungen sprachlich zu präzisieren und auf „sprachlich normierte Formulierungen" zurückzuführen, die dann in einer späteren Phase in das „Data Dictionary" einfließen können.

	verbal	**graphisch**	**mathematisch**
Umgangssprache	Briefe, Protokolle	Skizzen	
Fachsprache	Betriebswirtschaftliche Handbücher	Organigramme	Kalkulationsschemata
DV-Sprache	Data Dictionaries	SADT-Diagramm	Relationenalgebra

Abb. 3.1.1/5: Sprachebenen bei der Datenmodellierung (Beispiele)

Konstruktionsphasen

Die Konstruktion hat das Ziel, die von den Fachabteilungen verwendeten Begriffe inhaltlich zu klären, semantisch eindeutig zu definieren und in eindeutigen Objekten/Relationen festzuhalten.
Dazu sind folgende Schritte durchzuführen:

(1) Eindeutige, unternehmensweit abgestimmte Begriffe und Definitionen entwickeln

Die Fachabteilungen verwenden heutzutage noch eine Vielzahl individueller, nicht unternehmensweit abgestimmter Begriffe. Bei deren Vereinheitlichung muß zunächst einmal aufgespürt werden, wo in verschiedenen Anwendungen der Organisationseinheiten Gegenstände, Eigenschaften und Beziehungen unter unterschiedlichen Bezeichnungen, aber mit annähernd identischen Inhalten verwendet werden. (vgl. Vetter, S.282ff; Ortner/Söllner (1989); Ortner/Rössner/Söllner (1990))

Dieser Prozeß wird als „Integration unterschiedlicher externer Schemata" bezeichnet. Dies ist eine vornehmlich fachlich-betriebswirtschaftliche Aufgabe. Als solche ist sie von einer betriebswirtschaftlichen Fachabteilung verantwortlich durchzuführen; das Controlling in den deutschen Unternehmen bietet sich aufgrund der Fachkompetenz und seiner Zuständigkeiten dafür an.

Instrumente dieser betriebswirtschaftlichen Begriffsvereinheitlichung sind

-> Begriffskataloge, die präzise betriebliche Begriffe definieren und unternehmensweit (d. h. gegebenenfalls auch mehrsprachig) normieren
-> Handbücher, die den Verwendungszusammenhang dieser Begriffe präzise beschreiben und über ihre Gestaltungsfunktion die externen Schemata der Fachabteilungen aufeinander abstimmen.

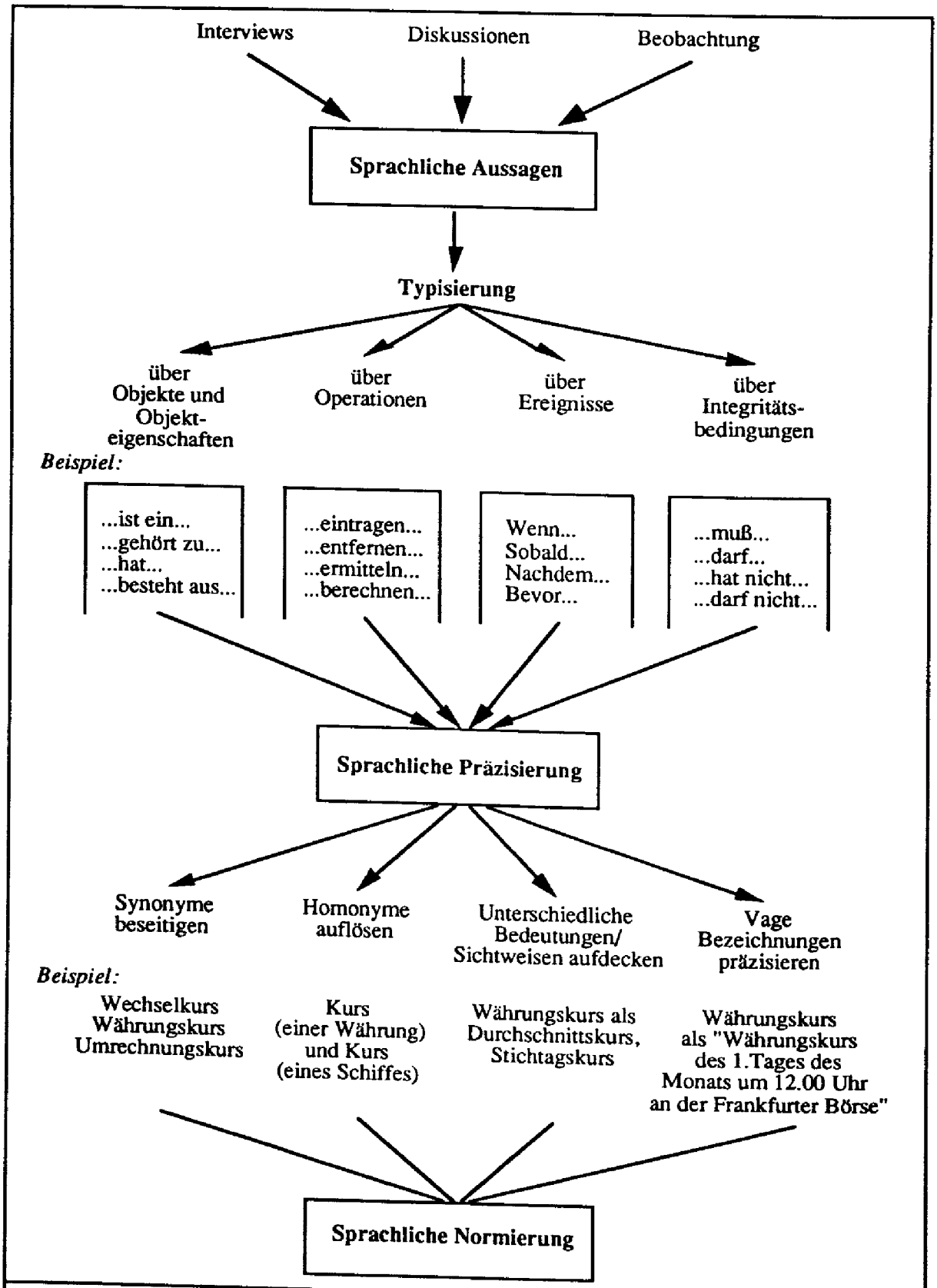


Abb. 3.1.1/6: Sprachliche Präzisierung als Schritt des Konstruktionsprozesses (vgl. Ortner/Söllner (1989), S. 36ff.)

(2) Bestimmung der typmäßigen Datenobjekte

Die Konstruktion der Datenobjekte und Datenbeziehungen aus den vereinheitlichten Fachbegriffen erfolgt mit Hilfe von Konstruktionsoperatoren, die die Begriffsstrukturen der Fachabteilungen auf der Typebene verallgemeinern. Dabei wird durch kluge Anwendung der Konstruktionsregeln versucht, möglichst wenige Objekttypen und Beziehungstypen zu bilden. In der Feindatenkonstruktion werden die Objekte und Beziehungen um die Attribute ergänzt.

Es erfolgt eine Integration des neuen Datenmodells in evtl. bereits existierende Datenmodelle. Das so entstehende semantische Datenmodell wird mit graphischen Modellierungssprachen (wie dem in der Folge beschriebenen Entity-Relationship-Modell) beschrieben und unter Verwendung spezieller datenbankgestützter Dokumentationssysteme (sogenannter Data Dictionaries) möglichst detailliert dokumentiert. Es stellt den Ordnungsrahmen für die weitere DV-technische und DV-organisatorische Entwicklung der Anwendungssysteme dar.

(3) Dokumentation der Datenobjekte und der Begriffe in einem Datenlexikon

Die betriebswirtschaftlichen Begriffssysteme sowie die daraus abgeleitete logische Datenstruktur wird in der nächsten Stufe in DV-Hilfsmitteln, z. B. DATA DICTIONARIES, dokumentiert. Solche DV-gestützten Begriffskataloge bieten darüberhinaus vielfältige Hilfen z. B. bei der Suche nach synonymen oder homonymen Begriffen, an.

3.1.2. Konstruktionshilfsmittel

Konstruktionshilfsmittel sind unabhängig von bestimmten Datenmodellen ausgelegt, erlauben also beispielsweise sowohl die Konstruktion eines Datenmodells für ein hierarchisches als auch für ein relationales Datenmodell.

Konstruktionshilfsmittel sollen

- die Kommunikation zwischen Anwendern (sowie Fachabteilung und DB-Entwurfsspezialisten) erleichtern,
- damit eine Brücke zwischen Fachsprache und "DV-Kauderwelsch" schlagen,
- die reale Datenwelt inhaltlich vollständig und logisch konsistent erfassen.

Methodisch bestehen die existierenden Ansätze aus 3 Teilen:

(1) einer Konstruktionsweltsicht
(2) einer strukturierten verbalen Beschreibungssprache
(3) einer graphischen Beschreibungssprache

Wie schon erwähnt, basiert die verbreitete **Konstruktionsweltsicht** auf der Systemtheorie und kennt aus statischer Sicht

- Datenobjekte
- Beziehungen zwischen den Datenobjekten
- Eigenschaften von Datenobjekten und Beziehungen

Entsprechende Konstruktionshilfsmittel werden als „Gegenstands-Beziehungs-Modelle" oder englisch als „Entity-Relationship-Models" bezeichnet. Sie unterscheiden sich im wesentlichen hinsichtlich der Art der unterstützten Beziehungen und in der Detaillierung des angebotenen Begriffssystems.

Im Grundsatz sind die Hilfsmittel gleichwertig, so daß es bei der Auswahl vor allem auf die Verbreitung und Standardisierung im Unternehmen sowie auf die DV-Unterstützung durch CASE-Instrumente ankommt.

Die **strukturierte verbale Beschreibung** charakterisiert die Datenelemente durch identifizierende (z. B. durch Angabe ihres Namens) und beschreibende Eigenschaften.

Üblicherweise unterscheidet man bei den Eigenschaften Attribute und Rollen (role). Eine Attributeigenschaft besteht aus einem bestimmten Wertetyp, der ebenfalls beschrieben werden muß. Der Wertetyp beschreibt die Domäne (einschließlich einer Meßvorschrift, der Meßgenauigkeit) und die zulässigen Operationen.

Eine Rolleneigenschaft beschreibt eine Beziehung (relationship) zwischen dem betrachteten Datenobjekt und anderen Datenobjekten des Informationssystems (vgl. Lindgreen (1983)).

Das verwendete Hilfsmittel zur verbalen Beschreibung sollte bestimmte semantische Strukturen des zugrundeliegenden Objektsystems unterstützen, also beispielsweise die Definition betriebswirtschaftlicher Begriffe, um die Kommunikation mit den Fachabteilungen zu erleichtern.

Angesichts der Vielfalt der sprachlich eindeutig zu kennzeichnenden Sachverhalte, ist eine Rechnerunterstützung bei den vielfältigen Editier- und Analysevorgänge unumgänglich.

Ein Beispiel ist die KWIC-Liste (Keyword in Context) (vgl. Mayr/Dittrich/Lockemann (1990), S. 512). Dieser Ansatz zerlegt die zusammengesetzten Gegenstands-, Attributs- und Beziehungstyp-Bezeichnungen in ihre Morpheme, d.h. in die kleinsten, bedeutungstragenden Sprachelemente und sucht anhand einer entsprechenden Liste nach sprachlichen Unschärfen (z. B. Homonymen (gleiche Bezeichnung für Verschiedenes) und Synonymen (verschiedene Bezeichnungen für Gleiches) und inkonsistenten Bezeichnungen für Attribute und Rollen, die oft auf Ungenauigkeiten im Konstruktionsprozeß hinweisen.

Die **formalisierte graphische Beschreibungssprache** eines Konstruktionshilfsmittels besteht in der Regel aus einem graphischen Konstruktionshilfsmittel sowie einer Umsetzungsanweisung in ein logisches Datenmodell.

Bei den Konstruktionshilfsmitteln kann man phasenorientiert unterscheiden zwischen

> Partialkonzepten, die nur für einzelne Phasen des Datenbankentwurfes geeignet sind, und
> Gesamtkonzepten, die alle Phasen eines Datenbankentwurfs unterstützen.

Phasenorientierte Gesamtkonzepte für alle 4 Teilaufgaben verfügen über eine Konstruktionsweltsicht, aus der dann logisch durchgängig eine Vorschrift zur strukturierten verbalen Beschreibung, eine Symbolik zur graphischen Beschreibung und schließlich zur mathematisch-formalen Beschreibung entwickelt wird.

Schemaübergreifende Gesamtkonzepte unterstützen nicht nur die 4 Phasen des Datenbankentwurfs, sondern bieten integrale Konstruktionshilfsmittel sowohl für den Entwurf des externen, des konzeptuellen und des internen Schemas an.

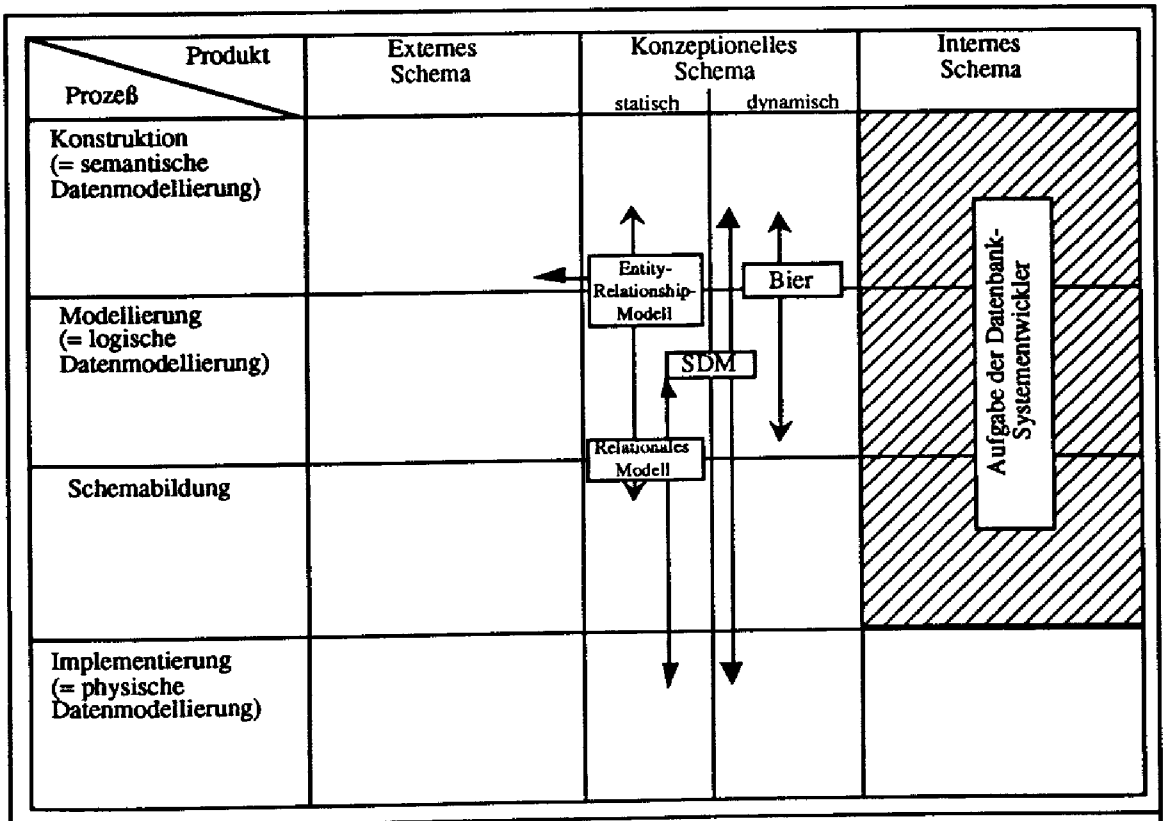


Abb. 3.1.2/1: Übersicht über die Phasen der Datenbankentwicklung und Einordnung wichtiger Konstruktionshilfsmittel

Wie die Abbildung zeigen soll, haben sich Gesamtkonzepte aus ursprünglichen Partialkonzepten entwickelt.

Konstruktionshilfsmittel sind auch danach zu unterscheiden, inwieweit für sie rechnergestützte Hilfsmittel auf dem Markt verfügbar sind und welche Mächtigkeit diese anbieten. Die Mächtigkeit reicht vom

-> rechnergestützten Zeichnen und Arrangieren z. B. von Datenobjekt-Beziehungs-Diagrammen,

-> über die Sicherung der logischen Integrität von Teil-Datenmodellen unterschiedlicher Funktionsbereiche (horizontale Integrität) oder von Datenmodellen verschiedenartiger Detaillierungsstufen (vertikale Integrität)

-> bis hin zur automatisierten und algorithmierten Anwendung bestimmter Konstruktionsoperatoren.

Die meisten Hersteller von Datenbanksystemen bieten für ihre Produkte unterstützende Hilfsmittel an, die jedoch in aller Regel nur die logische Datenmodellierung, die Schemabildung und die physische Modellierung unterstützen. Die verbreiteten CASE-Werkzeuge wie „IEW - Industrial Engineering Workbench" und „Teamwork" bieten eine Datenbankentwurfskomponente an, die meist bis zur Sicherung der logischen Integrität reicht.

3.2. Aufgaben des Konstruktionsprozesses

Der Konstruktionsprozeß kann in vier Aktivitäten aufgeteilt werden:

1. Konstruktion der statischen Struktur (structural properties)
2. Konstruktion der dynamischen Struktur (behavioral properties)
3. Integration beider Datenstrukturen
4. Dokumentation der Datenstruktur

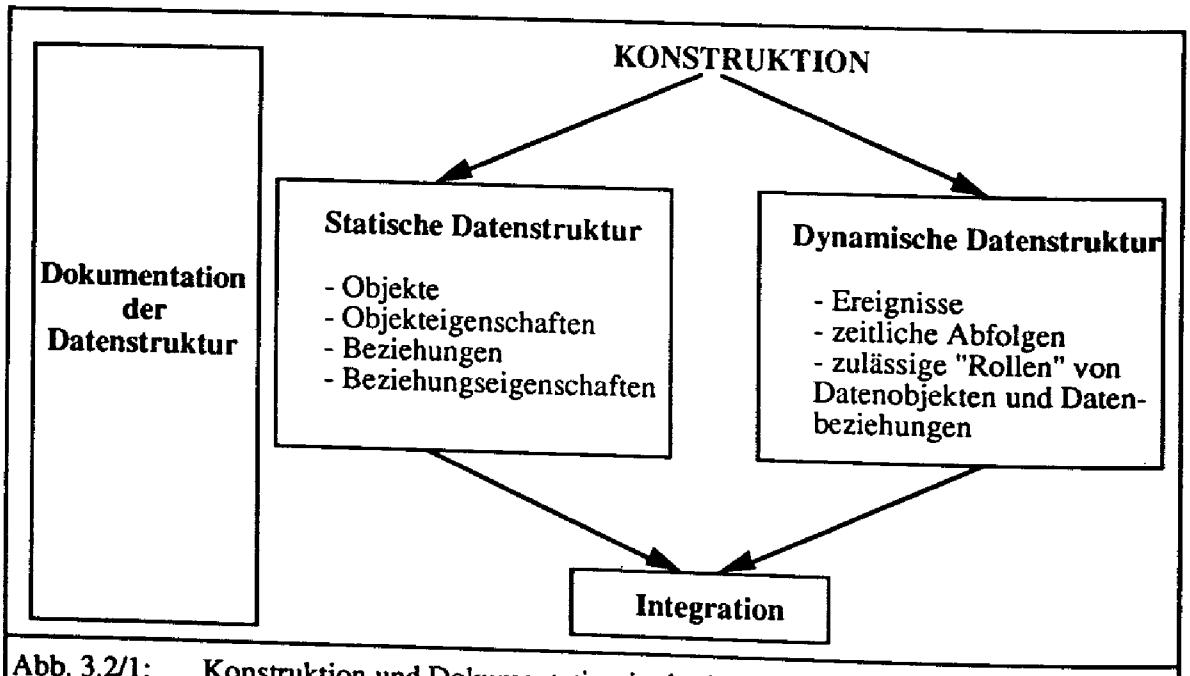


Abb. 3.2/1: Konstruktion und Dokumentation in der Konstruktionsphase

Im Rahmen der statischen Konstruktion sind die interessierenden Informationsbestände der Realität mit den relevanten Eigenschaften zu selektieren und in Datenobjekten zu beschreiben. Dies kann einerseits durch strukturelle Beschreibungen, andererseits durch die Angabe zusätzlich einzuhaltender Regeln (Konsistenzbedingungen) erfolgen. In der dynamischen Konstruktion sind die Operationen zu beschreiben, die auf Datenobjekte und deren Eigenschaften zulässig sind. Die Erkenntnisse der drei Phasen sind möglichst begleitend zu dokumentieren.

3.2.1. Statische Konstruktion

3.2.1.1. Kennzeichen, Ziele, Vorgehen

Kennzeichen

Die statische Konstruktion orientiert sich zum einen an den bereits im Unternehmen vorhandenen Datenstrukturen, zum anderen an den Ergebnissen der vorgelagerten Informationsbedarfsanalyse. Die vorhandenen Datenstrukturen sind aus den manuellen oder maschinellen Informationssystemen abzuleiten. Quellen dieser Analyse können zum Beispiele Karteikarten, Formulare, Dateistrukturen oder Eingabemasken sein. Die Datenobjekte werden darin üblicherweise mit Begriffen gekennzeichnet. Ausgehend von diesen im Unternehmen eingeführten Begriffen werden durch die Anwendung der (weiter oben beschriebenen) Konstruktionsoperatoren zunächst auf rein fachlicher (hier betriebswirtschaftlicher) Ebene neue Begriffe erzeugt und eindeutig definiert (Analyse der fachlichen Datenstrukturen).

Auf der strategischen Ebene war es das Ziel der vorgelagerten Informationsbedarfsanalyse die Informationen zu identifizieren, die dem Unternehmen im Markt Wettbewerbsvorteile verschaffen und die zur Zeit nur teilweise existieren (vgl. Ortner (1985), Ortner/Söllner (1989), S.83). Diese Informationen sind während der Datenkonstruktion zu konkretisieren; es sind die notwendigen Attribute und Verfahren zu bestimmen.

Primäres Ziel der statischen Datenkonstruktion ist die begrifflich eindeutige Bezeichnung von heute oder zukünftig relevanten (empirischen) Sachverhalten. Dabei sind alle bisher existierenden Fachbegriffe zu klären, Ungenauigkeiten und Mehrfachbegriffe zu bereinigen und somit ein begrifflich eindeutiger und einheitlicher Rahmen für das Zusammenspiel zwischen Fachabteilungen und DV-Abteilungen zu schaffen.

"Ein Datenbestand ist benutzeradäquat organisiert, wenn die Fachbegriffe für Informationsobjekte und deren Eigenschaften eindeutig geklärt, in ihrer Bedeutung und Struktur analysiert und künftig im Gesamtunternehmen einheitlich verwendet werden" (Ortner/ Söllner (1989), S.83).

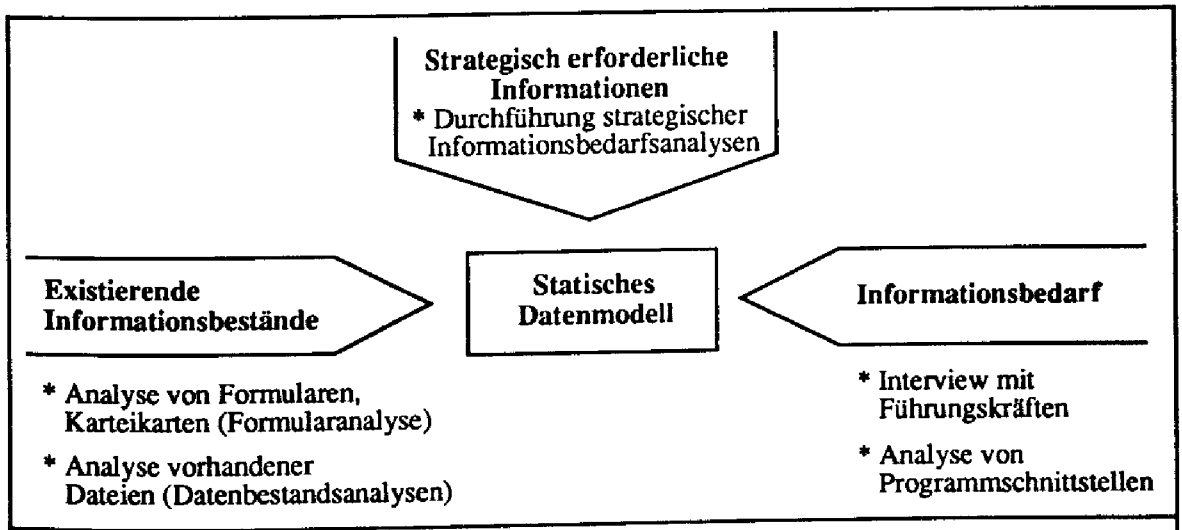


Abb. 3.2.1/1: Informationsbestandsanalyse und Informationsbedarfsanalyse als Ausgangspunkte der Datenkonstruktion

Formales Vorgehen:

Formal setzt jede Begriffsbildung entweder beim Vergleich von Gegenständen bzw. Begriffen (Komparation) und/oder bei der Verbindung von Gegenständen bzw. Begriffen (Kombination) an.

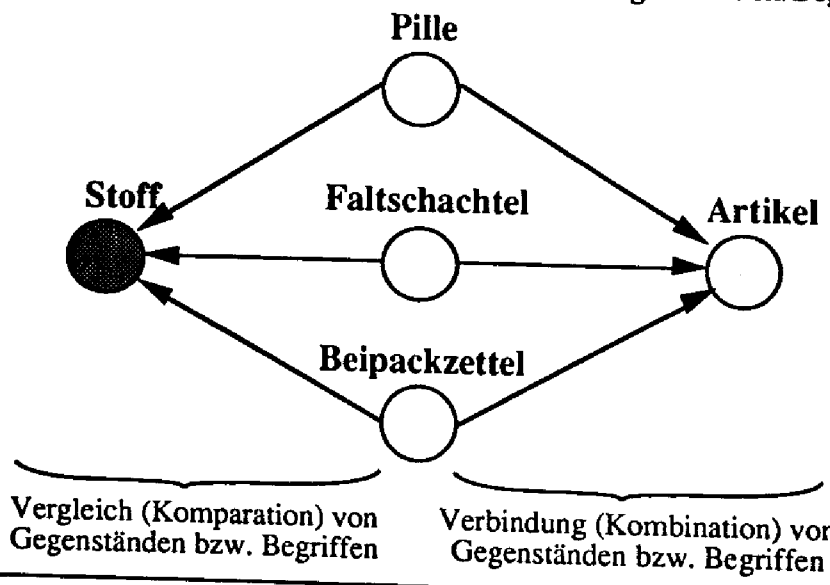


Abb. 3.2.1/2: Gegenüberstellung von Komparation und Kombination (vgl. Ortner (1985), S. 22)

Beim Vergleich wird nach den Gemeinsamkeiten von Gegenständen bzw. Begriffen gesucht, die sich aus deren allgemeinen oder betriebsspezifischen Eigenschaften ergeben.

Beispiel:
Aus betrieblicher Sicht können externe Kunden, Empfänger von Warenproben und in der Kantine essende Mitarbeiter zum Datenobjekt KUNDE zusammengefaßt werden.

Bei der Verbindung der Begriffe entstehen aus der Zusammenfassung von realen Gegenständen neue begriffliche Konstrukte.

Beispiel:
Die Durchführung eines spezifischen Bearbeitungsschrittes an einem bestimmten Material mit Hilfe bestimmte Maschinen kann als PROZESS bezeichnet werden.

Inhaltliches Vorgehen

Für diesen ersten Schritt der eigentlichen Datenbankkonstruktion bieten sich die bekannten Methoden und Techniken der Systemanalyse an. Allerdings konzentrieren sich diese Ansätze auf die durchzuführenden Operationen, während es bei der Datenbankkonstruktion darauf ankommt, die Zustände und Eigenschaften des zu modellierenden Gegenstandsbereichs möglichst vollständig zu durchdringen.

Analyse der betrieblichen Informationsstruktur

Die Analyse der betrieblichen Realität läßt sich vergleichsweise rasch mit Hilfe verbreiteter Instrumente der Systemanalyse durchführen.

Diese bedienen sich üblicherweise der eingeführten betriebswirtschaftlichen Kategorien. Dazu gehören beispielsweise Geschäfts- und Funktionsbereiche, Organisationsstrukturen, interne oder externe Koalitionsteilnehmer oder die Strukturen des traditionellen Rechnungs-

wesens (z. B. Kontenstrukturen der Finanzbuchhaltung, Kostenstellenstrukturen, Berichtswege)

Die Anlehnung an traditionelle Denkschemata hat auf der einen Seite der Vorteil der vollständigen und multidimensionalen Durchleuchtung, auf der anderen Seite jedoch die Gefahr, daß innovative Aspekte vergessen werden.

Um dieser Gefahr zu begegnen, sollte bei der Analyse der Realität in folgenden Stufen vorgegangen werden:

Stufen der Abstraktion	Kennzeichen	Beispiele
1. Stufe: Objektebene	Objekte des Systems „Unternehmung" und seines Umsystems werden erfaßt und in ihren relevanten Merkmalen beschrieben	- Kunden, Lieferanten, Arbeitnehmer - Gebäude, Fertigungsanlagen
2. Stufe: Typebene	Die bisher im Unternehmen vorgenommene Typenbildung der Objekte und die zugeordneten Merkmale werden analysiert	- Organisationsstruktur - Kundentypen - Maschinengruppen
3. Stufe: Konstruktebene	Die bisher im Unternehmen vorgenommene Abbildung von Sachverhalten in abstrakten Kategorien	- Kontenstrukturen des Rechnungswesens - Formularstrukturen für interne Arbeitsabläufe und externe Geschäftsprozesse

Abb. 3.2.1/3: Stufen der Realitätsbeobachtung

Im einzelnen werden folgende Möglichkeiten genannt, um die betrieblichen Informationsstrukturen zu durchdringen:

Die bisher im Unternehmen verwendeten Formulare, Karteikarten und Listbilder bilden die vorhandene Datenstruktur, wenn auch häufig unstrukturiert und unvollständig, ab. Eine **Formularanalyse** kann daraus die Informationsobjekte sowie deren Attribute und durch Zusammenfassung von Objekten auf einem Formular auch die Datenbeziehungen erfassen.

Oft kann die Datenmodellierung auf bereits vorhandenen DV-Dateistrukturen aufsetzen. Hinzu kommen die Strukturen manuell geführter Karteien, die oft eine wesentliche Informationsquelle darstellen, da sie die betriebsspezifischen Gegebenheiten besser widerspiegeln als die nur geringfügig modifizierten Dateistrukturen von Standardsoftware (**Datenbestandsanalyse**)

Die empirisch erhobenen Datenelemente sind zu klassifizieren, z. B. in folgender Struktur (vgl. Eftimie/Nikles (1988)):

Datenklasse	Beispiele	Bezeichnung
Statische Randbedingungen	Produkte, Ressourcen, Organisation	Objektdaten (Grundobjekte)
Typisierungsstruktur	- Organisationsstrukturen - Produkthierarchien	Strukturdaten (Strukturobjekte)
Dynamische Prozeßbestandteile	Anfragen, Aufträge, Lieferungen	Transaktionsdaten
Ergebnisse der Geschäftstätigkeiten	Deckungsbeiträge, Kostenbudgets	Berichtsdaten (Bestandsobjekte)
Umstrukturierte Kommunikationsdaten	Dokumente, Graphiken	Dokumente

Abb. 3.2.1/4: Klassifikation von Datenelementen

Aus der Analyse des Informationsbedarfs resultierte eine Auflistung der aktuell und zukünftig als geschäftspolitisch bedeutsam eingeschätzten Informationen. Diese sind im Rahmen der Datenkonstruktion zu konkretisieren.

Die **Interviewmethode** ist eine verbreitete Methode in der Systemanalyse. Sie geht vom externen Schema aus und versucht über Befragung der verschiedenen Benutzergruppen deren Anforderungsprofile an das Datenmodell zu identifizieren. Im ersten Schritt sind die Informationsnutzer zu bestimmen. Benutzergruppen sind zum einen Führungskräfte und Sachbearbeiter der unterschiedlichsten Managementebenen, zum anderen die heute und in Zukunft notwendigen DV-Anwendungssysteme. Deren Informationsbedarf drückt sich in den Anfrage an die zu entwerfende Datenbank aus. Die Möglichkeit der direkten Befragung beschränkt sich auf die Mitarbeiter aller Managementebenen. Ersatzweise ist bei DV-Anwendungssystemen die (hoffentlich) vorhandene Schnittstellen-Dokumentation zu analysieren.

Ein standardisiertes Verfahren ist das **Data-ID-Verfahren** (vgl. Ceri (1983), Mayr/Dittrich/Lockemann (1987)). Es beruht auf mehreren formatisierten Schritten, die durch Formulare unterstützt werden.

1. Schritt: Identifikation von Organisationseinheiten (Operatorsicht)
Im ersten Schritt wird der Ausschnitt der Realität auf logische Organisationseinheiten untersucht. Logische Organisationseinheiten bilden die Stellen (als kleinste organisatorische Einheit), die weitgehend homogen sind hinsichtlich ihrer Aufgaben und der dazu verwendeten Begriffswelt. Sie sind oft identisch mit den in den Unternehmen vorzufindenden realen Organisationsstrukturen, doch werden diese oft durch historisch gewachsene Strukturen, regionale Abhängigkeiten etc. verfälscht.

2. Schritt: Identifikation der zu unterstützenden Aufgaben (Objektsicht)
Im zweiten Schritt werden für die identifizierten Organisationseinheiten die zu unterstützenden Aufgaben herausgefunden. Es entsteht dadurch eine Querreferenz von Operator und Objekt.

3. Schritt: Identifikation der zu unterstützenden Personen
Jetzt erfolgt eine Ergänzung der Querreferenz durch die Personen, die die Aufgabenstellung, die notwendigen Informationen sowie die Systemanforderungen präzisieren können. Aus diesen drei Schritten entsteht eine dreidimensionale Sicht, von der speziell die Dimension Aufgabe - Person von Interesse ist.

4. Schritt: Analyse der Aufgabenstellung
Die Aufgabenstellung wird zerlegt in die
- Funktionssicht (erforderliche Arbeitsgänge, dazu notwendige Eingabebedingungen wie Daten, vorgelagerte Operationen und zu erstellende Ausgabedaten)
- Datensicht (vorhandene Karteien / Ablagen, dazu notwendige Arbeitsgänge)

Die Analyse erfolgt auf sogenannten Anforderungs-Sammelblättern, die einer Arbeitsablaufbeschreibung ähneln. Es werden die üblichen Instrumente der Organisationsanalyse (z. B. Beobachtung, Befragung, Dokumentenanalyse) und intensive Dialoge mit den betroffenen Personen genutzt, um die Aufgaben möglichst vollständig zu beschreiben.

5. Schritt: Filterung der Aufgabenstellung
Die auf den Anforderungs-Sammelblättern zusammengefaßte Aufgabenstellung wird mit dem Ziel der Eindeutigkeit und Verständlichkeit bereinigt. Hierzu gehören sprachliche Ungenauigkeiten wie Synonyme (Material = Rohstoff), Homonyme (Stelle im Sinne von Organisationseinheit oder Kostenstelle) und Wiederholungen. Weiterhin werden implizierte Informationen und Abläufe präzisiert (Beispiel: "der Lieferschein wird versandt" wird ersetzt durch "der 2. Durchschlag des Lieferscheins wird an die Rechnungsanschrift des Warenbestellers per Post versandt"). Die Objekte werden durch ihre Attribute beschrieben.

6. Schritt: Überführung in Datenbankstrukturen
Die einzelnen Sätze der gefilterten Anforderungssammelblätter werden jetzt daraufhin geprüft, ob sie Datenoperatoren (Operationen), Datenobjekte oder Ereignisse betreffen. Diese werden dann entsprechend auf Objekt-, Operations- oder Ereignis-Anforderungsblätter übertragen. Objekte lassen sich dadurch erkennen, daß sie durch Substantive beschrieben werden. Objektanforderungen stellen die späteren Attribute dar und lassen sich durch die dem Sustantiv zugeordneten Adjektive erkennen. Operationen und die zugehörigen Attribute werden meist durch Verben gekennzeichnet. Ereignisanforderungen sind in der Regel mit "wenn", "falls", "sofern" oder ähnlich bedingte Sätze, beschrieben.

7. Schritt: Übertragung in Datenbank-Verzeichnisse
Die so identifizierten und formalisierten Objekte, Operationen und Ereignisse werden abschließend detailliert beschrieben. Es werden jeweils pro Datenelement angegeben die identifizierenden und beschreibenden Attribute und die Bezeichnung. Objekte werden semantische und pragmatisch (Verwender, Wertebereich) beschrieben. Zu den Operationen werden der Auswirkungstyp, die Ausführungsart (on Line, Real), die beteiligten Eingabe- und Ausgabeobjekte und die Häufigkeit der Durchführung angegeben. Ereignisse werden zu logischen Organisationseinheiten, Personen und Prozessen zugeordnet und durch die Operations- und Datenausprägung sowie das Resultat spezifiziert.

Abb. 3.2.1/5: Schritte des DATA-ID-Verfahrens

Das Verfahren ist für eine Gesamtorganisation relativ aufwendig. Detailliert durchgeführt kommt es einer Analyse und Neugestaltung nicht nur der DV sondern auch der Organisation gleich. Doch es zeigt die zu durchlaufenden Schritte bei der Konstruktion eines Datenmodells auf.

3.2.1.2. Konstruktionsprobleme

3.2.1.2.1. Objektbildung

Objekttypen ergeben sich entweder aus realen Objekten des Wirklichkeitsbereiches und deren Zusammenfassung über die Gleichartigkeit von Merkmalen (Typisierung) oder als abstraktes Konstrukt.

Abstrakte Konstrukte existieren in einem spezifischen Realitätsbereich als menschliche Vorstellungen oder Strukturierungen. Beispielsweise werden die realen „Mitarbeiter" zu den Konstrukten „Stelle" zugewiesen, die ihrerseits wieder zu größeren Organisationseinheiten (= Konstrukten) wie „Abteilungen" zusammengefaßt werden.

Die Bildung von Objekten erfolgt somit entweder durch die Analyse der realen oder logischen Strukturen des Gegenstandsbereiches, wobei sich der Konstrukteur an den realen Objekten orientiert (objektgetriebenes Vorgehen) oder durch Analyse der relevanten Informationsinhalte, deren Definition in Attributen und Zusammenfassung in logischen Konstrukten (attributgetriebenes Vorgehen).

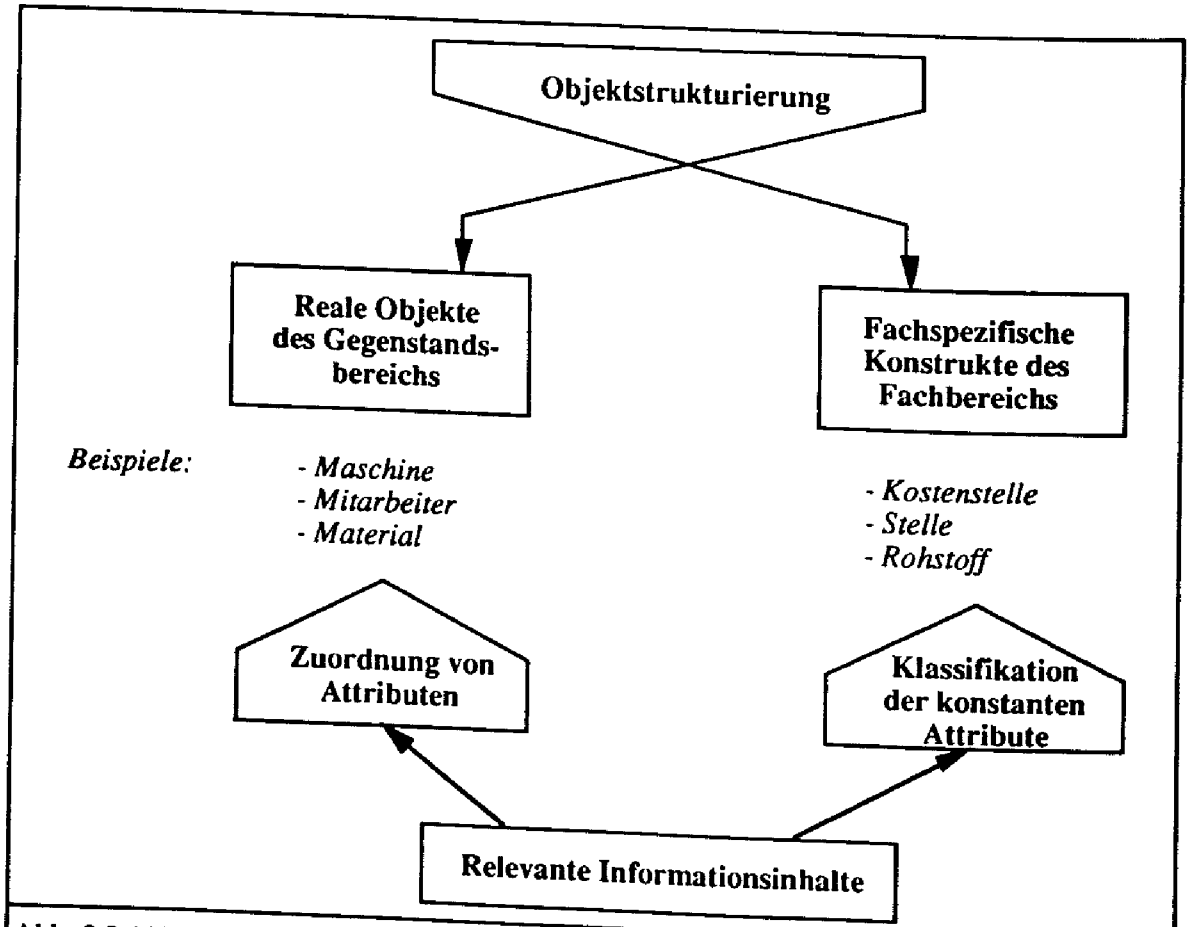


Abb. 3.2.1/6: Reale Objekte und fachspezifische Konstrukte

Datenobjekte sind strukturelle Kennzeichen des Gegenstandsbereiches mit einer längeren Lebensdauer. Sie unterscheiden sich durch diese Lebensdauer von Ereignissen. Datenobjekte werden daher in der Praxis auch "Grundobjekte" genannt.

In dynamischer Sicht beginnt und endet die Existenz eines Datenobjekts mit einem definierten Ereignis (z. B. Einstellung und Ausscheiden eines Mitarbeiters). Grundobjekte

stehen mit anderen Grundobjekten in struktureller Beziehung. Diese kann natürlich sein, sich aus physischen oder lokalen Gegebenheiten ergeben oder als "Konstrukt" eingeführt worden sein. Man spricht in der Praxis von Strukturobjekten.

Klassenbildung	**Beispiele**
aufgrund natürlicher Eigenschaften	- Mitglieder einer Familie
aufgrund physischer oder lokaler Gegebenheiten	- Maschinen einer Werkshalle - Kunden in einer Stadt
aufgrund eingeführter Konstrukte	- Maschinen einer Kostenstelle - Mitarbeiter einer Abteilung

Abb. 3.2.1/7: Formen der Klassenbildung

Die Klassen können wiederum eigene "Klassenobjekte" bilden (z. B. Abteilung) oder sie werden über Beziehungen zwischen den Grundobjekten jeweils neu gebildet.

Grundobjekte existieren entweder unabhängig von anderen Objekten (unabhängiges Objekt) oder nur im Zusammenhang mit anderen Objekten (abhängiges Objekt). Unabhängige Objekte sind zumeist Gegenstände der Realität, die eigenständig existent und relevant für den betrachteten Realitätsausschnitt sind (= strong entity). Abhängige Objekte (= weak entity) werden auch als Wert bezeichnet. Abhängige Objekte lassen sich häufig sinnvoll als Beziehung oder als Attribut eines Objektes oder einer Beziehung modellieren.

Eine der wesentlichen Aufgaben der Datenmodellierung, hier speziell der Objektbildung, ist die Festlegung des Abstraktionsniveaus. Gelingt es, Gemeinsamkeiten zwischen den Begriffen oder Gegenständen der Realität zu entdecken, so können dadurch bestimmte Systemkomponenten mehrfach verwendet werden.

Die Abstraktion beruht auf einer zielgerichteten Suche nach gemeinsamen Eigenschaften. Gemeinsame Attribute sind das Ergebnis des vorgelagerten Schrittes einer Standardisierung von Datenelementen mit Hilfe einer Begriffsklärung und Begriffsdefinition.

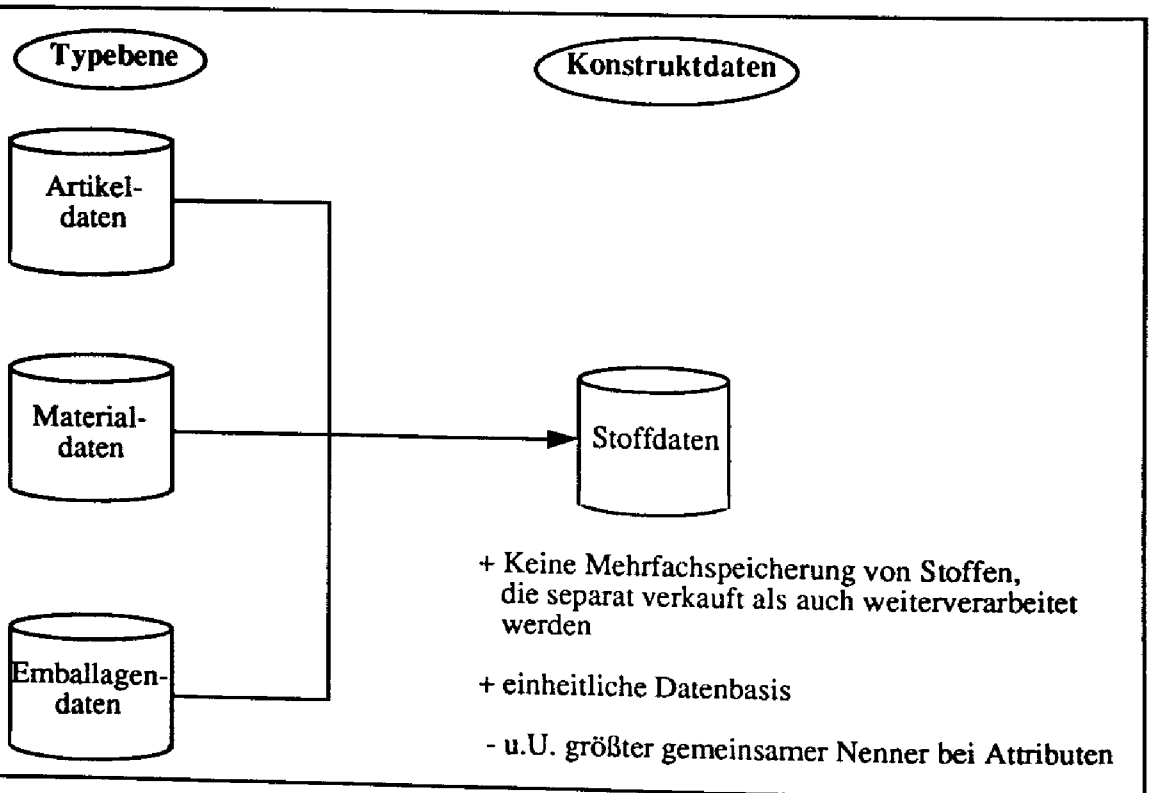


Abb. 3.2.1/8: Abstraktion und Konsequenzen für die Datenbankstruktur

Beispiele sind etwa:

- Kunden, Lieferanten, Angestellte werden auf einer höheren Abstraktionsebene zu MARKTPARTNERN zusammengefaßt oder
- verkaufsfähige Produkte, Halbfertigfabrikate sowie Roh-/Hilfs- und Betriebsstoffe zu STOFFEN

Ob eine entsprechende Abstraktion sinnvoll ist und wie hoch das Abstraktionsniveau zu wählen ist, entscheidet man sinnvoll vor

- dem Hintergrund gemeinsamer Attribute (Attributidentität)
- den Vorteilen bei der Gestaltung von Datenbank- und darauf aufbauenden Anwendungssystemen.

3.2.1.2.2. Beziehungsbildung

Beziehungen stellen reale oder logische Verknüpfungen zwischen Datenobjekten dar. Ihre Existenz ist somit abhängig von den zugehörigen Datenobjekten.

Reale Beziehungen sind meist Aktivitäten, die sich sinnvoll durch Verben (besser "Tuwörter") bezeichnen lassen. Logische Beziehungen ordnen zum Beispiel Objekten bestimmte Obermengen zu (Beispiel: Artikel zu Produkten) oder stellen logische Abhängigkeiten dar (Artikel besteht aus Bauteilen).

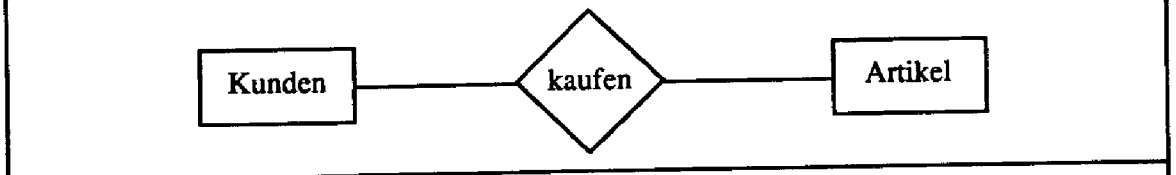


Abb. 3.2.1/9: Reale Beziehungen

Beziehungstypen ergeben sich entweder aus Beziehungen, die Aktivitäten zwischen realen Objektausdrücken oder aus abstrakten Konstrukten, die zum Beispiel rechtliche oder organisatorische Zuordnungsverhältnisse ausdrücken sollen.

Es ist eine bewußte Konstruktionsentscheidung, welche Erscheinungen der Realität als „Objekte" und welche als „Beziehungen" modelliert werden. Verschiedene Konstrukteure können demzufolge zu durchaus unterschiedlichen Beschreibungen derselben Realität kommen (vgl. Mayr/Dittrich/Lockemann (1990), S. 500 f)).

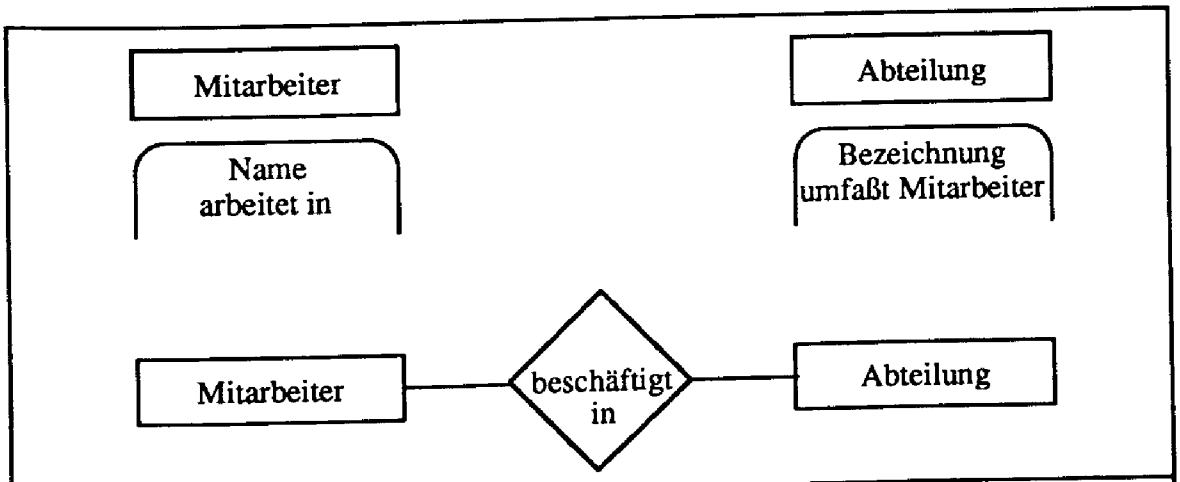


Abb. 3.2.1/10: Modellierung einer Beziehung als Relation oder als Attribut

Beziehungen können existieren

(a) zwischen zwei Datenobjekten unterschiedlichen Typs
(b) zwischen zwei Datenobjekten des gleichen Typs

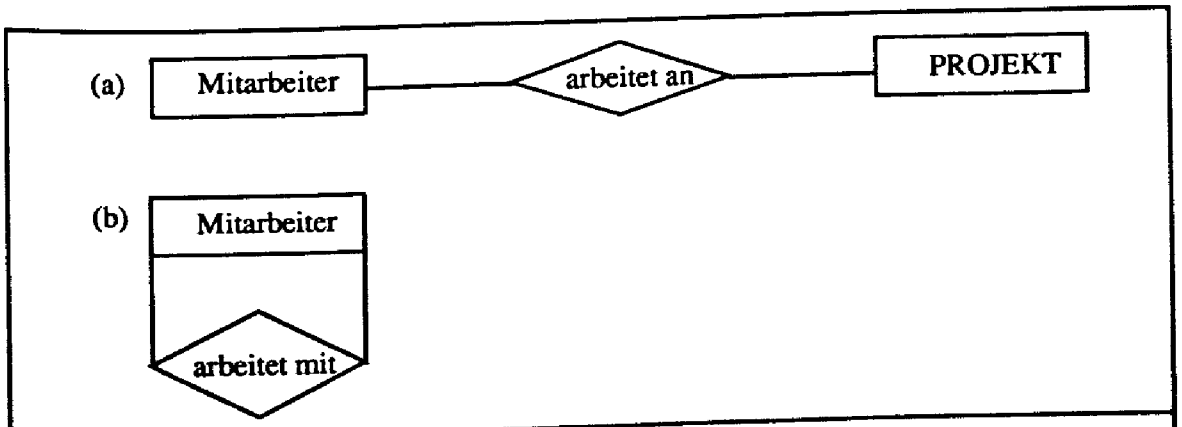


Abb. 3.2.1/11: Relationen zwischen Entities (a) unterschiedlichen und (b) identischen Typs

Ein wichtiges Kriterium für eine Beziehung (Relation) ist ihr Komplexität (Kardinalität); hierunter werden Angaben darüber verstanden, mit wievielen anderen Entities ein Entity eines bestimmten Typs in einer konkreten Beziehung stehen kann. Es gibt mehrere Notationen, um die Komplexität auszudrücken. Die gebräuchlichste ist die 1:n-Notation. In dieser Notation werden die maximal möglichen Verknüpfungen angeben.

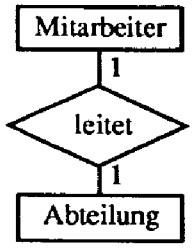
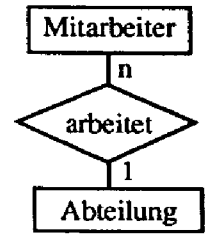
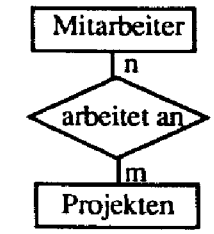
Beziehung	Beschreibung	Beispiel
1 : 1	Ein Datenobjekt steht mit maximal einem anderen Objekt in Beziehung	
1 : m	Ein Datenobjekt steht mit mehreren anderen in Beziehung	
n : m	Einem Datenobjekt der ersten Menge werden mehrere Objekte der zweiten Menge zugeordnet und umgekehrt	

Abb. 3.2.1/12: Komplexität von Beziehungstypen

Sinnvoll ist es, bei der Angabe der Komplexität von Beziehungstypen Minimalkardinalitäten und Maximalkardinalitäten zu unterscheiden. Durch diese Differenzierung läßt sich der semantische Aussagegehalt erheblich erweitern; zudem erleichtert sie die Umsetzung in ein logisches Datenmodell.

Zugeordnete Datenobjekte	Beispiel	vollständige Notation	verkürzte Notation (nach Schlageter/Stucky)
genau 1	Mitarbeiter arbeitet in Abteilung	(1,1)	1
maximal 1	Mitarbeiter leitet Abteilung	(0,1)	c(hoice)
minimal 1 maximal beliebige		(1,*)	m(ultiple)
minimal keines maximal beliebige	Mitarbeiter arbeitet an Projekten	(0,*)	mc

Abb. 3.2.1/13: Alternative Notationen für die Kennzeichnung von Beziehungen

Dabei wird in der Notation angegeben, wieviele Entities minimal bzw. maximal an einer Beziehung beteiligt sind. Die Wahl der angemessenen Komplexität von Beziehungstypen ist von erheblicher Bedeutung für die Datenstrukturen, da sie später die Redundanz und Performance einer Datenbank beeinflußt.

Grundsätzlich sind parallele Beziehungen zwischen Entity-Typ und Relationship-Typ zugelassen. Dies entspricht dann einer rekursiven Beziehung und bedeutet, daß an einer Beziehung mehrere Entities desselben Typs beteiligt sein können.

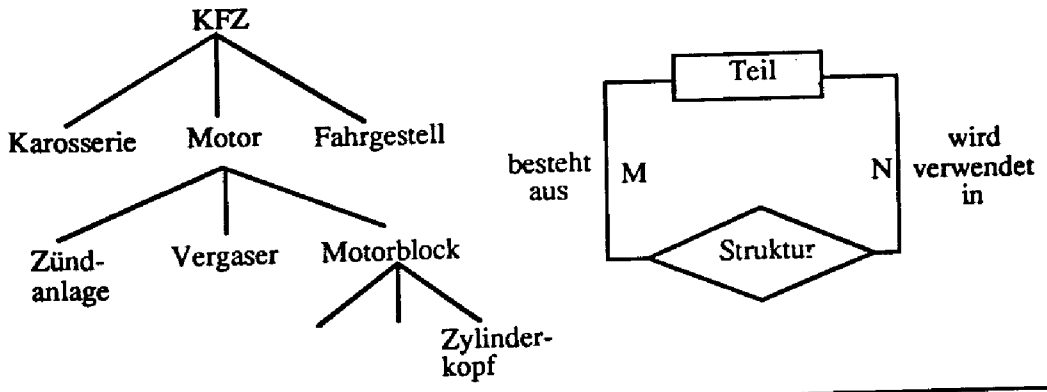


Abb. 3.2.1/14: Stücklistenstruktur im ERM als Beispiel für parallele Beziehungen zwischen Entitytypen (vgl. Sinzig (1990), S. 20 und Scheer (1991), S. 79)

In dieser graphischen Notation lassen sich hierarchische und vernetzte Strukturen abbilden.

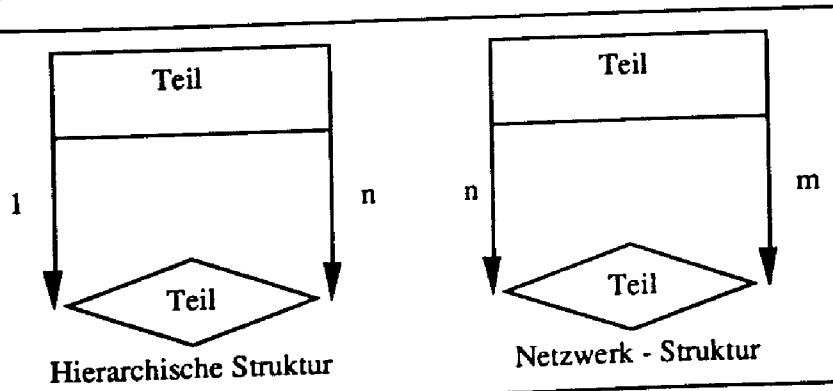


Abb. 3.2.1/15: Abbildung hierarchischer und vernetzter Strukturen im Entity-Relationship-Modell

3.2.1.2.3. Attributzuordnung

Attribute ordnen Objekten und Beziehungen und ggf. auch Ereignissen Werte zu; dabei unterstützt die bewußte Untersuchung der Werte, die ein Attribut zuordnen soll, den Konstruktionsprozeß.

Attribute von Objekten (Entities)

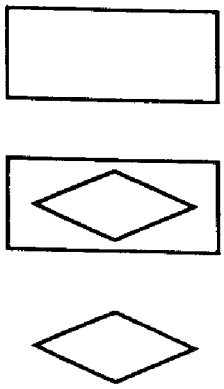
Attribute ordnen Objekten bestimmte Eigenschaften zu. Die Attribute werden entweder benötigt zur Beschreibung der relevanten Eigenschaften eines Objektes aus dem Realitätsbereich oder für Zwecke der Informationsverarbeitung.

Datenobjekt Kunde	**Eigenschaften**	**Beispiele**
Realitätsrelevante Attribute	dienen dem Güter- und/oder Zahlungsfluß	Name Kundenadresse Lieferadresse
Verarbeitungsrelevante Attribute	dienen der Informationsverarbeitung	Debitorenkonto Kundentyp

Abb. 3.2.1/16: Realitäts- und verarbeitungsrelevante Attribute

Attribute von Beziehungen

Es ist in den Arbeiten von Chen umstritten, ob neben den Datenobjekten auch die Relationen Attribute besitzen dürfen. In späteren Arbeiten (Chen (1985)) verneint er diese Frage und führt ein zusätzliches Konstrukt ein für Relationen mit Attributen, das er "composite entity type" nennt.



Datenobjekt (entity type)	- besitzt Attribute
Konstruiertes Datenobjekt (composite entity type)	- besitzt Attribute
Beziehung (relationship type)	- besitzt keine Attribute

Abb. 3.2.1/17: Konstruiertes Datenobjekt als Verbindung von Beziehung und Datenobjekt

Es sei den früheren Arbeiten von Chen (1976) gefolgt, die Attribute für Beziehungen vorsehen, d. h. in der Folge wird auf konstruierte Datenobjekte verzichtet. Grund dafür ist, daß der Verzicht auf künstliche Datenobjekte vom Konstrukteur zumindestens in den ersten Schritten eine stärkere Anlehnung an die Realität verlangt.

Beziehungen müssen nicht, können jedoch Attribute besitzen. Beziehungen mit Attributen bilden aus betriebswirtschaftlicher Sicht sogenannte „Bestandsobjekte", die einer Kombination von Grundobjekten Werte zuweisen.

Formale Arten von Attributen

Attribute können formal folgenden Charakter haben:

Attributtyp	Kennzeichen
identifizierend	Vergabe eines eindeutigen Attributs zur Identifizierung eines Datenobjektes oder einer Datenbeziehung
typisierend	Zuordnung eines Datenobjektes oder einer Beziehung zu einem Objekttyp oder einem Beziehungstyp
deskriptiv	Zuordnung beschreibender Merkmale
chronologisch	Zuordnung chronologischer Merkmale mit dem Ziel der Ableitung einer zeitlichen Historie
prozedural	Zuordnung von Merkmalen, die von Anwendungsprogrammen für Auswertungsprozeduren benötigt werden

Abb. 3.2.1/18: Formale Struktur von Attributen

Identifizierende Attribute (sogenannte Schlüsselattribute) sollen ein Datenobjekt eindeutig kennzeichnen, ihm einfach im laufenden Betrieb zuordbar sein und möglichst wenig Speicherplatz verbrauchen. Typisierende Attribute verknüpfen einzelne Datenobjekte mit bestimmten Typen; beispielsweise wird „Max Müller" der Klasse der Mitarbeiter zugewiesen.

Werteattribute können deskriptiver, chronologischer oder prozeduraler Art sein. Sie kennzeichnen ein Datenobjekt inhaltlich durch bestimmte Werteausprägungen. Werteattribute werden dem Datenobjekt durch bestimmte funktionale Vorschriften zugeordnet, die

- auf Operationen im Objektsystem (d.h. in der Realität) basieren (z. B. Meßoperationen) und Eingabe-Transaktionen im Informationssystem bedingen,
- auf Operationen im Informationssystem basieren (z. B. Berechnungen) und somit Verarbeitungstransaktionen (mit entsprechenden Prozeduren) bedingen (berechnete Attributausprägungen).

Berechnete Attributausprägungen lassen sich durch Operationen im Informationssystem jeweils neu rekonstruieren, so daß im späteren Schritt der Datenschema-Bildung zwischen Berechnungsaufwand und Speicherplatzanforderungen abzuwägen ist.

Domänen von Attributen

Die zulässigen Ausprägungen der Attribute (Wertebereiche) werden durch sogenannte Domänen definiert. Die Domänen von Attributen lassen sich durch Aufzählung, durch Angabe von Bereichsgrenzen oder von Definitionsfunktionen beschreiben.

Form	Beispiel
Explizite Auflistung	Produktnamen = (POLO, GOLF, PASSAT,SCIRROCCO, CORRADO)
Bereichsgrenzen	000001 < Artikelnummern < 999999
Definitionsfunktion	BRUTTOBETRAG = NETTOBETRAG + (1+ UMSATZSTEUERSATZ/100)

Abb. 3.2.1/19: Definition von Domänen für Attribute

Es sollte eine Vereinheitlichung und Standardisierung der Attribute angestrebt werden. Diese geht mit einer exakten Definition der zugrundeliegenden Begriffe einher.

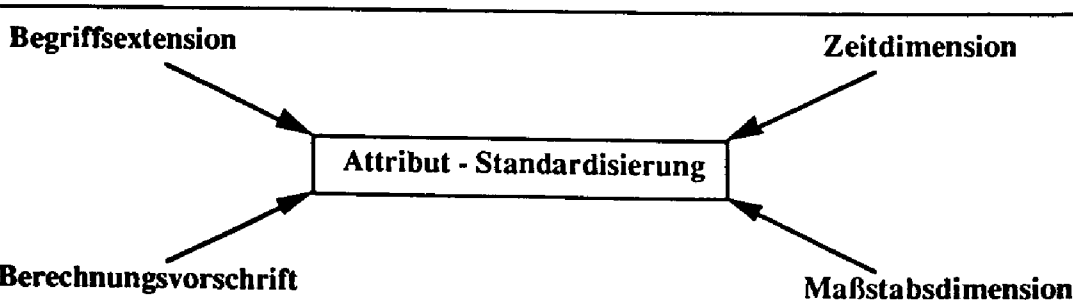


Abb. 3.2.1/20: Möglichkeiten zur Vereinheitlichung von Attributen

Die Heterogenität der Begriffswelt in Unternehmen ist häufig auf geringfügige Unterschiede im Zeitbezug der Maßstabsdimension oder bestimmte Definitionsunterschiede zurückzuführen. Beispielsweise existieren in den Unternehmen oft viele unterschiedliche Definitionen für Wechselkurse, die in vielen Fachabteilungen für Umrechnungs- und Bewertungszwecke verwendet werden.

		Beispiel: Wechselkurs
Zeitdimension	Zeitperiode	Tages-, Monatsmittel-, Jahresmittelkurs
	Zeitpunkt	12.00, Börsenschluß
Berechnungsvorschrift		ungewichteter Durchschnittskurs gewichteter Durchschnittskurs
Maßstabsdimension		X DM für 100, 1000 ... Fremdwährungseinheiten
Begriffsextension	Inhalt	Geld- oder Briefkurs Kassa- oder Terminkurs Devisen- oder Sortenkurs
	Ort	Frankfurter Börse, Londoner Börse
	Umfang	ohne/mit Bankspesen

Abb. 3.2.1/21: Einflußgrößen auf die Domänen des Attributs "Wechselkurse"

Attribute können auf der Real- wie auch auf der Typebene für Objekte, Beziehungen und Ereignisse existieren.

Attributbezug **Abstraktionsebene**	**Objekt**	**Beziehung**	**Ereignis**
Attribute der Realebene	einfache (single value)		
	komplexe (multi value)		
Attribute der Typebene (Typattribute)	COUNT = Anzahl der Objekte in einem Typ;		
	AVERAGE = durchschnittliche Ausprägung eines Attributs über alle Objekte)		

Abb. 3.2.1/22: Formale Typen von Attributen

Attribute können in sich einfache Datenobjekte sein (single value attributs), die numerische oder alphanumerische Werte umfassen; es kann sich aber auch um komplexe Datenobjekte

handeln. Komplexe Attribute können beispielsweise aus Graphiken, Dokumenten oder Algorithmen bestehen.

Merkmal	Denkbare Ausprägungen			
Attribut-struktur	einfach (single-value)	komplex (multi-value)		
Attribut-kompetenz (z.B. Änderungs-rechte)	extern definiert (z.B. im Rahmen einer Datenbank-sprache)	intern definiert		
Attribut-integrität	extern gesichert (z.B. durch externe Programme)	intern gesichert		
Attributwerte-ausprägungen	extern bestimmt (measured attributs)	intern definiert (z.B. durch interne Algorithmen) (calculated attributs)		
Attributbezug	Objekt	Beziehung	Ereignis	
Attribut-abstraktions-ebene	Real-ebene	Typebene	Konstruktebene	

Abb. 3.2.1/23: Typisierung von Attributstrukturen in formaler Sicht

Betriebswirtschaftliche Arten von Attributen

Attribute in betriebswirtschaftlichen Anwendungssystemen lassen sich in die Kategorien

- Identifizierende / beschreibende Attribute
- Leistungsverkehrsbestimmende Attribute
- Zahlungsverkehrsbestimmende Attribute
- Rechnungswesen-bestimmte Attribute
- Steuerrechtliche Attribute
- Strategische Attribute

einteilen. Diese Einteilung korrespondiert weitgehend mit den Ebenen der Unternehmung: Steuerungsebene, Leistungsebene, Finanzebene und den dort verwendeten DV-Systemen.

		Kunde	kauft	Produkt
Operative Attribute	Identifizierende Attribute	- Kundennr. - Kundennamen - Anschrift	- Auftragsnr. - Auftragsdatum	- Produktnr. - Produktbezeichnung
	Leistungsverkehrsbestimmte Attribute	- Lieferanschrift	- Auftragsmenge	- Produktabmessungen - Verpackungsart - Gefahrguteinstufung - Aggregatzustand
	Zahlungsverkehrsbestimmte Attribute	- Kreditlimit - Rabattkonditionen - Zahlungs-/Skonto-Konditionen	- Auftragspreis des Produktes - Rabatte	- Bruttopreis des Produktes
	Steuerrechtliche Attribute	- Umsatzsteuervorschriften des Kunden	- Umsatzsteuerabnehmerpräferenz	- Umsatzsteuersatz
Strategische Attribute		- Kundenspezifika	- "Geschichte" der Kundenbeziehungen	- Kundeneignung des Produktes

Abb. 3.2.1/24: Kategorisierung von strategischen Attributen nach betriebswirtschaftlichen Gesichtspunkten mit Beispielen

Während die operativen Attribute des Güter- und Zahlungsverkehrs innerhalb des Wirtschaftsverkehrs eines Landes bzw. Wirtschaftsraumes aufgrund der üblichen kaufmännischen Gepflogenheiten zumindestens in Branchen weitgehend normiert sind, sind die strategischen Attribute das eigentliche Feld der Datenkonstruktion aus betriebswirtschaftlicher Sicht.

Strategische Attribute sind zum einen Informationsnotwendigkeiten, die sich aus den angestrebten Wettbewerbsfaktoren auf dem Markt, zum anderen aus internen Leistungsvorteilen der Produkt- und Verfahrenstechnologie und zum Dritten aus strategischen Vorteilen im Controlling-Prozeß ergeben.

Beispiel:
Betrachtet werden soll ein Hersteller und Vermarkter von Kosmetik- und Körperpflegemitteln.

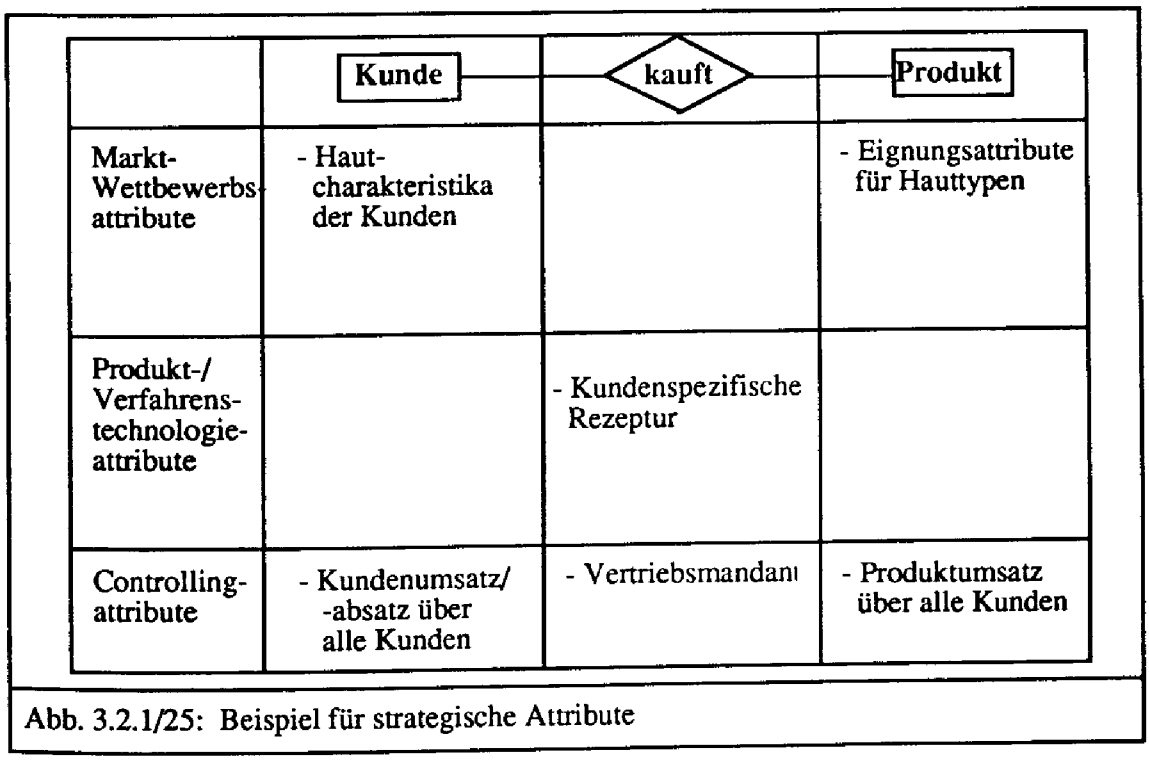


Abb. 3.2.1/25: Beispiel für strategische Attribute

Die Speicherung der Hautcharakteristika der Kunden ermöglicht dem Anbieter eine kundenspezifische Produktdifferenzierung. Allerdings stellt dieses sehr hohe Anforderungen an die Produktionsplanung und Produktionsdurchführung. Unter anderem müssen kundenspezifische Rezepturen gehalten werden, die Technologie muß die Fertigung sehr kleiner Chargen und deren kundenspezifische Kommissionierung ermöglichen und schließlich muß die Logistik auf den Endverbraucher eingestellt werden.

Controlling-Attribute bilden die Grundlage für die Ermittlung der jeweiligen Erfolgspositionen. Im Falle eines Kundenauftrages wird

- *sowohl der Kundenumsatz zum Zwecke einer Kunden-Ergebnisrechnung*
- *als auch der Produktumsatz für eine Produkt-Ergebnisrechnung*

ermittelt. Möglich ist weiterhin die Berechnung des Vertriebsergebnisses des jeweiligen Vertriebsmandanten (z. B. einer Verkaufsniederlassung).

Betriebswirtschaftliche Zuordnung von Attributen

Die Attributierung aus betriebswirtschaftlicher Sicht kann sich an den Merkmalen der vom Datenmodell zu bedienenden Anwendungssysteme orientieren. Dabei sind systemweite von objektspezifischen und beziehungsspezifischen Fragen zu unterscheiden.

Systemweite Entscheidungen

Systemweit sind folgende Fragen zu unterscheiden:

1. Ist das Anwendungssystem mandantenfähig zu gestalten?

Unter mandantenfähigen Systemen soll ein Anwendungssystem verstanden werden, das jeweils Abrechnungskreise für die wirtschaftlichen, rechtlichen und technischen Einheiten

eines Unternehmens bereit hält. Bei einem mandantenfähigen System ist für jedes Datenobjekt eine Mandantenkennung als identifizierendes Attribut vorzusehen.

2. Welcher Integrationsgrad des Systems wird angestrebt?

Informationssysteme sind in der Regel Bestandteile einer unternehmensinternen Systemarchitektur oder einer unternehmensübergreifenden Informationsinfrastruktur.

Die Integration von Informationssystemen in horizontaler, vertikaler und zeitlicher Richtung erfordert die Definition von DV-System- und organisatorischen Schnittstellen. Software-Schnittstellen beschreiben die Menge an Informationen, die ein Teilsystem anderen Systemen zur Verfügung stellt oder von diesen benötigt (vgl. Spitta (1989), S. 22).

Schnittstellen können betreffen
-> DV-Systeme des Unternehmens oder in der unternehmensübergreifenden Wertschöpfungskette (DV-Schnittstellen),
-> organisatorische Einheiten im oder außerhalb des Unternehmens (organisatorische Schnittstellen).

Organisatorische Schnittstellen beschreiben die Daten, die außerhalb der Systemgrenzen des zu konzipierenden Datenmodells erzeugt werden und in dieses einfließen und umgekehrt. Sie sind in aller Regel identisch mit den Benutzerschnittstellen eines zu entwickelnden Dialogsystems (externes Schema).

Die DV- und die Organisationsschnittstellen definieren die Attribute, die als Input von anderen Einheiten geliefert oder von diesen benötigt werden. Hiefür bedarf es

-> semantischer Input-Prozeß-Output-Analysen, in denen pro Attribut geklärt wird, welche Begriffsdefinition verwendet wird.

Beispiel:
In einem Konzern traten erhebliche Probleme bei der Schnittstelle „Monatsumsatz pro Produkt" auf, da
-> das Vertriebsberichtssystem diesen berechnete als
*Umsatz = Absatz (Monat, Produkt) * Preis (Monat, Produkt) * Kurs (Monat)*
(additive Berechnung)
-> das Fakturierungssystem rechnete
*Gesamt-Umsatz = Absatz(Teilperiode, Produkt) * Preis(Teilperiode, Produkt) * Kurs(Teilperiode, Produkt)*
Umsatz = Gesamtumsatz(aktueller Monat) ./. Gesamtumsatz (Vormonat)
(subtraktive Berechnung)

Struktur- und Wechselkurseinflüsse machen beide Größen nicht vergleichbar.

-> syntaktischer Input-Prozeß-Output-Analysen, in denen pro Attribut geklärt wird, welche Datenformate verwendet werden.

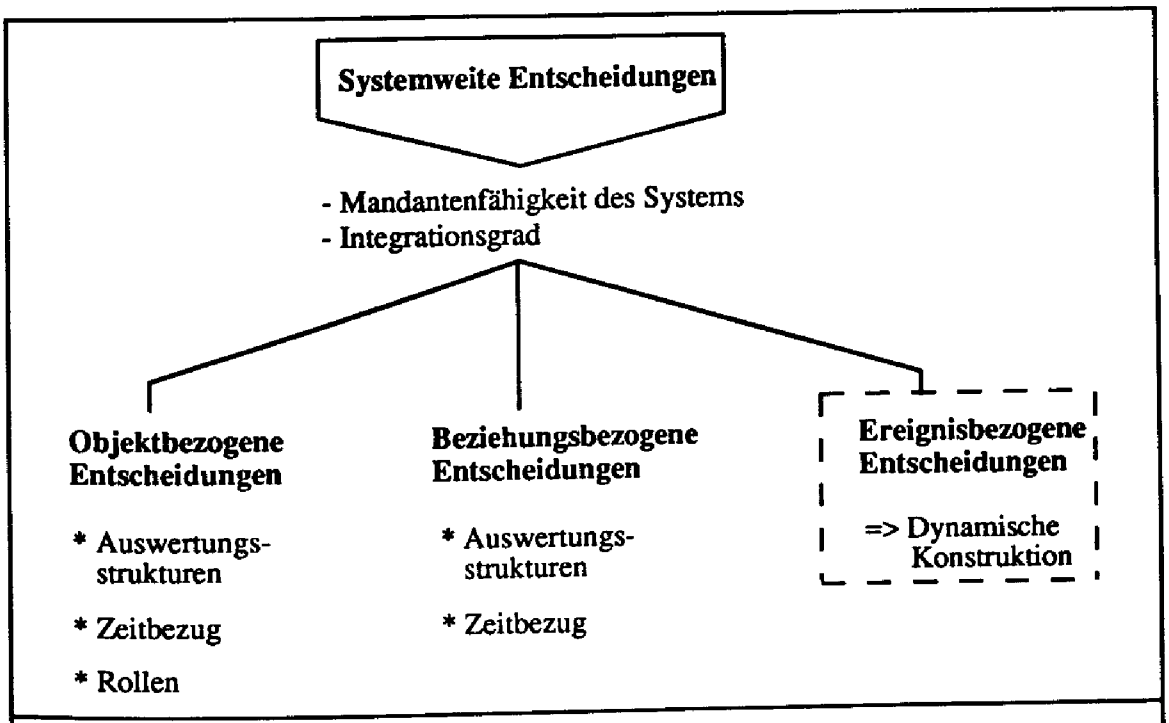


Abb. 3.2.1/26: Entscheidungen der Attributierung

Objektbezogene Entscheidungen:

1. Welchen Auswertungsstrukturen ist ein Datenobjekt zuzuordnen?

Auswertungsstrukturen können hierarchischer oder vernetzter Natur sein. DV-technisch können sie über direkte Verkettung oder indirekt über Strukturobjekte realisiert werden.

	Direkt	**Indirekt**
Objekte	Grundobjekte	Grundobjekte Strukturobjekt
Attributierung	Verkettungsattribut je Auswertungsstruktur	
Vor-/Nachteile	+ keine zusätzlichen Strukturobjekte	+ Änderungsfreundlichkeit, d. h. es werden bei neuen Auswertungsstrukturen die Grundobjekte nicht verändert, sondern nur die Strukturobjekte + Strukturgenerationen, d. h. es können mehrere zeitliche Ausprägungen von Auswertungshierarchien gehalten werden

Abb. 3.2.1/27: Objektbezogene Entscheidung „Realisierung von Auswertungsstrukturen"

Beispiel:
Bei einer direkten Verkettung werden innerhalb einer Organisationsstruktur die Stellen Abteilungen, die Abteilungen Bereichen etc. über entsprechende Verkettungsattribute zugewiesen.

Bei einer indirekten Verkettung werden spezielle Strukturobjekte gebildet, die jeweils die identifizierenden Attribute der zu verknüpfenden Objekte (Stelle 1, 2 .... u und Abteilung x) sowie ggf. die Verknüpfungsprozedur enthalten. Die Verknüpfungsprozeduren können pro Attribut der Objekte unterschiedlich sein.

Auswertungsstrukturen können auf Attribute von Objekten oder Beziehungen angewandt werden. Es ist möglich, daß sie sich aus Aggregationen der Objekte ergeben oder davon unabhängig sind.

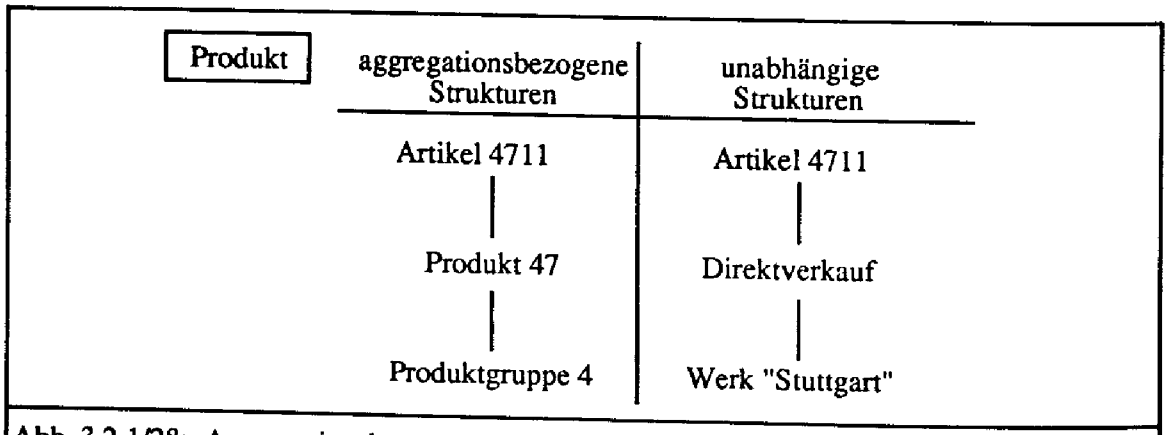


Abb. 3.2.1/28: Aggregationsbezogene und unabhängige Strukturen

2. Welchen Zeitbezug sollen Objekte haben?

Die Existenz von Objekten und deren Zuordnung zu bestimmten Auswertungsstrukturen ist zeitabhängig.

Beispiel:
Ein Artikel kann aus dem Sortiment gelöscht oder in dieses aufgenommen werden.
Der Artikel kann einem bestimmten Produkt zugerechnet werden und zu einem späteren Zeitpunkt einem anderen Produkt.

Attribute von Objekten können zum einen Plan-Werte oder Prognose-Werte, bezogen auf die Zukunft, zum anderen Ist-Werte verschiedener Perioden sein.

Die Existenz von Objekten kann von vornherein auf einen bestimmten Zeitraum begrenzt sein (z. B. bei bestimmten Sonderartikeln), sie ergibt sich im Zeitablauf (z. B. durch Einstellung oder Kündigung von Mitarbeitern) oder sie ist auf einen unbegrenzten Zeitraum ausgelegt (z. B. bei Standorten). Je nach Fall sind bestimmte Attribute für die Existenzzeiten der Objekte vorzusehen.

3. Welche Rollen sollen Objekte erfüllen?

Ein Objekt erfüllt in verschiedenen betrieblichen Teilbereichen bestimmte Rollen. So hat ein Produkt Bedeutung

-> für die Produktion
-> für den Vertrieb
-> für das Marketing
-> für die Fakturierung / Buchhaltung.

Die von einem Objekt zu erfüllenden Rollen vervielfachen sich, wenn aus realen Objekten mittels Konstruktionsoperatoren bestimmte Konstrukte entwickelt werden.

Konstrukt	Typ	Rollen
Stoff	Material	- Beschaffungseinheit - Logistikeinheit
	Bauteil	- Logistikeinheit - Produktionseinheit
	Baugruppe	- Logistikeinheit - Produktionseinheit - Verkaufseinheit
	Artikel	- Logistikeinheit - Beschaffungseinheit - Produkteinheit - Verkaufseinheit
	Set	- Logistikeinheit - Beschaffungseinheit - Produkteinheit - Verkaufseinheit
	Gebinde	- Logistikeinheit - Beschaffungseinheit - Produkteinheit - Verkaufseinheit

Abb. 3.2.1/29: Rollen von Objekttypen und Objektkonstrukten

Die Anzahl und die Ausprägungen der Rollen bestimmen die notwendigen Attribute.

Bei der Ableitung von Objektkonstrukten aus Objekttypen sollte darauf geachtet werden, daß möglichst Objekttypen mit gleichen Rollen zu Konstrukten zusammengefaßt werden, um eine möglichst große Zahl gemeinsamer Attribute zu erreichen.

Beziehungsbezogene Entscheidungen:

1. Welchen Auswertungsstrukturen ist eine Beziehung zuzuordnen?

Aus Beziehungen zwischen Objekten ergeben sich die Bestandsdaten der Objekte.

Beispiel:
Aus der Beziehung „Kunde kauft Produkt" ergibt sich beispielsweise als Bestandsdaten
-> der Produktumsatz
-> der Kundenumsatz

Für Auswertungszwecke sind die Bestandsdaten entweder den Datenobjekten mit Attributen zuzuweisen oder es sind spezielle Auswertungsstrukturen über Beziehungen zu legen.

2. Welchen Zeitbezug soll eine Beziehung haben?

Betriebswirtschaftliche Anwendungssysteme unterscheiden üblicherweise drei Arten des Zeitbezuges.

Beziehung „Kunde kauft Produkt“	Ist- Daten	Prognose-Daten (forecast)	Planungsdaten
	Ist-Umsatz	Vorschau Umsatz	Plan-Umsatz

Abb. 3.2.1/30: Beziehungsbezogene Entscheidung „Zeitbezug“

Planungsdaten werden vor Beginn eines Zeitraumes durch die verantwortlichen Führungskräfte ermittelt, Prognosedaten werden innerhalb einer Periode ermittelt und schätzen den Periodenwert ab. Ist-Daten ermitteln den innerhalb einer Periode realisierten Wert.

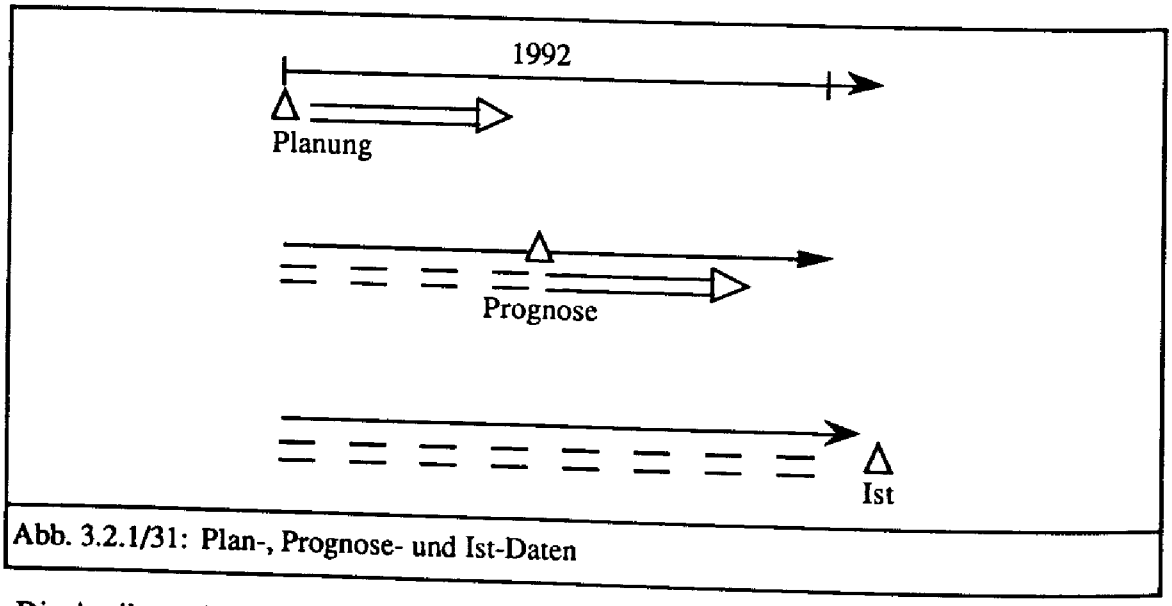


Abb. 3.2.1/31: Plan-, Prognose- und Ist-Daten

Die Attribute einer Beziehung (in der Praxis wird häufig von Bestandsobjekten gesprochen) können damit vom Zeittyp Plan, Prognose oder Ist sein. Für jeden dieser Zeittypen sind unterschiedliche Zeitperioden zu unterscheiden.

Beispiel:
Plan-Umsatz für einen Kunden mit einem Produkt bei einer Drei-Jahresplanung.

Planungszeitpunkt		
Dezember 1989	Plan-Umsatz 1992 (aus 1989)	= 3. Planjahr
Dezember 1990	Plan-Umsatz 1992 (aus 1990)	= 2. Planjahr
Dezember 1991	Plan-Umsatz 1992 (aus 1991)	= 1. Planjahr

Abb. 3.2.1/32: Abhängigkeit eines Plandatums vom Planungszeitpunkt

Die Beobachtung dieser zeitlichen Bezüge ist besonders dann wichtig, wenn Systeme mit zeitlicher Reihung mit Systemen mit rollierender Zeit verbunden werden. Während letztere nacheinander Perioden konstanter Länge betrachten, schreiben Systeme mit rollierender Zeit kontinuierlich einen Zeitraum konstanter Länge fort.

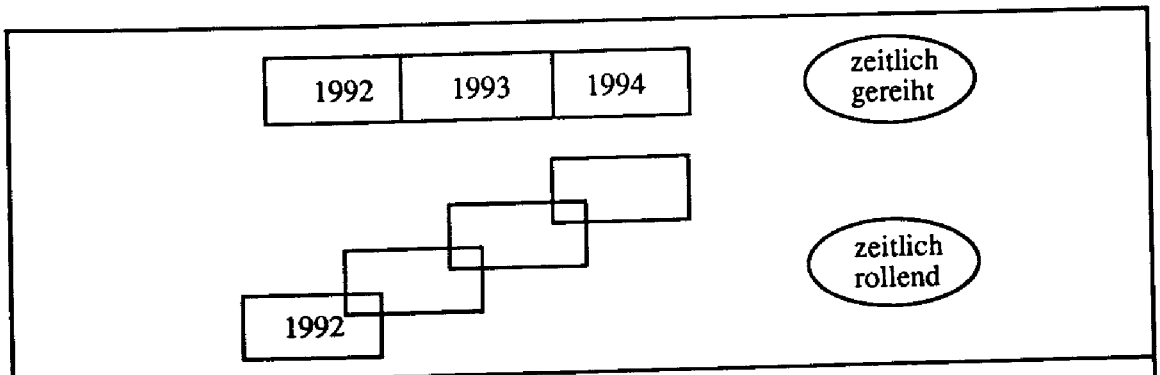


Abb. 3.2.1/33: Zeitliche Reihung und Rollierung

Controlling- und Rechnungswesensysteme verwenden in der Regel die zeitliche Reihung während Logistik- und Produktionsplanungssysteme häufig zeitlich rollierend konzipiert sind.

Beschreibung von Attributen

Die Attribute können graphisch eingefügt werden. Die verwendete Symbolik ist in der Literatur verschieden, wir verwenden abgerundete Rechtecke oder Ellipsen für die Attributtypen und gegebenenfalls Kreise für die Attributwerte.

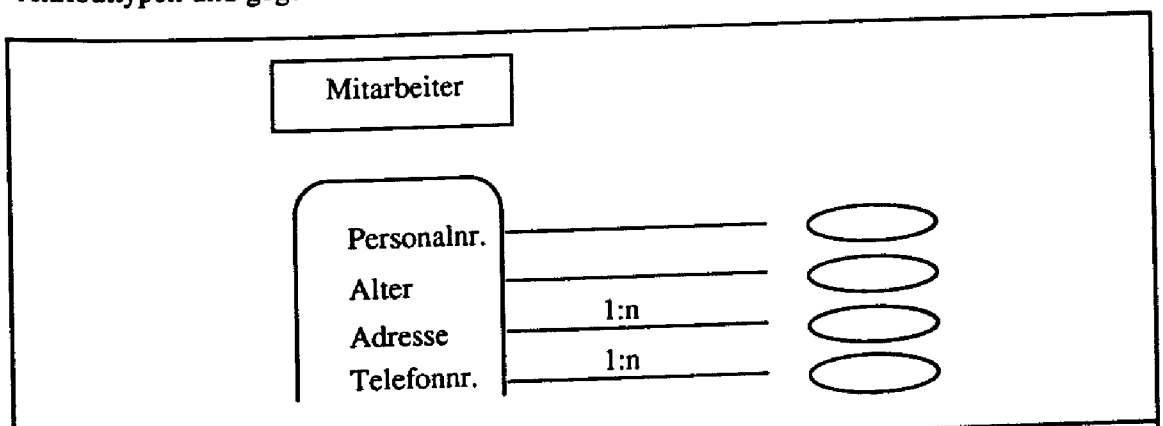


Abb. 3.2.1/34: Graphische Darstellung von Attributtypen und Attributwerten

Zwischen Attributtypen und Attributwerten können 1:1- und 1:n-Beziehungen auftreten.

Attribut-Typ: Wert	Erläuterung	Beispiel
1:1	der Attributtyp eines Objektes kann nur einen Wert annehmen	Ein Mitarbeiter hat nur eine Altersausprägung.
1:n	der Attributtyp eines Objektes kann mehrere Werte annehmen	Ein Mitarbeiter kann mehrere Adressen besitzen.

Abb. 3.2.1/35: Beziehungen zwischen Attributtypen und Attributwerten

Teilweise wird auf die Kennzeichnung der Attribute im ERM Diagramm verzichtet, und stattdessen werden diese verbal beschrieben.

Entity-Name: Arbeitspaket
Zweck (was? warum?) -> eine abgrenzbare Einheit eines Projektes -> eine durchzuführende Aktivität -> einen Verantwortlichen und Durchzuführenden eindeutig zuzuordnen **Eigenschaften** -> ein Arbeitspaket besitzt ein definiertes Start- und Endedatum -> ein Arbeitspaket verbraucht im Plan und im Ist eine bestimmbare Menge an Arbeitszeit **Attribute** - Bezeichnung des Arbeitspaketes - Beschreibung - Plan-Start-Datum - Plan-Ende-Datum - Plan-Arbeitszeitbedarf - Ist-Start-Datum - Ist-Ende-Datum - Ist-Arbeitszeitbedarf - Verantwortlicher - Durchführender

Abb. 3.2.1/36: Beispiel für die textuelle Beschreibung eines Entities und seiner Attribute (vgl. Schuldt, (1987), S. 235f.)

3.2.1.2.4. Interdependenzen

In einem Datenmodell bestehen zwischen den Elementen Interdependenzen, die zur Bildung neuer Datenelemente führen können. Beispielsweise entsteht das Attribut "Kundenumsatz" erst dann, wenn der Kunde ein Produkt kauft.

Komplexe Hilfsmittel bemühen sich, auch die Rückwirkungen dieser Strukturelemente zueinander zu erfassen (vgl. Hull/King (1988)).

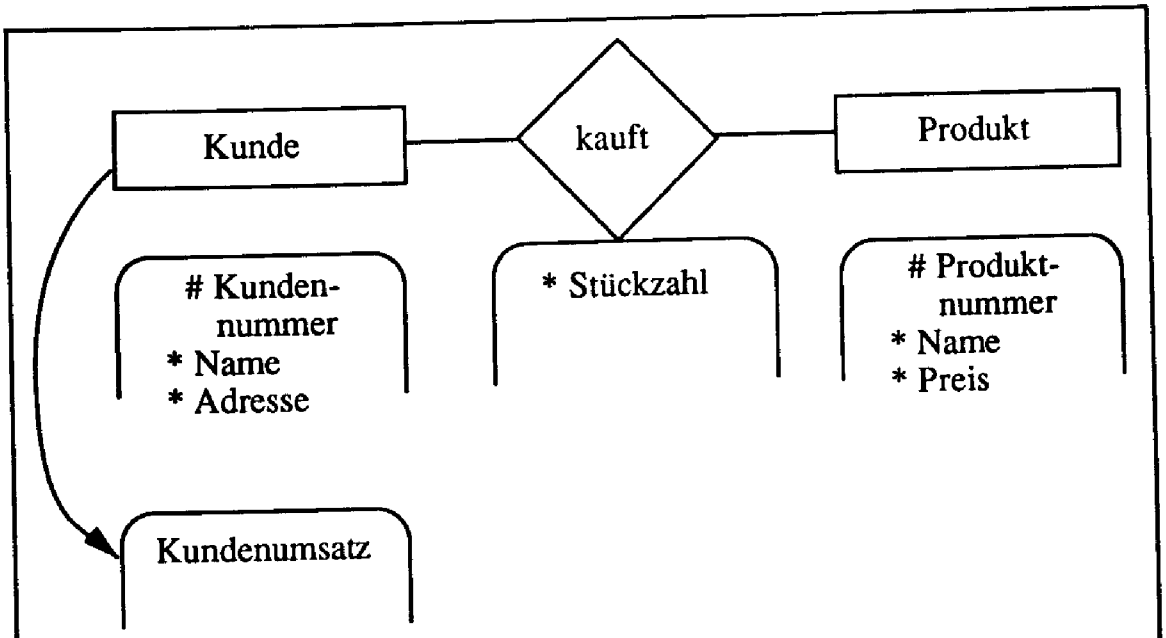


Abb. 3.2.1/37: Ableitung eines Attributs aus Attributen zweier (fremder) Beziehungen bzw. Objekte)

Es werden folgende abgeleitete Datenmodellkomponenten unterschieden:

Abgeleitetes Element	Kennzeichen	Beispiel
Abgeleitetes Attribut	Ableitung eines Attributtyps/einer Attributausprägung aus der Existenz objektfremder Attribute	Kundenumsatz
Abgeleiteter Objekttyp	Ableitung eines Objekt-Subtyps aus Merkmalsausprägungen der Attribute (Beispiel: Preis des Produktes ≤ 0)	Musterkunde/Gutschrift-kunde
Abgeleiteter Beziehungstyp	Ableitung eines Beziehungs-Subtyps aus Merkmalsausprägungen der Attribute (Beispiel: Lieferadressat ≠ Rechnungsadressat)	Streckengeschäft

Abb. 3.2.1/38: Typen abgeleiteter Datenmodellelemente

Zur Ableitung solcher Datenmodellelemente bedarf es entweder der Definition expliziter Ableitungsregeln (z. B. Kundenumsatz = Stückzahl des Kaufs * Preis des Produkts) oder impliziter Strukturierungsregeln im Datenmodell.

Eine andere Art von Interdependenz resultiert aus der Zuordnung von Datenobjekten zu flexibel gestalteten Strukturen. So wird beispielsweise das Datenobjekt "Apotheke Max Müller, Rostock" anhand seiner Attribute einmal nach der Regionalstruktur dem Markt "Mecklenburg-Vorpommern" zugewiesen, nach der Produktstruktur der Sparte "Pharma" und nach der Vertriebsstruktur dem Vertriebsweg "Handel". Diese Strukturobjekte sind im Zeitablauf variabel; beispielsweise gehörte die Apotheke bis 1989 in der Regionalstruktur noch zur DDR und zum RGW, wurde dann aus dem RGW ausgegliedert und nach Bildung des Bundeslandes "Mecklenburg-Vorpommern" dem neu gebildeten Strukturobjekt zugeordnet.

3.2.1.2.5. Statische Integritätsbedingungen

Integritätsbedingungen (integrity constraints) beschreiben die zulässigen Zustände eines Datenmodells und später einer Datenbank.

Integritätsbeziehungen existieren grundsätzlich in statischer und dynamischer Hinsicht. Aus statischer Sicht bestehen folgende Integritätsbedingungen:

Integritätsbedingungen	Typen	Beispiel
Attributintegrität	Attributexistenz	Sobald ein Kunde eine ausländische Adresse angibt, wird ein Attribut „Zollsatz" eingeführt.
	Attributausprägung (Domänenintegrität)	Notenskala 1, 2, 3, 4, 5, 6
Objektintegrität	Disjunktion - bestimmte Objektsubtypen schließen sich aus	Ein Mitarbeiter kann nicht gleichzeitig Lieferant im Datenobjekt „Marktpartner" sein.
	Existenz	Sobald ein Kunde ein Produkt kauft, wird ein Datenobjekt „Auftrag" generiert.
Beziehungstyp	Beziehungstyp	1 : 1 - wenn ein Mitarbeiter in einer Abteilung „arbeitet", kann er dies nicht noch in einer anderen tun. 1 : n - in einer Abteilung sind mehrere Mitarbeiter beschäftigt, ein Mitarbeiter gehört jedoch zu genau einer Abteilung
	Existenz	Sobald ein Datenobjekt „Kunde" existiert, muß auch die Beziehung „kauft" zu einem „Produkt" existieren.

Abb. 3.2.1/39: Statische Integritätsbedingungen

3.2.1.3. Konstruktionshilfsmittel

3.2.1.3.1. Beispiel 1: "Entity Relationship Modell"

Das Entity Relationship Model (ERM) von Chen (1972) ist eine aufgrund seiner Einfachheit und Einsichtigkeit sehr verbreitete Konstruktionshilfe. Die Konstruktions-Weltsicht des ERM besteht aus Objekt-Typen und Beziehungs-Typen, denen jeweils Attribute zugeordnet werden.

(1) Datenobjekt-Typ (Entity-Type)

Im ERM werden als „Entities" abgrenzbare Objekte (Gegenstandstypen) des (realen oder gedachten) Realitätsbereiches verstanden. Das ERM betrachtet diese Objekte nicht auf der Elementebene („Max Müller"), sondern auf der Typebene (KUNDE „Max Müller")

Datenobjekte sind im ERM Typen von Individuen, Gegenständen oder Konzepten. In der Regel lassen sich damit die Gegenstandstypen durch Substantive bezeichnen.

Chen unterscheidet unabhängige Datenobjekte (Kern-Entity) von abhängigen Objekten. Das Kern-Entity ist eine immer eindeutig identifizierbare Einheit, die auf einem Speichermedium

aufgrund eines Identifikationsmerkmals darstellbar ist und zwar unabhängig von der Existenz anderer Datenobjekte.

Beispiel:
Mitarbeiter, Maschinen, Produkte

In der Regel werden in einem Unternehmen kaum mehr als 20 Kernentitätstypen vorkommen.

Abhängige Datenobjekte sind ebenfalls immer eine eindeutig identifizierbare Einheit und auf einem Speichermedium mit Identifikationsmerkmal darstellbar, jedoch sind sie abhängig von der Existenz eines anderen Entities. Abhängige Datenobjekte gibt es in verschiedenen Formen.

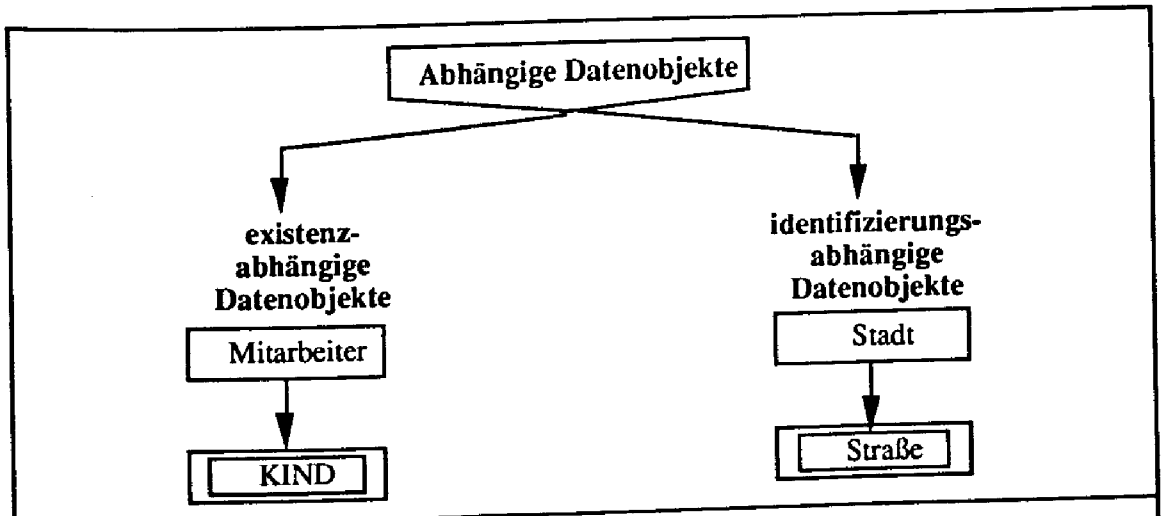


Abb. 3.2.1/40: Arten abhängiger Datenobjekte

Existenzabhängige Datenobjekte hängen davon ab, daß ein starkes (unabhängiges) Datenobjekt existiert. Im Beispiel heißt das, daß die Kinder eines Mitarbeiters für das Unternehmensdatenmodell nur so lange von Interesse sind, wie der Mitarbeiter im Unternehmen beschäftigt ist. Identifizierungsabhängige Datenobjekte lassen sich nur über ein unabhängiges Datenobjekt bestimmen.

Im ERM werden starke und schwache Entity-Typen unterschieden. Starke Entity-Typen lassen die Identifizierung eines Entities aufgrund seiner Attributausprägungen zu; bei schwachen Entity-Typen ist eine solche (isolierte) Identifizierung nicht möglich. In diesen Fällen muß der Primärschlüssel des schwachen Entity-Typs um den Primärschlüssel eines damit in Beziehung stehenden starken Entity-Typs ergänzt werden.

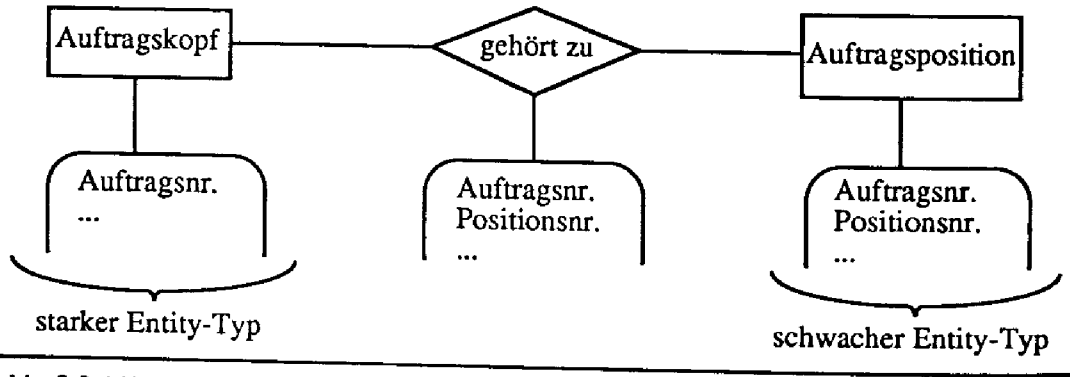


Abb. 3.2.1/41: Starke und schwache Entity-Typen im ERM (vgl. Sinz (1990), S.21)

Jedem Datenobjekt lassen sich Attribute zuordnen. Schlüsselattribute sollen ein Datenobjekt eindeutig kennzeichnen, ihm einfach im laufenden Betrieb zuzuordnenbar sein und möglichst wenig Speicherplatz (Kürze) verbrauchen. Werteattribute kennzeichnen ein Datenobjekt inhaltlich durch bestimmte Werteausprägungen. Sie werden dem Datenobjekt durch bestimmte funktionale Vorschriften zugeordnet, die

- auf Operationen basieren (z. B. Meßoperationen) und Eingabe-Transaktionen im Informationssystem bedingen oder
- auf Operationen im Informationssystem basieren (z. B. Berechnungen) und somit Verarbeitungstransaktionen bedingen (berechnete Attributausprägungen).

Berechnete Attributausprägungen lassen sich durch Operationen im Informationssystem jeweils neu rekonstruieren, so daß zwischen Berechnungsaufwand und Speicherplatzanforderungen abzuwägen ist.

Die zulässigen Ausprägungen der Attribute (Wertebereiche) lassen sich durch sogenannte Domänen definieren.

(2) Beziehungs-Typ (Relationship-Type)

Im ERM werden als Beziehungen beobachtbare und beschreibbare Zusammenhänge zwischen Objekten des Realitätsbereichs verstanden. Auch Beziehungen werden nicht auf der Elementebene sondern auf der Typebene betrachtet.

Beziehungen können entweder reale Zusammenhänge abbilden (z. B. Aktivitäten, die in der Regel durch Verben beschreibbar sind) oder zur Strukturierung des Gegenstandsbereiches dienen (beschreibbar z. B. durch "gehört zu", "besteht aus").

Die Beziehungs-Typen werden mit einem eigenen Namen versehen (es bietet sich dafür die Benutzung von Verben an) und mit ihrer Kardinalität bzw. Komplexität (1:1, 1:m, n:1 und n:m) gekennzeichnet, die angibt, wieviele Datenobjekte an einer Beziehung jeweils beteiligt sind.

Beziehungen zwischen Objekten können Attribute besitzen.

Beziehungstyp	Erläuterung	Beispiele
Logische Unterordnung (is a)	Logische Unterordnung zwischen Datenobjekten	Rohstoff ist ein Stoff
Hierarchische Unterordnung (is part of)	Existenznotwendige (hierarchische) Beziehung für existenzabhängiges Datenobjekt	Auftrag besteht aus Auftragspositionen

Abb. 3.2.1/42: Gebräuchliche Beziehungstypen im ERM

Für Beziehungen zwischen den Attributen der Objekte bietet das ERM keine explizite Modellierungshilfe an.

Ein Entity-Relationship-Modell kann graphisch beschrieben werden mit sogenannten ERM-Diagrammen:

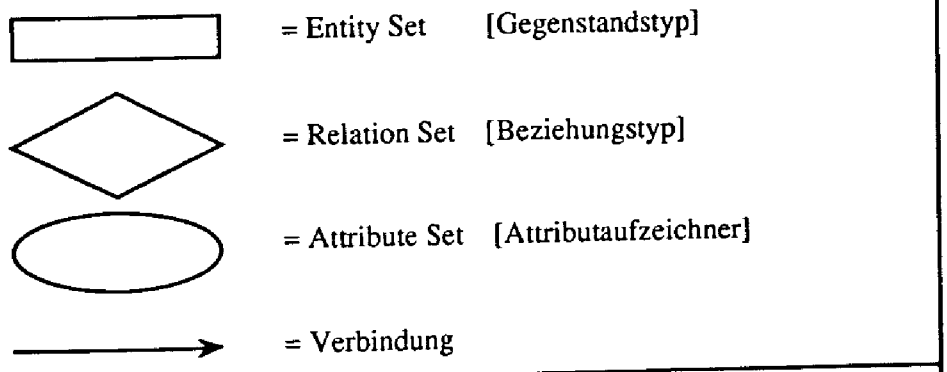


Abb. 3.2.1/43: Graphische Grundelemente des Entity-Relationship-Modells

Alternativ oder ergänzend kann auch eine strukturierte verbale Beschreibung erfolgen. Lindgreen (1983) schlägt ein Hilfsmittel vor, daß dazu dient, den ERM-Rahmen zu erweitern. Er beschreibt die verschiedenen Datenelemente durch identifizierende und klassifizierende Angaben und durch deren Eigenschaften.

Beschreibungs -merkmal		Kennzeichen	Merkmals-typ	Anzahl pro Daten-element	Umgangs-sprachliches Beschrei-bungsmittel
Name		Einmalige und einheitliche Kennzeichnung	Identifi-zierend	1	Substantive bei Entities, Verben bei Beziehungen
Klassifizierung (subset indications)		beschreiben über welche Konstruktionsoperatoren das Datenelement entstanden ist.	klassifizie-rend	1	
Eigenschaften (characteristic properties)		beschreiben mit Namen, Art und Wertedomäne die Attribute	deskriptiv	m+n	
	Attribute	beschreiben die einem Datenelement zugeordne-ten statischen Eigen-schaften über - Namen - Wertebereich - Meßvorschriften - Werteausprägungen - zulässige Operationen	statisch deskriptiv	m	Adjektive
	Rollen	beschreibt die einem Datenelement zugeordne-ten Beziehungen zu anderen Datenelementen	dynamisch deskriptiv	n	Verben

Abb. 3.2.1/44: Datenelemente nach Lindgreen (1983)

Für jedes Entity werden die Attribute und die Beziehungen in folgender Notation festgehalten:

Entity (
Attribut 1: Datentyp
...
Attribut n: Datentyp
Beziehung 1: zugeordnetes Entity i Beziehungstyp,
...
Beziehung m: zugeordnetes Entity j Beziehungstyp),

Attribute und Beziehungen können in beliebiger Reihenfolge deklariert werden. Datentypen sind z. B. real, integer, Text oder money. Bei den Beziehungstypen steht ein (+) für eine 1:n-Beziehung, ein (*) für eine 0:n-Beziehung.

Entity-Typen

Kunde (Name: Text bekommt: Lieferung (+)),	Lieferung (ist_für: Kunde enthält: Verkauf (+) ist_am: Tag Betrag: real),	Verkauf (Menge: real gehört zu: Versand betrifft: Produkt Verkaufs-Wert: Geld),
Produkt (Name: Text Bestellwert: real Preis: money Bestellmenge: real verkauft von: Verkauf (*)),	Tag (Tagesdatum: integer ist_im: Monat hat: Lieferung (*)),	Monat (Monatsbezeichnung: Text Monatsnr.: integer enthält: Tag (+) ist_im: Jahr),
Jahr (Jahreszahl: integer enthält: Monat (+));	Bemerkung: (*) 0:n-Beziehung (+) 1:n-Beziehung (ohne Angabe) 1:1-Beziehung	

Abb. 3.2.1/45: Verbale Beschreibung eines ERM (vgl. Lindgreen (1983) S. 96).

Attribute und deren Ausprägungen können durch Beschreibung und Restriktionen näher spezifiziert werden:

Objekt	Beispiel	Erläuterung:
Attributausprägung	Lagerwert (Produkt) = Lagerbestand * Preis	Lagerbestand und Preis sind Attribute des Entities "Produkt".
	Verkaufs-Wert (Verkauf) = Menge * Preis	Preis ist ein Attribut des Entities Produkt. Eine solche Zuordnung ist möglich, wenn sich der Preis eindeutig bestimmen läßt.
	Betrag (Lieferung) = SUM OVER enthält Verkauf	Die Funktion SUM ... OVER ... bestimmt, daß alle Attributausprägungen über die angegebene Regel aufsummiert wird.
Restriktionen		
Attributrestriktionen bestimmen eine Menge von Attributen, die durch den Operator WHERE und einen logischen Ausdruck spezifiziert werden.	Meier =: Kunde WHERE Name = "Meier"	Kunde mit dem Namen Meier
	Großauftrag =: Versand WHERE (Verkaufs-Wert > 50000 OR Menge > 1000)	Großauftrag ist abgeleitet von Versand mit Verkaufs-Wert ist größer als 50000 oder Menge ist größer als 1000.
Objektrestriktionen wenden auf ein Entity einen logischen Ausdruck an, der durch den Operator THAT getrennt wird.	Spezialauftrag =: Versand THAT enthält Großauftrag	Spezialauftrag ist abgeleitet von Versand, der über die Beziehung "enthält" auf einen Großauftrag abgeleitet werden kann.
	Meier_Auftrag =: Versand THAT ist_für Meier	Meier_Auftrag ist abgeleitet von einem Versand, der über die Beziehung "ist_für" auf Meier abgeleitet werden kann.
anonyme Restriktionen setzen sich aus vorher definierten Restriktionen zusammen (müssen nicht wie im Beispiel konkrete Ausprägungen enthalten).	August_91_Großauftrag_ Meier_Auftrag =: Großauftrag THAT gehört zu Meier_Auftrag THAT ist_am Tag WHERE (Monatbezeichnung = "August" AND Jahreszahl = 1982)	August_91_Großauftrag_ Meier_Auftrag ist abgeleitet von Großauftrag über die Beziehung "gehört zu" auf Meier_Auftrag, über "ist_am" auf Tag. Der Tag wird spezifiziert durch die Monatsbezeichnung "August" und die Jahreszahl 1982.

Abb. 3.2.1/46: Beschreibung von Attributausprägungen nach Lindgreen (1983)

Es existieren neben dem ursprünglichen Ansatz von Chen (1972) inzwischen eine ganze Reihe von Weiterentwicklungen des ERM (vgl. Chen (1981)).

	binäre ERM	Entity-Association-View	Entity-Association-Property-View
Kennzeichen/ Elemente	- Entities - binäre Relationen (1:1) - keine Attribute	- Entities - Relationen (n-stellig) - keine Attribute	- Entities - Relationen (1:1, 1:m, n:m) - Attribute (bei Entities und Relationen
Kennzeichen	- nicht binäre Relationen müssen als Entities modelliert werden - Eigenschaften von Objekten müssen als separate Entities modelliert werden - erfordert aufgrund von Binärstruktur keinen Normalisierungsprozeß	- Eigenschaften (Attribute) bilden separate Entities und werden durch Relationen mit den Objekten verknüpft	- Attribute werden den Entities und (strittig) den Relationen zugeordnet - erfordert Normalisierungsprozeß
Autoren	Abrial (1974) Bracki/Paolini/Pelegatti (1976) Berild/Nachmens (1977)	Brentmann/Falkenberg/Mauer (1979) Nijssen (1977)	Chen (1976) Pirotte (1977) Schmid (1979) Tsichtritzis/Lochowsky (1978)

Abb. 3.2.1/47: Verschiedene Typen von Entity-Relationship-Modellen (vgl. Bubenko (1980))

Diese Typen unterscheiden sich zum einen darin, ob nur binäre oder auch n-stellige Beziehungen zugelassen sind und zum anderen, ob diese Beziehungen Attribute besitzen dürfen oder nicht.

Wie Chen (1981) untersucht hat, sind diese Ansätze generell gleichwertig. Sie bieten

-> teilweise einen umfangreicheren graphischen Symbolvorrat als das ursprüngliche Entity-Relationship-Modell,
-> eine bessere Unterstützung der Konstruktionsoperatoren,
-> eine strengere formale Anbindung an logische Datenmodelle, speziell das relationale Modell und sichern zum Beispiel im Konstruktionsprozeß die 1. bis 4. Normalform.

Die formalisierte Beschreibung der ERM unterscheidet sich durch die oft implizite Anlehnung an ein logisches Datenmodell.

Beispiele:
ERM, die auf das logische Relationenmodell zielen, lassen oft nur eine Wertausprägung pro Attribut zu und zwingen damit den Konstrukteur, ggf. mehrere Attributtypen für den gleichen semantischen Ausdruck zu verwenden. Hat beispielsweise ein Kunde zwei Lieferanschriften, so müssen diese in zwei Attributtypen gehalten werden.

Es wird versucht, die Leistungsfähigkeit des Entity-Relationship-Gedankens durch Einführung neuer semantischer Konzepte zu steigern. Dabei werden z. B. Rollen den Objekten oder Beziehungen zugewiesen.

Beispiel: Ein bestimmter Marktpartner kann zu einem Zeitpunkt „Kunde" sein, zu einem anderen Zeitpunkt aber die Rolle „Lieferant" einnehmen.

3.2.1.3.2. Beispiel 2: Objekttypenmodell

Der Objekttypen-Ansatz (nach Wedeking/Ortner) verwendet ebenfalls Informationsobjekte auf der Typebene (Objekttypen) und Objekteigenschaften (Attribute). Beziehungen werden durch Kanten dargestellt; n:m-Beziehungen zwischen Objekttypen werden zu neuen Objekttypen zusammengefaßt.(vgl. Ortner (1989)).

Neu und wesentlich gegenüber dem ursprünglichen Entity-Relationship-Modell ist die explizite, auch graphische Unterstützung der Konstruktionsoperatoren und die Unterscheidung von Beziehungstypen.

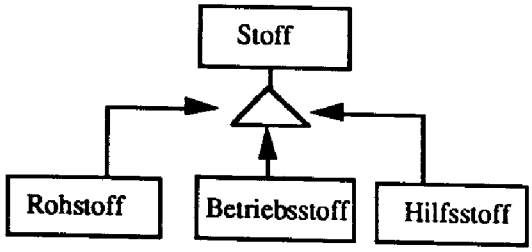
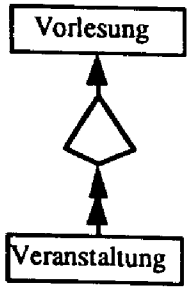
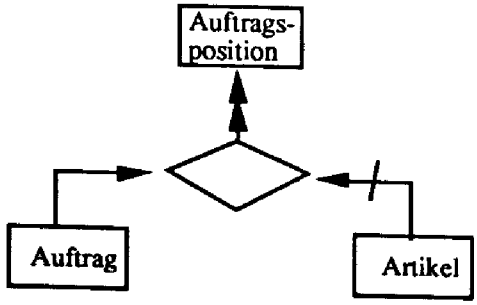
Graphische Darstellung	*Konstruktionsoperator*
	**Generalisierung/ Spezialisierung** hier: - Stoffe sind entweder Roh-, Hilfs- oder Betriebsstoffe
	**Gruppierung/ Assoziation** hier: - Eine Vorlesung besteht aus mehreren Veranstaltungen
	**Konnexion/ Verknüpfung** hier: - Jeder Auftrag betrifft mehrere oder einen Artikel - Ein Artikel kann in mehreren Aufträgen bestellt worden sein.

Abb. 3.2.1/48: Graphische Darstellung der Konstruktionsoperatoren nach Wedekind/Ortner (vgl. Ortner/Söllner (1989))

Hinsichtlich der Beziehungen unterscheidet das Objekttypenmodell das Beziehungsverhältnis, die Beziehungshäufigkeit und die Beziehungswirkung. Ein Objekttyp steht danach in einem genau zu spezifizierenden „Beziehungskomplex" zu anderen Objekttypen.

Beziehungsverhältnis	ein Objekt n Objekte ($1 \leq n$)
Beziehungshäufigkeit	für alle Objekte für einige Objekte ($0 \leq n$) für genau ein Objekt
Beziehungswirkung	Generalisierung/ Spezialisierung Gruppierung/ Assoziation Konnexion/ Verknüpfung

Abb. 3.2.1/49: Beziehungen in der Objekttypenmethode

3.2.1.3.3. Beispiel 3: Semantic Data Model

Semantische Datenmodelle erweitern die Konstruktionshilfsmittel durch eine umfassendere Betrachtung der Beziehungen zwischen Datenobjekten. Dazu werden Ansätze verwendet, die gewisse Ähnlichkeiten mit der objektorientierten Programmierung besitzen.

1. Klassenkonzepte, d.h. die Objekte, werden eingeordnet in Objektklassen (hierarchische Generalisierung)
2. Die Oberklassen vererben bestimmte Eigenschaften (Attribute) an Unterklassen.

Als Beispiel sei das Semantic Data Model (vgl. Hammer/Mc Leod (1977), vgl. auch Smith/ Smith (1977)) betrachtet.

Das SDM ist ein klassenlogischer Ansatz, bei dem die Objekte nach ihrer spezifischen Bedeutung klassifiziert, gruppiert und deren Beziehungen untereinander analysiert werden. Das SDM ist nicht nur ein Konstruktionshilfsmittel, sondern es enthält auch eine Daten-definitionssprache und ein Dateninteraktionskonzept mit einer Benutzerschnittstelle und eine Datenmanipulationssprache, die auf einer objektorientierten Klassenlogik basiert (vgl. Ortner (1985), S. 25).

Ein SDM-Datenmodell besteht aus einer Ansammlung von Klassen, die durch Attribute gekennzeichnet werden und jeweils Objekte umfassen. Es werden zwei Typen unterschieden: Eine Basisklasse ist eine Klasse, die unabhängig von allen anderen in die Datenbasis eingeführt wurden. Sie umfaßt die realen Objekte des Gegenstandsbereiches (Grundobjekte), z. B. Maschinen, Personen, Räume. Nichtbasisklassen wurden über andere Klassen eingeführt und symbolisieren deren Beziehungen zueinander. Dabei werden zwei Typen von Nichtbasisklassen unterschieden:

Die „Teilklassenbildung" (subclass connection) bildet entsprechend dem Konstruktions-operator „Spezialisierung" Teilmengen einer Oberklasse, beispielsweise wird die Klasse MARKTPARTNER in die Teilklassen „KUNDE, LIEFERANT UND MITARBEITER" zer-

legt. Die „Gruppenklassenbildung" (grouping connection) entspricht der Generalisierungsoperation.

Typ	Subtyp	Beschreibung	Beispiel
Klasse		werden durch eine „Klassenbeschreibung" in ihrer Natur, ihren Eigenschaften und ihrer Nutzanwendung beschrieben.	Schiffe eines Typs: Frachter
	Basisklassen	sind unabhängig von der Existenz anderer Klassen	Frachter
	Nichtbasisklassen	werden in Abhängigkeit von mindestens einer Basisklasse definiert	Containerfrachter
Objekte = beobachtbare oder konstruierte Erscheinungen der Anwendung		beobachtbare oder konstruierte Erscheinungen der Anwendung	
	konkret beobachtbare Objekte	empirische Objekte der Realität	Bestimmte Schiffe, Hafenanlagen
	abstrakte Objekte	werden durch Anwendung bestimmter Operatoren (Generalisierung, Aggregation) konstruiert	Schiffskonvoi
Ereignisse	Zeitpunkt-Ereignisse Zeitraum-Ereignisse	Namen von Objekten oder Ereignissen	
Attribute	Mitgliedsattribute (member attribute)	Eigenschaften jedes Mitgliedes einer Klasse	Klasse Schiffe: Name, Heimathafen, Tragfähigkeit
	Klassenattribute (class determined attribute)	sind allen Mitgliedern einer Klasse gemeinsam und haben jeweils den gleichen Wert	Anzahl der Container
	Klasseneigenschaften (class attribute)	beschreiben die Eigenschaften der Klasse an sich	Klasse Schiffe: Anzahl der vorhandenen Schiffe

Abb. 3.2.1/50: Typisierung des SDM

Das SDM verzichtet auf ein explizites Element für Beziehungen. Beziehungen zwischen Elementen verschiedener Klassen werden stattdessen durch Attribute (Schlüssel) gebildet.

Neben dem SDM existieren eine Reihe weiterer Ansätze, von denen in der folgenden Tabelle einige überblicksweise charakterisiert werden.

	Erweitertes relationales Modell	**Semantic Data Modell**	**Semantic Hierarchy Modell**	**Event Model**
Autor	Codd (RM/T)	Hammer/ McLeod (SDM)	Smith/Smith (SHM)	King/Mc Leod
Eigenschaft/ Beschreibung	Eigenschaft	Eigenschaft	Attribut	Attribut
Objekt	Entity	Element oder Entity	Objekt	Objekt
einfaches Objekt	Ausprägung einer Eigenschaft	Bezeichnung des Wertes	einfaches Objekt	Objektdeskriptor
aggregiertes Objekt	Entity	Element oder Entity	aggregiertes Objekt	abstraktes Objekt
Objektmenge	abgegrenztes Element	Benutzer-kontrollierte Gruppe	vereinigtes Objekt	-
Klasse	Entity-Type	Klasse	Objektklasse	Typ
Metaklasse	-	ausdrucksdefinierte oder ausgezählte Gruppierung	-	-

Abb. 3.2.1/51: Terminologien in semantischen Datenmodellen (vgl. Urban/Delcambre (1986), Seite 388).

3.2.1.4. Konstruktionsoperatoren

Konstruktionsoperatoren sollen den Konstruktionsprozeß durch Vorgabe bestimmter formallogischer Operationen strukturieren, die auf die Informationselemente der Wirklichkeit anzuwenden sind. Konstruktionsoperatoren geben also Hinweise, wie Objekte und/oder Beziehungen der modellierten Realität in die Struktur eines Datenmodells zu überführen sind (vgl. Ortner, 1985; Scheer (1988) S.16ff, Schönfelder).

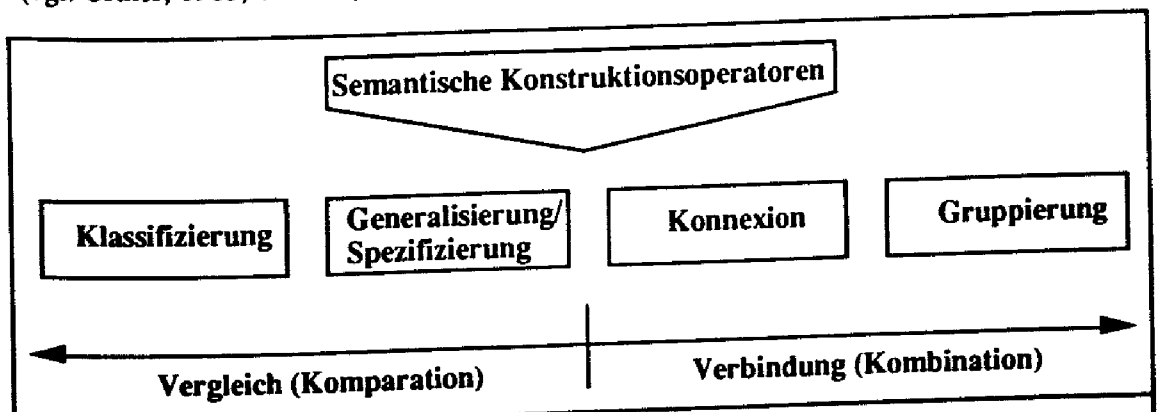


Abb. 3.2.1/52: Semantische Konstruktionsoperatoren

(1) Klassifizierung

Bei der Klassifizierung werden Objekte einer realen oder hypothetischen Welt zu Klassen zusammengefaßt, denen ein Begriff zugeordnet wird. Dazu wird nach gleichartigen Elementen gesucht, d.h. nach Elementen, die sich mit den gleichen Attributen beschreiben lassen. Zwischen den Objekten sind hinsichtlich der Attributmengen keine Unterschiede erkennbar, so daß ihnen Datenobjekte zugeordnet werden können.

Die Klassifizierung ist eine notwendige Aktivität bei der Bildung von Begriffstypen aus realen Objekten, d.h. beim Übergang von der Realebene zur Typebene.

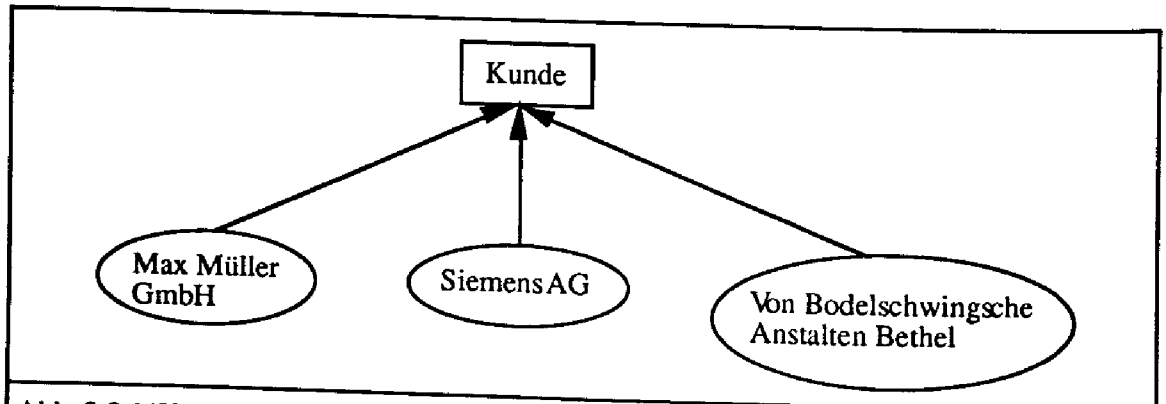


Abb. 3.2.1/53: Klassifikation von realen Objekten und Zuordnung eines Datenobjektes

(2) Generalisierung/Spezifizierung (auch Inklusion)

Bei der Generalisierung werden Objekte als Subtypen unter einem Supertyp zusammengefaßt (Teilmengen zu Obermengen) bzw. aus einer Obermenge werden Teilmengen abgeleitet (Spezialisierung) und es wird ein entsprechender Gattungsbegriff zugewiesen (vgl. Smith/Smith (1977a).

Beispiel:
Roh-, Hilfs-, Betriebsstoffe, Halbfertigfabrikate und Fabrikate zu STOFFEN
oder Objekttyp Kunde und Objekttyp Lieferant zum Objekttyp GESCHÄFTSPARTNER

Mit der Generalisierung/Spezialisierung wird das Abstraktionsniveau festgelegt. Im Deutschen kann man von Verfeinerung bzw. Verallgemeinerung sprechen.

Die Typisierung von Objekten erfolgt über den Vergleich von semantischen oder pragmatischen Eigenschaften der Datenobjekte und Beziehungen. Dieser Vergleich kann sich auf die Existenz oder auf die Werteausprägungen von Attributen erstrecken.

Bei der Spezialisierung vererbt der generalisierte Objekttyp seine Attribute auf die spezialisierten Objekttypen. Die spezialisierten Objekttypen können diese unverändert übernehmen oder auch verändern, sie können aber auch zusätzliche eigene Attributtypen erhalten.

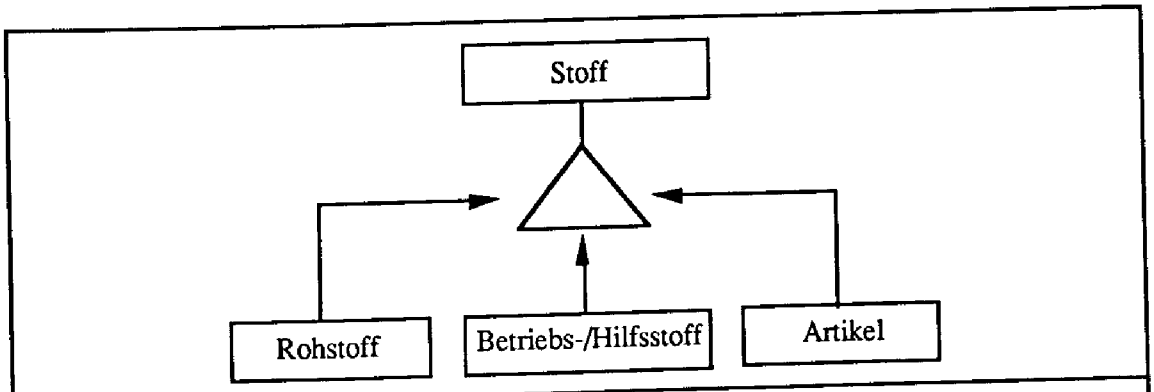


Abb. 3.2.1/54: Symbol der Generalisierung/Spezifizierung (nach Wedekind/Ortner)

Die Spezialisierung kann danach unterschieden werden, ob dabei überschneidungsfreie Teilmengen gebildet werden können oder nicht (vgl. Scheer (1991), S. 27).

SPEZIALISIERUNG	in disjunkte Teilmengen	in sich überschneidende Teilmengen
Beispiel	inländische/ausländische Lieferanten	* Lieferanten und Kunden * ein Lieferant kann auch gleichzeitig ein Kunde sein

Abb. 3.2.1/55: Typen der Spezialisierung

Die Generalisierung/Spezialisierung kann sich in analoger Weise auch auf Beziehungen erstrecken:

Beispiel:
liefert, zahlt, leistet wird zu LIEFERT

(3) Verknüpfung/Konnexion

Bei der Konnexion (in der Literatur wird auch von Aggregation gesprochen (vgl. Scheer (1990)) wird aus einzelnen Typen (z. B. Entities, Relationen, Attributen) ein neuer, realer oder abstrakter Typ abgeleitet, wobei jedes Teil eine spezifische Aufgabe für das Funktionieren des neuen Ganzen hat. Chen spricht von einem „Composite Entity Typ" (Chen (1985)). Diese abstrakten Datenobjekte entstehen im Konstruktionsprozeß und stehen gewissermaßen zwischen „Objekten" und „Beziehungen" (vgl. Smith/Smith (1977b)).

Beispiel:
Aus der Verbindung der beiden Objekte KUNDE und ARTIKEL entsteht das abstrakte Objekt KUNDENANFRAGE oder aus der Verbindung der Objekte Pille-Beipackzettel und Faltzettel entsteht der verkauffähige Artikel.

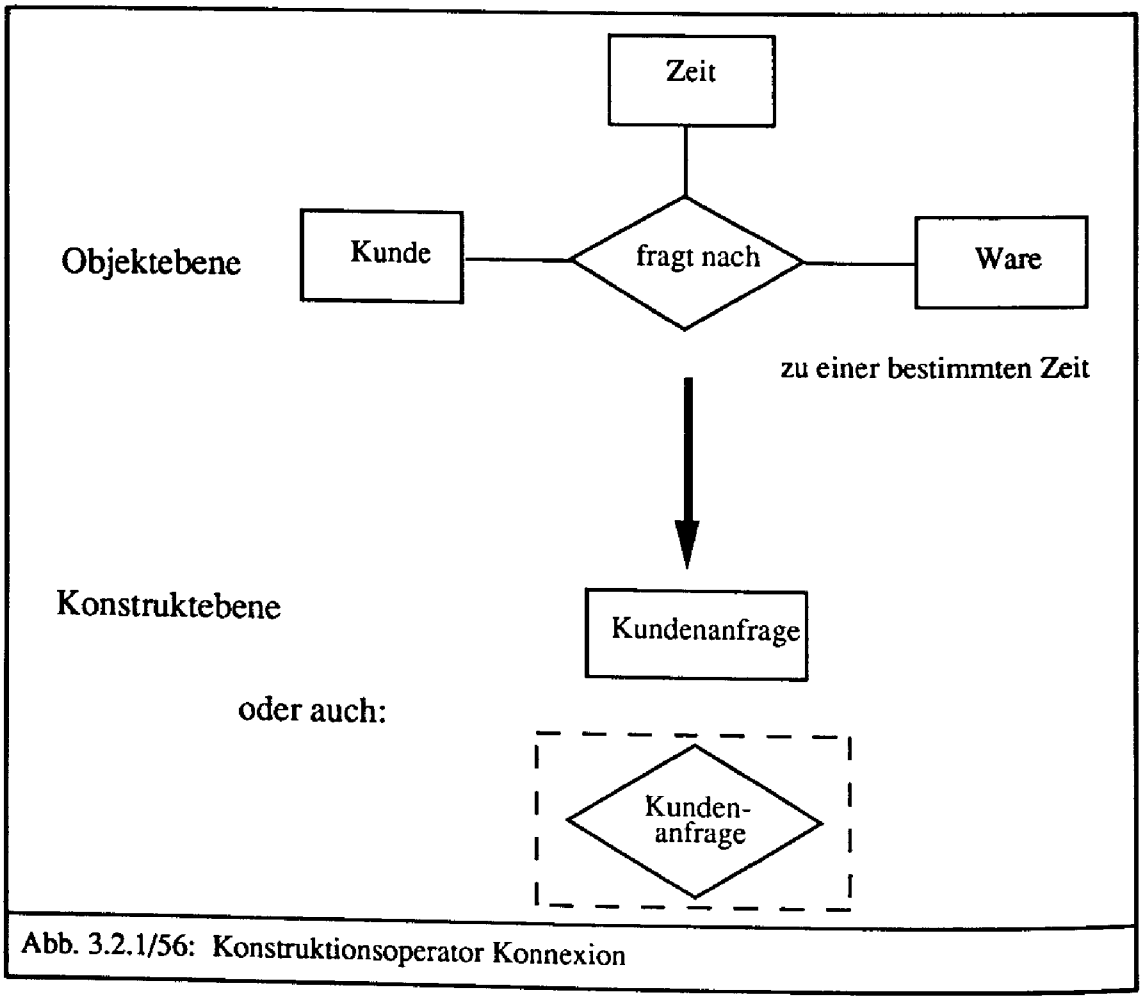


Abb. 3.2.1/56: Konstruktionsoperator Konnexion

Die Konnexion kann sich nicht nur auf die Beziehung zwischen zwei Objekten (Beziehungskonnexion), sondern auch auf Attribute und Objekte erstrecken.

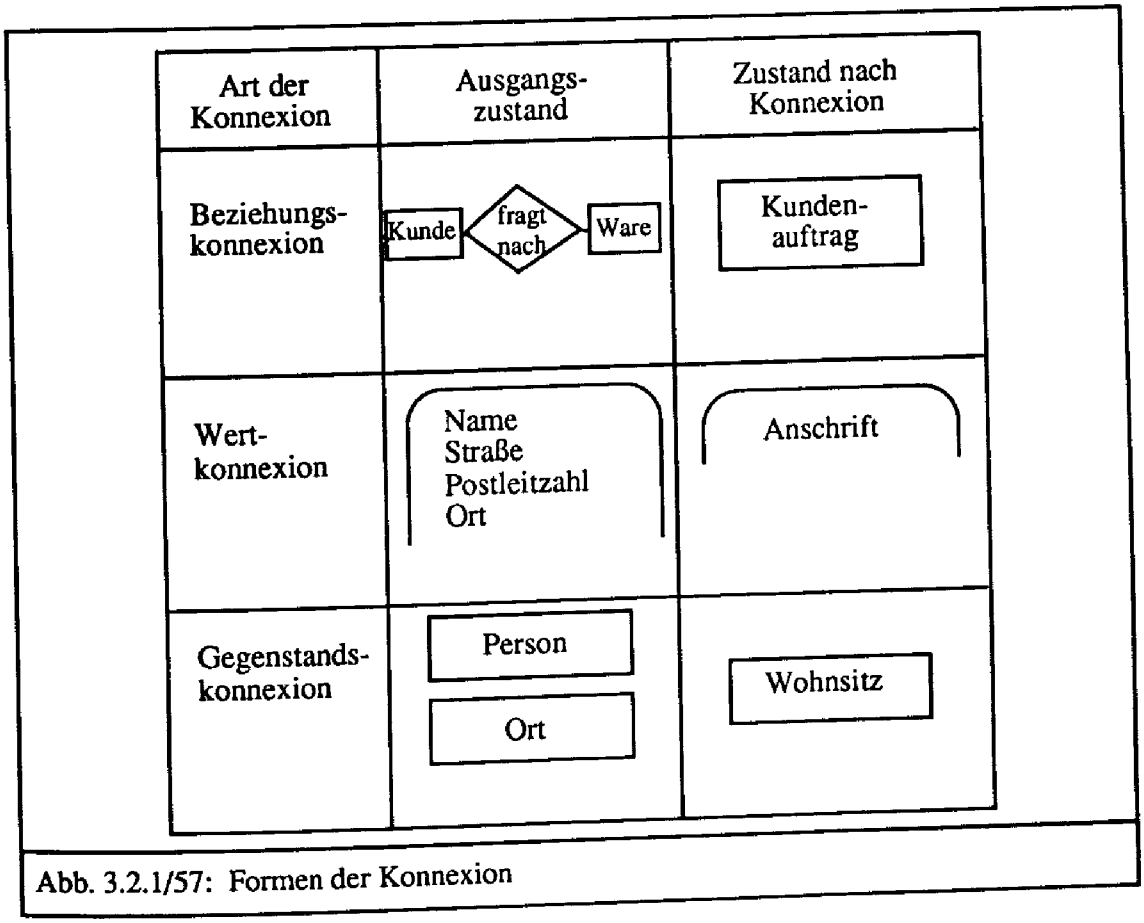


Abb. 3.2.1/57: Formen der Konnexion

(4) Gruppierung/Assoziation

Gruppen entstehen durch Zusammenfassung von gleichen oder verschiedenen Gegenständen zu einem relativen einheitlichen Ganzen. Beispielsweise werden Arbeitsplätze zu Abteilungen, Schüler zu Klassen oder Maschinen zu Maschinengruppen zusammengefaßt. Zwischen den Gegenständen einer Gruppe besteht teilweise Gleichheit, teilweise sachliche Abhängigkeit.

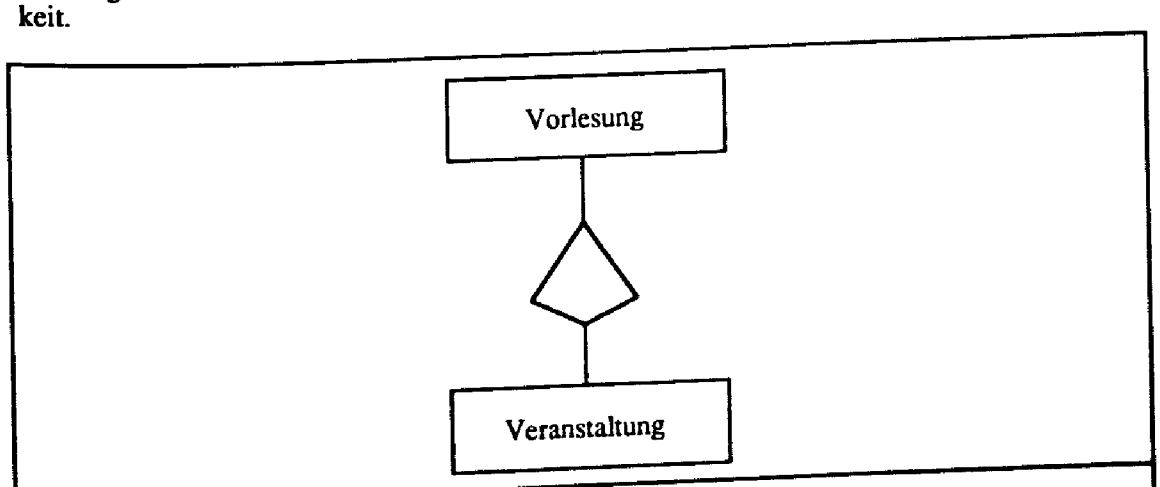


Abb. 3.2.1/58: Symbol der Gruppierung (nach Wedekind/Ortner)

Beispiel: Die verschiedenen Veranstaltungen sind Bestandteil einer gesamten Vorlesung.

(5) Identifizierung

Bei der Identifizierung wird ein Attributtyp spezifiziert oder konstruiert, der einen Beziehungs- oder Objekttyp eindeutig kennzeichnet (Zuweisung von Schlüsselattributen).

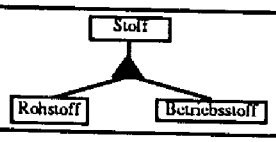
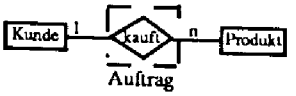
Konstruktions-operator	**Konstruktionsbefehl**	**Graphisches Symbol (mit erläuternden Beispielen)**	**Erläuterung**
Klassifizierung	IS-A-CLASS OF (instance of)		Bestimmte Elemente der Realität werden einem Oberbegriff zugewiesen.
Generalisierung /Spezialisierung	IS-A		Unterschiedliche Objekte einer Sprachebene werden auf der nächsthöheren Sprachebene zu einem abstrakteren Begriff zusammengefasst.
Assoziation/ Gruppierung	IS-ASSOSIATED-WITH (is-member-of)		Aus gleichartigen Objekten werden Gruppen gebildet und als spezieller Entitytyp spezifiziert.
Verknüpfung/ Konnexion	IS-PART-OF (property-of)		Aus der Beziehung unterschiedlicher Objekte entsteht ein neues Objekt.
Identifizierung	IS-IDENTIFIED-BY		Ein Attribut oder eine Teilmenge von Attributen identifiziert ein Datenobjekt oder eine Beziehung.

Abb. 3.2.1/59: Übersicht über semantische Konstruktionsoperatoren

Die Anwendung der Konstruktionsoperatoren führt jeweils zu einem neuen Datenmodell eines höheren (oder bei der Spezifizierung auch niedrigeren) Abstraktionsgrades, das jeweils hinsichtlich des Abstraktionsgrades zu kennzeichnen ist. Die verschiedenen Konstruktionshilfmittel unterstützen die Konstruktionsoperatoren in unterschiedlichem Maße.

<table>
<tr><th>Operator</th><th>Ortner</th><th>Smith/Smith</th><th>E.F. Codd</th><th>P.P.S. Chen</th><th>Hammer/
Mc Leod</th></tr>
<tr><td>Klassifikation
Prädikation</td><td>  </td><td rowspan="2">Generalization</td><td rowspan="2">Generalization</td><td> 
Entity Set</td><td rowspan="4">Entity
Classes

Interclass
Connections

- Subclass
connection
- Grouping
connection</td></tr>
<tr><td>Generalisierung
(Inklusion)</td><td>  </td><td rowspan="3"> 
Relationship
Set</td></tr>
<tr><td>Konnexion</td><td>  </td><td rowspan="2">Aggregation</td><td>Cartesian
Aggregation</td></tr>
<tr><td>Gruppierung
(Aggregation)</td><td>  </td><td>Umhüllende
Aggregation
(Cover
Aggregation)</td></tr>
</table>

Abb. 3.2.1/60: Vergleich verschiedener Konstruktionshilfsmittel (vgl. Ortner (1985), S. 26)

3.2.2. Dynamische Konstruktion

3.2.2.1. Kennzeichen, Ziele, Vorgehen

Kennzeichen

Im Zuge der dynamischen Datenbankkonstruktion wird das zulässige Verhalten der Datenelemente im Zeitablauf beschrieben. Datenobjekte, Beziehungen und Attribute werden durch zeitgebundene „Ereignisse" beeinflußt.

Ereignisse sind Vorgänge kurzer Zeitdauer, deren Auftreten an bestimmte Bedingungen und/oder bestimmte Zeitpunkte gebunden ist.

Ereignisse verändern den Zustand einer Datenbank und generieren eine Folge von Zuständen (Zustandsgeschichte).

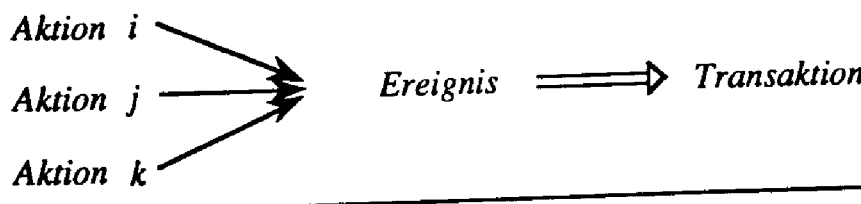


Abb. 3.2.2/1: Bestandteile eines Prozedurmodells (nach Brodie/Ridjanovic (1984))

Ereignisse lassen sich ebenfalls auf der Typebene definieren (event types).

Das konzeptuelle Datenmodell muß sowohl die statische Struktur als auch das dynamische Verhalten abbilden. Eine gute statische strukturelle Beschreibung erleichtert die Konstruktion zulässiger Transaktionen; auf der anderen Seite ist eine umfassende strukturelle Konstruktion der Datenobjekte und Relationen nur möglich, wenn die denkbaren dynami-

schen Transaktionen umfassend konstruiert und logisch eindeutig beschrieben sind (vgl. Sakai (1989), S. 111, Brodie/Ridjanovic (1984)).

Ziel einer dynamischen Konstruktion ist es, die vielfältigen dynamischen Integritätsanforderungen in einer Datenbank abzubilden. Integritätsbedingungen beschreiben die zulässigen (inhaltlichen) Operationsfolgen auf die Datenbasis. Eine zulässige Folge transformiert einen bezüglich der Realität widerspruchsfreien Zustand in einen neuen, ebenfalls widerspruchsfreien Zustand.

Dynamische Integritätsbedingungen werden im Prozeß der Modellierung und Schemabildung weiterentwickelt zu (logisch zulässigen) Transaktionen.

Konstruktionsebene	**Konstruktions-instrument**	**Beispiele**
Ereignisebene (konzeptuelles Schema)	Aktionen	Pre-Konditionen je Datenobjekt/Beziehungen Post-Konditionen (Randbedingungen, Ableitungsregeln)
Transaktionsebene (externes Schema)	Transaktionen	INSERT Altering DELETE Operationen UPDATE FIND Retrieval CREATE Operationen REQUEST

Abb. 3.2.2/2: Schema bei der dynamischen Konstruktion

Die Aktionen der konzeptuellen Ebene sind

-> vollständig und widerspruchsfrei für alle Datenobjekte und Beziehungen so präzise zu beschreiben, daß die dynamische Integrität der Datenbank zu jedem Zeitpunkt gewährleistet ist,
-> präzise durch Angabe der Wirkungen auf Attribute von Objekten/Relationen durch Angabe der Pre-Konditionen und der Post-Konditionen zu kennzeichnen.

Die Aktionen sind in Transaktionen des externen Schemas (= der Benutzerebene) umzusetzen. Eine Transaktion ist eine anwendungsorientierte Operation von Anwendungsprogrammen oder menschlichen Benutzern, eines oder mehrere Datenobjekten verändert.

Dynamische Konzepte zur Objekttypenkonstruktion beschäftigen sich mit der „Lebensgeschichte" (life history) oder auch dem „zeitlichen Verhalten" (behaviour) von Datenobjekten.

Formales Vorgehen

Zentrale Begriffe der dynamischen Konstruktion sind „Ereignis" und „Zeit". Ereignisse resultieren entweder aus einer Folge von Aktionen im externen Schema, aus vorgelagerten Ereignissen im System oder sie werden durch Zeitablauf generiert.

Die Zeit spielt eine entscheidende Rolle für die Entwicklung eines Datenbestandes. Zum einen generiert sie autonom beeinflußbare und unbeeinflußbare Ereignisse (z. B. Systemabstürze), die den Datenbestand beeinflussen. Zum zweiten ist der Datenbestand in seiner chronologischen Entwicklung im Zeitablauf nachzuvollziehen.

Es ist daher wichtig, die wichtigen Zeittypen inhaltlich zu beschreiben und formal zu definieren (z. B. Zeitgenauigkeit, Abhängigkeiten der Zeiten untereinander) (vgl. Abschnitt "Zeitorientierte Datenbanken").

Weiterhin ist es sinnvoll, ein Konstruktionshilfsmittel zu wählen, das die grafische und verbale Beschreibung dynamischer Abläufe gestattet. Bewährt haben sich sogenannte PETRI-Netze.

Inhaltliches Vorgehen

Wie auch bei der statischen Konstruktion bietet es sich an, zunächst den betrieblichen Realitätsausschnitt mit Hilfe der traditionellen Instrumente der Systemanalyse zu durchleuchten.

Analyse der betrieblichen Vorgangsketten

Während die statische Konstruktion sich zunächst einmal der Kategorien der betrieblichen Aufbauorganisation bedient, setzt die dynamische Konstruktion an der Ablauforganisation an.

Betriebliche Abläufe geschehen durch Vorgänge und Vorgangsketten. Vorgänge sind die Elementarbausteine, die bestimmte betriebliche Teilaufgaben realisieren. Als Elementarbausteine werden sie nicht weiter zerlegt. Sie werden durch ein beobachtbares Ereignis gestartet und beendet (vgl. Ferstl/Sinz (1990), S. 578).

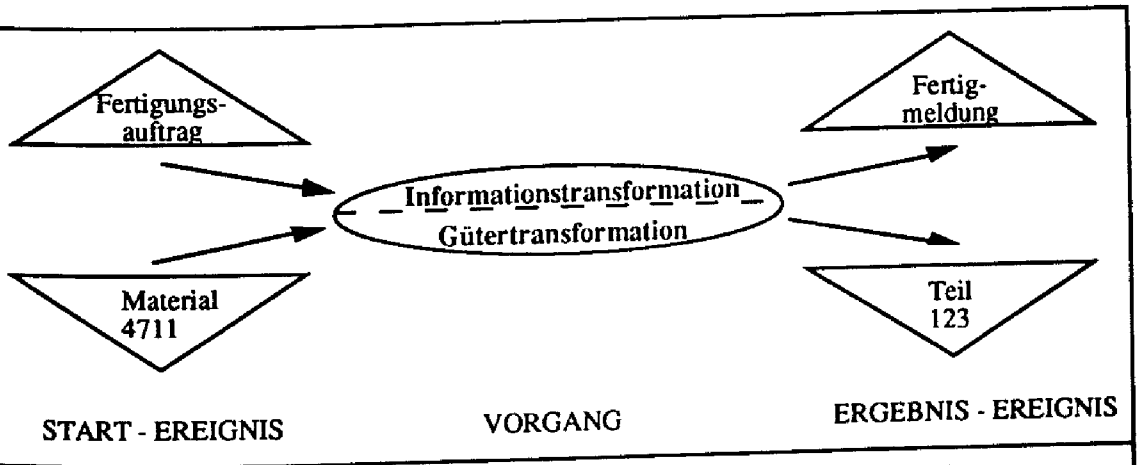


Abb. 3.2.2/3: Vorgänge des Güter- und Informationsstroms (vgl. Scheer (1991))

Vorgänge und Ereignisse lassen sich in solche der Informationstransformation und solche der Gütertransformation einteilen.

Unter Ablauforganisation versteht man die Gestaltung von Arbeitsprozessen zur Erziehung des Arbeitsergebnisses. Es handelt sich um einen in Raum und Zeit fortschreitenden Prozeß zur Erfüllung des betrieblichen Sachzieles, der aus Elementen der Gütertransformation (= Güterfluß) und der Informationstransformation (= Informationsfluß) besteht.

Bezogen auf den Gesamtbetrieb umfaßt die Ablauforganisation alle Arbeitsprozesse, die zur Erfüllung der Gesamtaufgabe der Unternehmung notwendig sind.

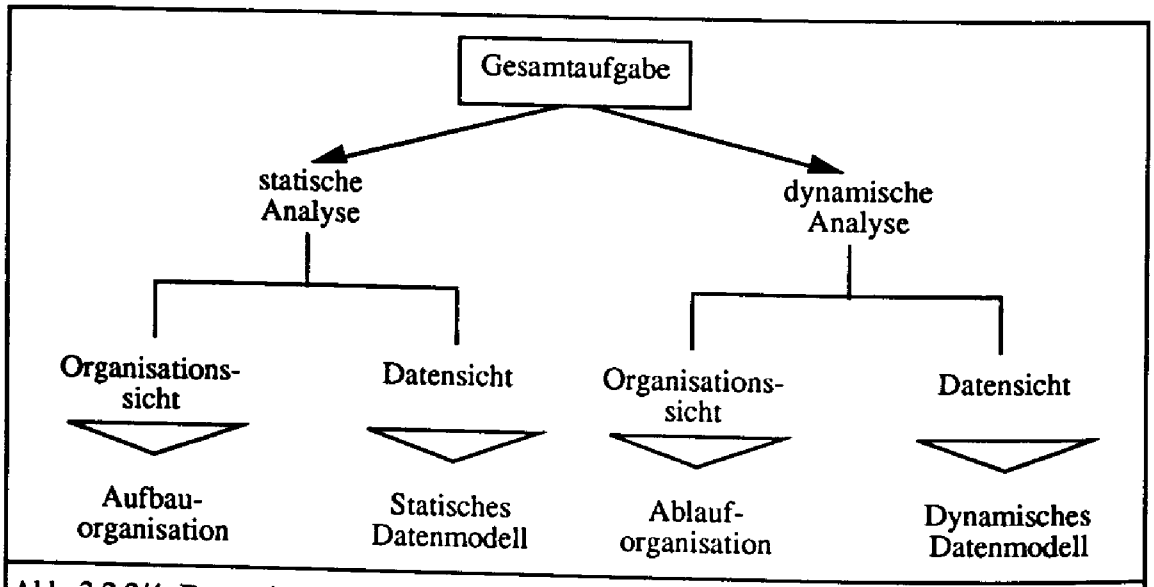


Abb. 3.2.2/4: Dynamische und statische Sicht der Organisationsanalyse

Diese Abläufe manifestierten sich in Vorgängen der Güter- und Informationstransformation. Die Vorgänge der Informationstransformation lassen sich in horizontal, vertikal oder zeitlich gerichtete Vorgangsketten gruppieren, die je nach Reichweite den Betrieb, das Unternehmen oder unternehmensübergreifende Einheiten umfassen.

Horizontale Informationsvorgangsketten verlaufen parallel oder entgegengesetzt zur Gütertransformation. Beispielsweise handelt es sich bei dem üblichen Ablauf von Produktionsplanungssystemen (Absatzprognose, Stücklistenauflösung, Nettobedarfsermittlung, Bestellplanung) um eine entgegengesetzte Informationsvorgangskette.

Vertikale Informationsvorgangsketten dienen zum einen der Unterrichtung übergeordneter Organisationseinheiten über realisierte Güter- und Informationstransformationsprozesse auf unteren Ebenen. Zum anderen dienen sie der Information über Entscheidungen oberer Ebenen und der Disaggretation der Entscheidungsinhalte

Beispiel:
Der Absatz wird auf Spartenebene vielleicht für Produktsegmente geplant, muß dann auf den unteren Managementebenen bis auf Artikelebene heruntergebrochen werden. Zeitliche Informationsvorgangsketten realisieren zum Beispiel Plan-Ist-Regelkreise auf den verschiedenen Transformationsebenen. Sie umfassen dazu jeweils korrespondierende Plan-Vorgänge und (realisierte) Ist-Vorgänge und deren Ausprägungen.

Informationsvorgangsketten dienen auch der unternehmensübergreifenden Integration. Dazu werden entweder EDI-Nachrichten ausgetauscht oder es wird auf einheitlich strukturierte Datenbestände zurückgegriffen.

3.2.2.2. Konstruktionsprobleme

3.2.2.2.1. Zeitenbildung

Für die dynamische Konstruktion eines Datenmodells müssen die in der Unternehmung verwendeten Zeitbegriffe geklärt werden. Dazu sind zu differenzieren:

(a) allgemeine Zeitbegriffe wie Datum - Zeit

(b) unternehmens- oder branchenspezifische Zeitbegriffe wie Fabrikkalender, Geschäftsjahre; Arbeitstage; Mann-Monate; Planjahre

Sinnvollerweise zu unterscheiden sind Zeitpunkte und Zeitperioden. Zeitpunkte lassen sich sinnvoll als Attribute von Datenobjekten oder Relationen darstellen; Zeitperioden benötigen normalerweise eigene Datenobjekte mit eigenen Attributen.

Zeitart	Planzeit	Planhorizont, Planjahr, Planperiode
	Istzeit	Ist-Horizont, Ist-Jahr, Ist-Periode
Zeitumfang	Zeitpunkt	30. März 1992, 11.10 Uhr
	Zeitraum	Monat, Woche, Tag
Zeittypen	Existenzzeit	Wann wird eine Veränderung in der Realität wirksam?
	Registrierungszeit	Wann wird eine Veränderung im Datenbanksystem registriert?
	Gültigkeitszeit	Wann wird eine Veränderung in der Realität bekannt?
Zeitgranularität	Meßeinheit	Sec., Min. Std.....
	Meßgenauigkeit	Sekunden, Microsekunden, Millisekunden, Nanosekunden
Zeitbestimmung	intervall-orientiert	durch systeminterne Uhr bestimmt
	ereignisorientiert	durch externe Ereigniszeit bestimmt

Abb. 3.2.2/5: Merkmale der Zeit

Die Zeit kann auf die Existenz oder die Ausprägung von Objekten, Beziehungen und Attributen wirken und das Auftreten oder die zeitliche Länge von Ereignissen oder Vorgängen beeinflussen.

Zwischen den Elementen eines Datenmodells bestehen zeitlogische und zeitumfangbestimmende Verhältnisse.

	Verhältnis	**Erläuterung**
Zeitlogische	Start - Start	Vorgang A beginnt gemeinsam mit Vorgang B.
	Start - Ende	Vorgang A beginnt gemeinsam mit dem Ende von Vorgang B.
	Ende - Start	Vorgang A muß beendet sein bevor Vorgang B beginnen kann.
	Ende- Ende	Vorgang A endet gemeinsam mit Vorgang B.
Zeitumfang-bestimmende	Maximalabstände	Zwischen Start von Vorgang A und Start von Vorgang B dürfen höchstens x, müssen jedoch mindestens y Zeiteinheiten liegen.
	Minimalabstände	

Abb. 3.2.2/6: Zeitverhältnisse zwischen Datenmodellkomponenten

Für die verschiedenen Zeitbegriffe in einem Unternehmen läßt sich ein Entity-Relationship-Modell aufstellen.

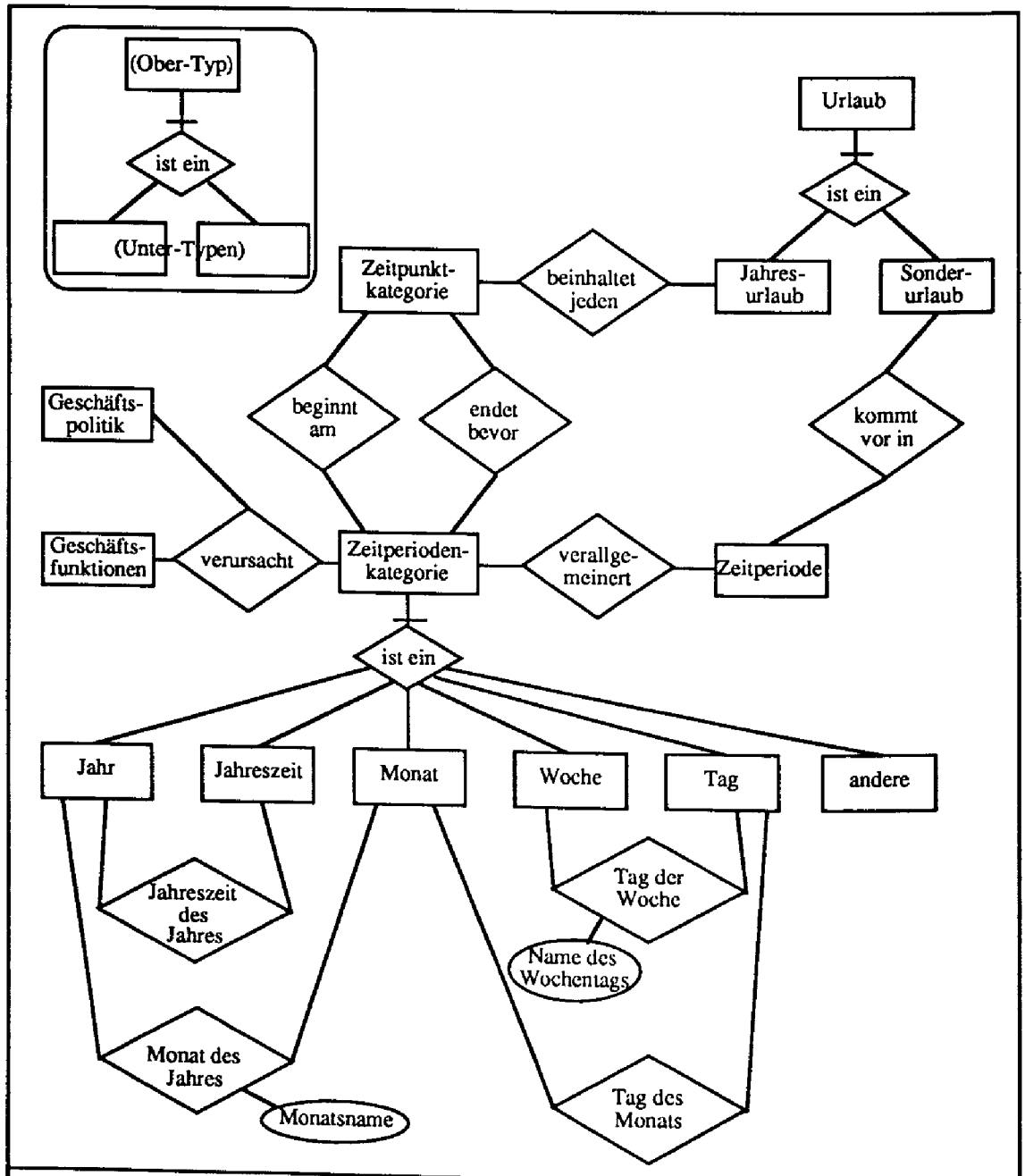


Abb. 3.2.2/7: Entity-Relationship-Konzepte der Zeit (vgl. Tasker (1988), S.246)

Wichtig ist, daß die Zeiten im Unternehmen einheitlich und eindeutig definiert werden, da nur dann die verschiedenen Anwendungssysteme zeitkompatible Daten verwenden.

3.2.2.2.2. Lebenszyklus

Ereignisse wirken auf die Existenz von Objekten und Beziehungen und auf die Wertausprägungen der zugeordneten Attribute.

Das Verhalten und der „Lebenszyklus" eines Datenobjektes wird durch die zugeordneten Ereignisse bestimmt. Es ist somit zu analysieren, wie sich die Zustände von Datenobjekten

und Beziehungen im Zeitablauf verändern. Man spricht von "Rollen", die diese dynamisch durchlaufen.

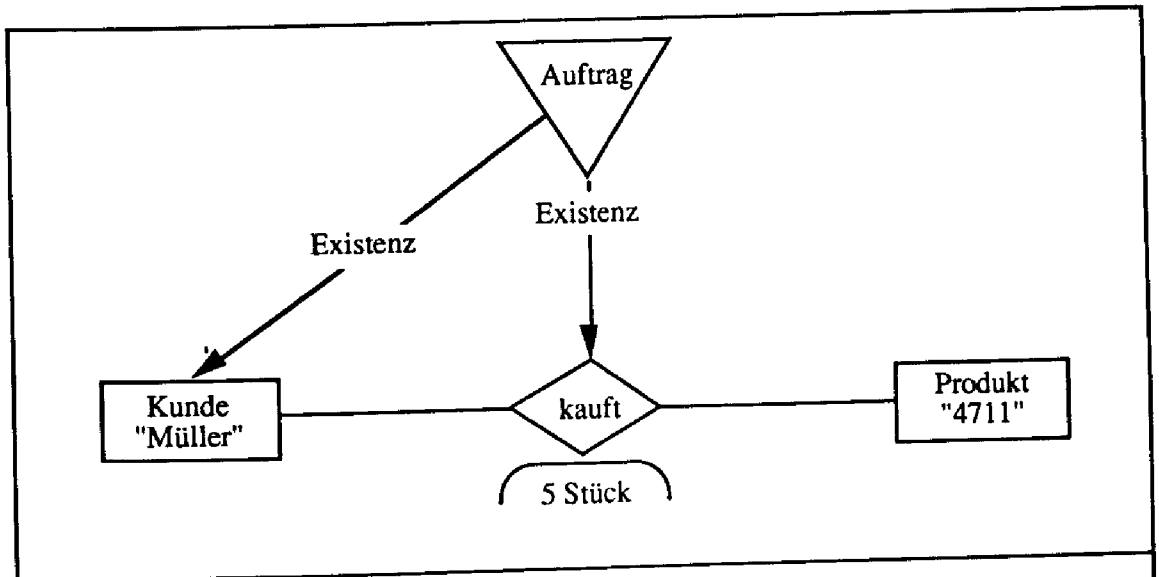


Abb. 3.2.2/8: Wirkung eines Ereignisses

Das Ereignis „Auftrag" führt unter Umständen zur Einrichtung eines neuen Objektes „Kunde" und ordnet der Beziehung „kauft" die Auftragsmenge als Attributsausprägung zu.

Aus betriebswirtschaftlicher Sicht ist es zum Beispiel von Interesse, wann ein „Individuum" ein Datenobjekt „KUNDE" wird. Denkbare Antworten sind:

- bei einer ersten Anfrage?
- beim Zusenden unseres Informations-, Werbematerials?
- bei einem definitiven Angebot?
- bei Vorliegen eines Kundenauftrages?

In ähnlicher Art und Weise ist zu fragen, welche Phasen ein Kunde durchläuft, wann ein „KUNDE" seinen Status wieder verliert. Denkbar ist eine Steuerung durch Zeitablauf oder durch Ereignisse.

Beispiel:
Wann wird ein Kunde ein Stammkunde? Wann ein Großkunde? Wann verliert ein Kunde seinen Status? (z. B. 2 Jahre kein Auftrag)

Ähnlich läßt sich der Lebenszyklus von Beziehungen konstruieren.

Beispiel:
Die abstrakte Beziehung zwischen Kunde und Produkt durchläuft im Geschäftsverkehr einen typischen Lebenszyklus, der unter anderem als Grundlage für die EDIFACT-Nachrichtenstruktur dient.

Kettenelemente	VERTRIEB	
1. Element	Anfrage (request for quote)	
2. Element	Angebot (quote)	
3. Element	Bestellung (purchase order)	
4. Element	Bestellbestätigung (purchase order response)	**RECHNUNGSWESEN**
5. Element	Rechnung (invoice)	Rechnung (invoice)

Abb. 3.2.2/9: Zyklus als Grundlage für die EDI-Nachrichtenstruktur

3.2.2.2.3. Ereignisbildung

Ereignisse starten oder beenden Vorgänge. Ein Ereignis wird durch bestimmte Bedingungen bewirkt.

Ereignisse sind entweder systeminterner oder systemexterner Art. Systeminterne Ereignisse werden durch andere Ereignisse, durch Datenobjekte oder Beziehungen innerhalb des Systems oder durch Zeitablauf ausgelöst. Systemexterne Ereignisse lassen sich nicht vollständig aus dem System heraus erklären, sie werden durch externe Eingaben allein oder in Kombination mit internen Systemelementen ausgelöst.

Die Auslösung eines Ereignisses kann von scharfen oder unscharfen Bedingungen abhängen. Bei scharfen Bedingungen tritt bei einer bestimmten Bedingungsausprägung das Ereignis unbedingt ein, bei unscharfen Konstellationen wird die Eintrittswahrscheinlichkeit durch eine mathematische Funktion abgestuft.

Ereignistyp	internes Ereignis		externes Ereignis	
Ereigniszeit	intern bestimmt	extern bestimmt	zufällig	
Ereignisdauer	Punktereignis		Dauerereignis	
Wirkung	Existenz von Datenobjekten/ Datenbeziehungen	Attributausprägungen von Datenobjekten/Datenbeziehungen	Existenz eines Nachereignisses	
Ereignis-bedingungen	Existenz eines(r) Datenobjektes/ Datenbeziehung	Existenz bestimmter Attributausprägungen	Existenz eines Vorereignisses	

Abb. 3.2.2/10: Merkmale zur Typisierung von Ereignissen und mögliche Ausprägungen in einem morphologischen Kasten

Ereignisse können sich beispielweise aus juristischen Tatbeständen (z. B. Angebot, Annahme, Vertragsabschluß), aus technischen Vorgängen (z. B. Ausfall einer Maschine) oder aus wirtschaftlichen Zustandsänderungen (z. B. Auftreten eines neuen Konkurrenten) ergeben. Nur häufige Ereignisse (sogenannte Routineereignisse) werden in aller Regel für DV-Systeme analysiert und erfaßt.

Ereignisse können als Vorbedingung, als Folge oder parallel zu einem anderen Ereignis, zu einem Datenobjekt oder einer Beziehung auftreten.

	Vorbedingung	**Parallelereignis**	**Folge (Nachbedingung)**
Datenobjekt (Kunde)	Auftrag wird erteilt	Kundendaten werden erfaßt	
Beziehung (kauft)	Auftrag wird erteilt		Auftrag wird erfüllt (Lieferung)
Ereignis (Auftragserteilung)	Angebot wird übermittelt	Angebotsdaten werden erfaßt	Auftrag wird bestätigt

Abb. 3.2.2/11: Typen von Ereignissen (am Beispiel: Kunde kauft Produkt)

Solche zeitlichen Strukturen aus Ereignissen, Objekten und Beziehungen werden oft Prozesse genannt.

Aus der zeitlichen Anordnung von Ereignissen zueinander lassen sich drei Prozeßklassen ableiten. Man unterscheidet:

- sequentielle Prozesse (mit zeitlich voll geordneten Ereignissen),
- parallele Prozesse (ohne zeitliche Ordnung der Ereignisse),
- überlappende Prozesse

(vgl. Klopprogge (1983), S. 27ff.).

3.2.2.2.4. Attributierung der Ereignisse

Die Zuordnung von Attributen auf Ereignissen hängt von deren Zusammenhang mit Objekten und Beziehungen ab.

Ereignisse enthalten als Attribut immer Referenzen auf die dadurch berührten Objekte und Beziehungstypen (niemals umgekehrt). Sie enthalten immer ein Datum als Attribut und teilweise einen identifizierenden Schlüssel, die zum Teil dann an das korrespondierende Objekt oder die Beziehung übergeben werden.

Beispiel:

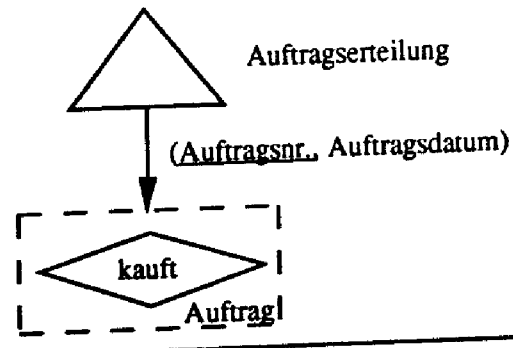


Abb. 3.2.2/12: Übergabe von Ereignisattributen auf eine Beziehung

Ein Ereignis muß die Attribute enthalten, die in der gesamten Vorgangskette benötigt werden.

Beispiel:

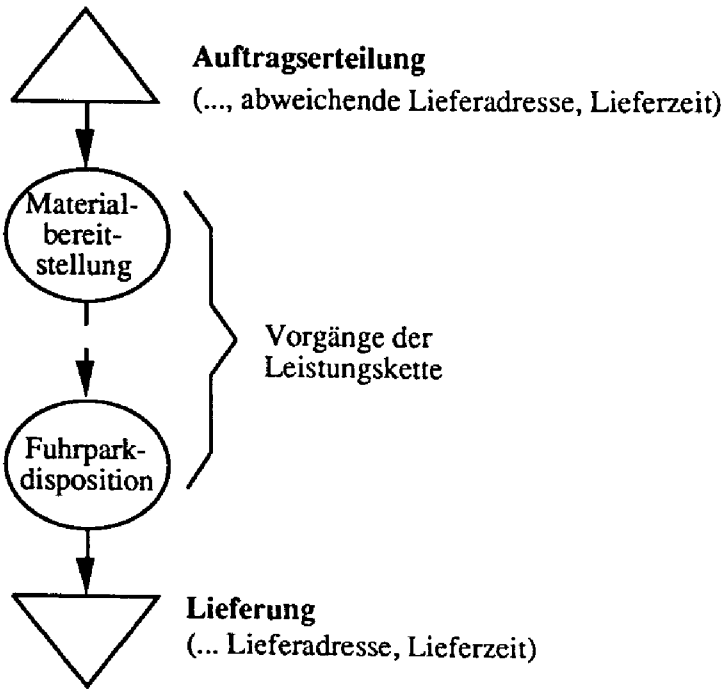


Abb. 3.2.2/13: Attributübergabe in der Vorgangskette

Da für die Lieferung sowohl die abweichende Lieferadresse als auch die Lieferzeit benötigt wird, muß das vorgelagerte Ereignis "Auftragserteilung" diese enthalten und weitergeben. Die Weitergabe kann innerhalb der gesamten Vorgangskette erfolgen; dadurch werden jedoch die Attributstrukturen aller Objekte, Beziehungen und Ereignisse unnötig vergrößert. Sinnvolle Alternative ist, eine Referenz (z. B. Auftragsnummer) weiterzugeben und bei Bedarf die erforderlichen Attribute bei der entsprechenden Beziehung oder dem Objekt abzufragen.

Eine besondere Schwierigkeit entsteht bei der Attributbildung extern ausgelöster oder extern strukturierter Ereignisse, da deren notwendige Merkmale durch die Vielfalt der kaufmännischen Gepflogenheiten geprägt werden.

Beispiel:
Ein Auftrag kann eine Vielzahl von Merkmalen enthalten, die aus Kundenwünschen resultieren und die bestimmend sind für die Auftragsabwicklung und Lieferung. Dazu gehören z. B.

- *-> Teillieferungen*
- *-> mehrere (abweichende) Lieferadressen*
- *-> abweichende Lieferungsempfänger*
- *-> Realrabatte in Form von Teil-Gratis-Lieferungen*
- *-> Fakturierung in unterschiedlichen Währungen.*

Eine vollständige Abbildung in den Anwendungssystemen aller denkbaren Attribute ist in der Regel in der Realität nicht möglich, doch kann dies vielfältige Auswirkungen auf die betrieblichen Abläufe haben.

Beispiel:
Ein Hersteller von Pflanzenschutzmitteln hatte als Attribute für einen Auftrag vorgesehen:
Gewichtseinheiten (t, ...)
Anzahl Verpackungseinheiten (Kanister etc.)
Ein Auftrag der Bundesbahn über "zu besprühende km" war nicht ohne erhebliche Umstellungen möglicher Attribute in den integrierten Systemen handhabbar.

3.2.2.2.5. Dynamische Integritätsbedingungen

Die logische Struktur und Eindeutigkeit eines Datenbestandes muß durch sogenannte Integritätsbedingungen gesichert werden. Diese beschreiben semantisch und pragmatisch zulässige Zeitfolgen der Attributsausprägungen von Objekten und Beziehungen.

Bedingungstyp	Beispiel
Inkompatibilitätsbedingung (incompatibility constraint)	Eine Person kann nicht gleichzeitig - Student - Ex-Student einer Universität sein.
Existenzbedingung (temporal existence constraint)	Eine Person kann nicht Student sein, bevor sie die Einschreibungsprozedur durchlaufen hat.
Interdependenzbedingung (existence dependency)	Die Beziehung "ist eingeschrieben" kann erst existieren, wenn die Objekte „Student" und „Kurs" existieren.
Übergangsbedingung (transition constraint)	Nur ein Student kann Examen machen. Mit dem Examen verliert er seinen Status und wird Ex-Student.

Abb. 3.2.2/14: Bedingungstypen zur Sicherung der dynamischen Integrität

3.2.2.3. Konstruktionsoperatoren

Es lassen sich folgende Konstruktionsoperatoren für die Prozeßkonstruktion unterscheiden (vgl. Brodie/Ridjanovic (1984)).

Dynamischer Konstruktionsoperator	Kennzeichen	korrespondierender statischer Operator
Sequenz/Parallelität (sequence)	zeitliche Anordnung	Aggregation (Konnexion)
Auswahl (choice)	logische Anordnung	Generalisierung/Spezialisierung
Wiederholung (repetition)	Schleife und deren Bedingungen	Association/Gruppierung

Abb. 3.2.2/15: Darstellung der Konstruktionsoperatoren für die Prozeßkonstruktion

Die dynamische Konstruktion von Datenmodellen befindet sich erst am Anfang ihrer Entwicklung. Es ist zu erwarten, daß sich in den nächsten Jahren mächtigere Konstruktionsoperatoren herausbilden.

3.2.2.4. Konstruktionshilfsmittel

Hilfsmittel für die dynamische Struktur unterscheiden in aller Regel zwischen Datenobjekten (entities), Relationen (relationships) sowie Datenereignissen (events). Im Mittelpunkt steht die spezielle Natur der Ereignisse.

Ein verbreitetes Konzept in den Hilfsmitteln sind sogenannte PETRI-Netze.

	Denkbare Interpretationen	**Symbol**
Transitionen	Steueroperationen Bearbeitungsabschnitte Aktionen, Operationen Regeln	
Plätze/ Stellen	Prozeßbedingungen Systemzustand	
Marken	Arbeitszeitpunkt Prozeßzustand	
Kanten	Logische Beziehungen	

Abb. 3.2.2/16: Kategorien von Petri-Netzen

In PETRI-Netzen werden dynamische Elemente (sogenannte Transitionen) von statischen Elementen (sogenannte Stellen) unterschieden. Marken lösen die Aktionen der dynamischen Elemente aus und befördern die statischen Elemente in bestimmte Zustände.

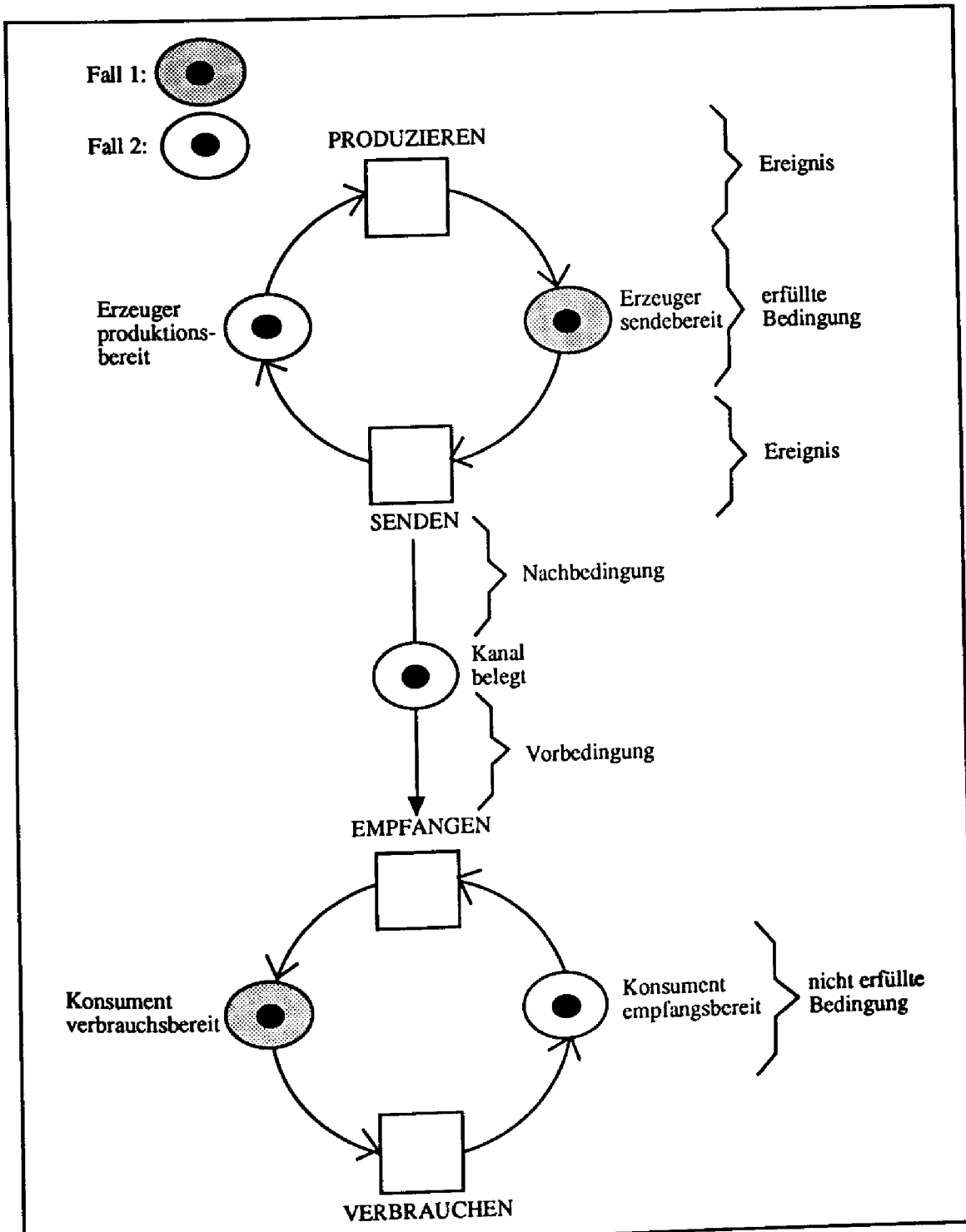


Abb. 3.2.2/17: Beispiel für ein einfaches PETRI-Netz

3.2.2.4.1. Beispiel 1: Verbindung von Petri-Netzen und ERM

Eine Reihe von Ansätzen ergänzen das statische ERM um Petri-Netzstrukturen (vgl. Kappel/Schreffl (1989), Sakai (1983)). Vorgestellt sei hier der Ansatz von Sakai.

Dieser ergänzt das statische ERM mit den Elementen E = Entity und R = Relation um ein drittes Element Z = Zustands- oder Zeit-Faktoren und daraus resultierende „Augenblick-Aufnahmen" (instances), die die Ausprägung jedes Datenobjekts und/oder jeder Beziehung zu jeder Ausprägung des Zustands- oder Zeitfaktors F beschreibt.

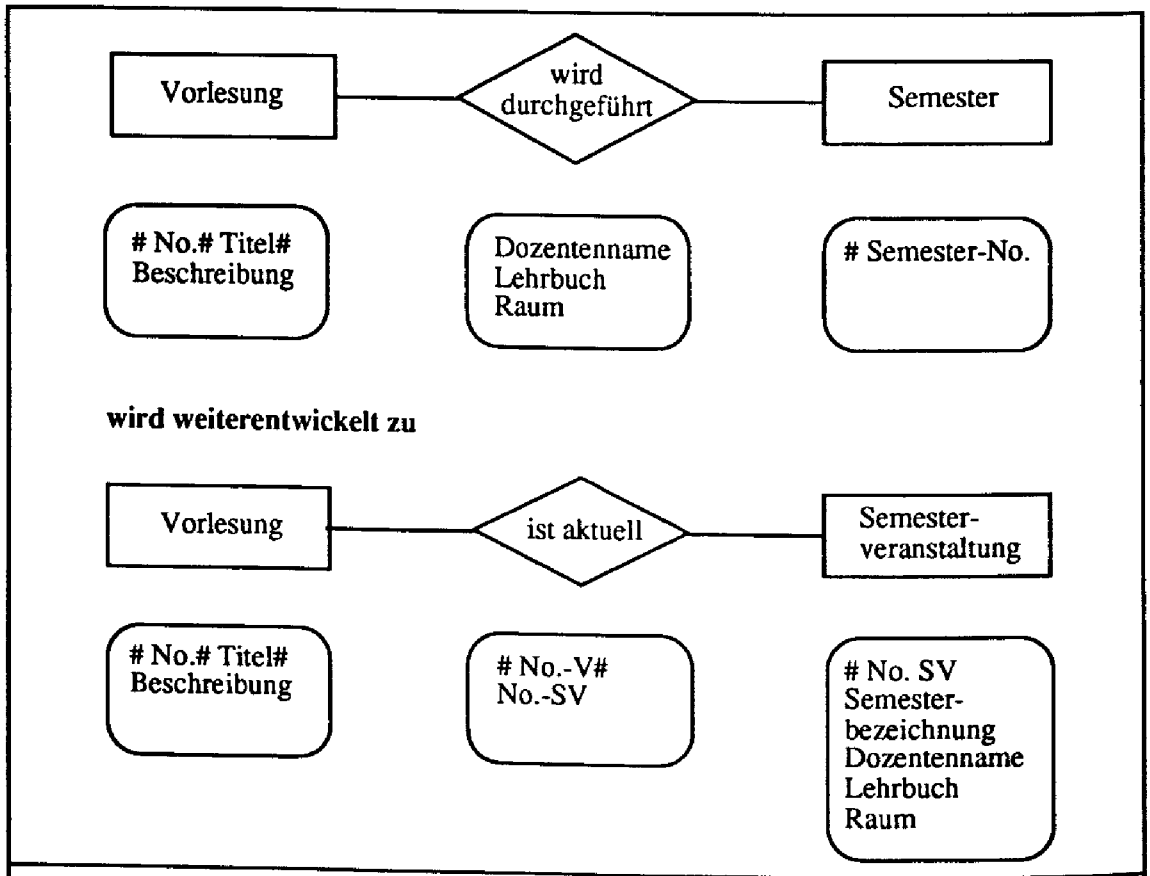


Abb. 3.2.2/18: Erweiterung des Entity-Relationship-Modells um Zustands- oder Zeit-Faktoren (nach Sakai (1983))

Das Datenobjekt „Semesterveranstaltung" ist damit eine zeitliche Ausprägung des Datenobjekts „Vorlesung".

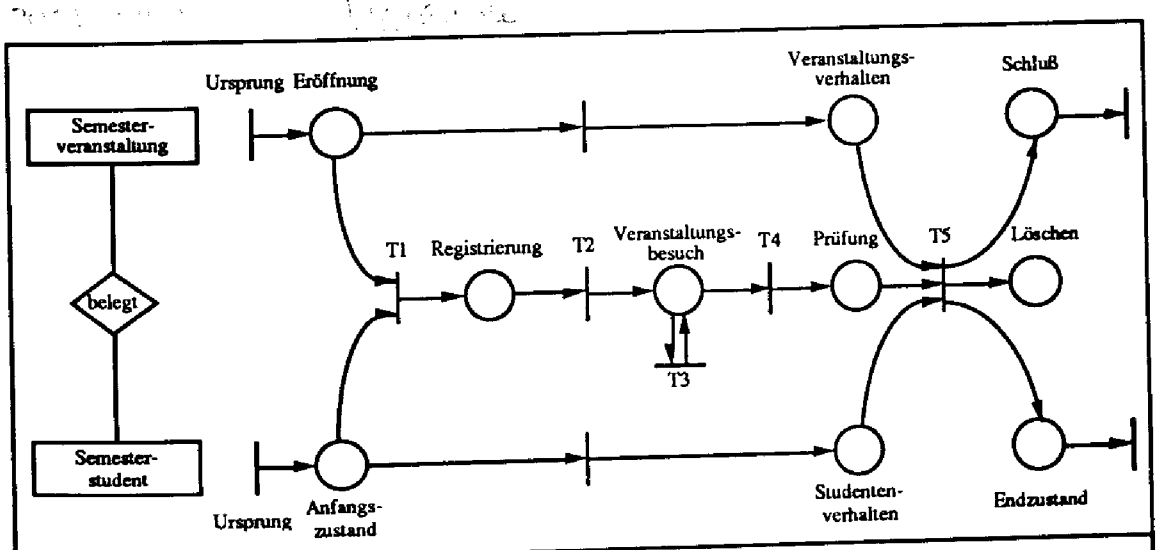


Abb. 3.2.2/19: Das "ER-Verhaltens-Diagramm" einer Universitätsdatenbank (vgl: Sakai (1983), S.116)

Auf der linken Seite dieser Abbildung findet man das ERM mit den zwei zeitabhängigen Datenobjekten Semesterveranstaltung und Semesterstudent und der Relation „belegt". In der Petri-Netz-Darstellung hat die Relation „belegt" jetzt eine Historie, die von der Registrierung im Zeitpunkt T1 über den Veranstaltungsbesuch zwischen den Zeitpunkten T2 bis T4 und die anschließende Prüfung in T5 zum Löschen der Beziehung im Anschluß an die Prüfungsnote führt.

Sakai faßt bestimmte Transaktionen zu Typen mit gleichem zeitlichen Verhalten zusammen und kommt zu einer weiteren Abstraktionssicht.

Auf der 4. Abstraktionsebene werden die Typen unterschieden, die sich im Zeitablauf gleichgerichtet verhalten. So wird beispielsweise für alle Studenten zum gleichen Zeitpunkt festgestellt, ob sie eine bestimmte Veranstaltung besucht haben oder nicht.

Um die Attribute zu identifizieren, die sich durch Prozesse dynamisch im Zeitablauf verändern können, wird ein zusätzliches Datenobjekt eingefügt, das diese Attribute zusammenstellt.

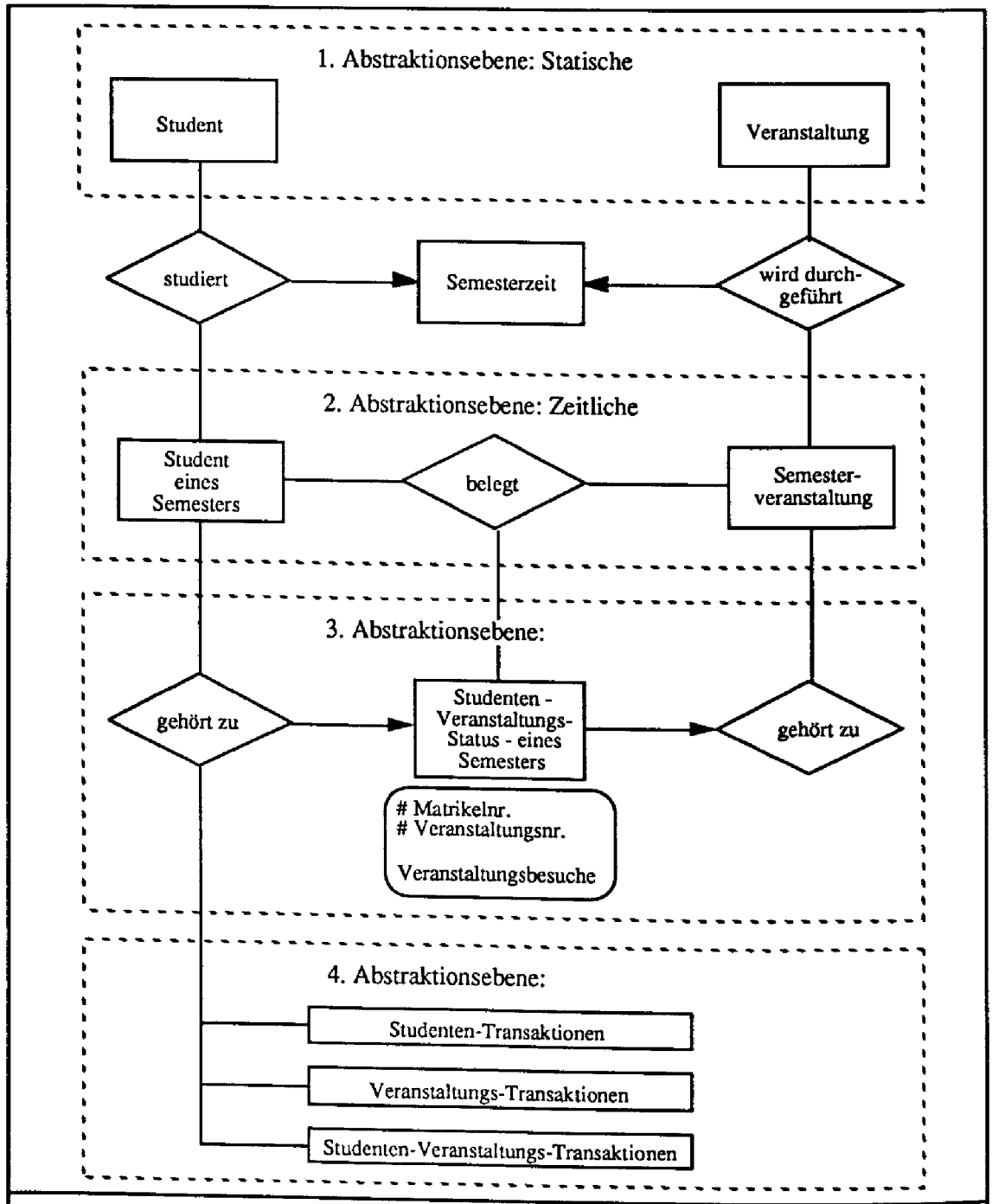


Abb. 3.2.2/20: Abstraktionsebenen (vgl. Sakai (1983)

Sakai ergänzt somit das ERM um Verhaltensdiagramme (E-R behavior diagram) auf der Basis von Petri-Netzen. Diese leisten eine strukturierte verbale Beschreibung (behavior description), die für jedes Datenobjekt und jede Relation existiert und die Transaktionen definiert, die

-> zur Existenz von Datenobjekten oder Relationen führen
-> zur Existenz von Attributen bei Objekten oder Relationen führen
-> bestimmten Attributen Werte zuordnen oder diese verändern.

Diese Transaktionsbeschreibungen werden als Pseudo-Code wie folgt niedergelegt:

```
S1: GET          SEMESTERVERANSTALTUNG
S2: GET          SEMESTERSTUDENT
S3: CREATE       (SEMESTERVERANSTALTUNG, 'belegt durch',
                 SEMESTERSTUDENT)          (S1, S2)
```

Zweites Element der verbalen Beschreibung als Pseudo-Code-Anweisungen sind Zustandsbeschreibungen

`OPENED`: EXS (`SEMESTERVERANSTALTUNG`)
(= Existenz)

`ATTENDING`: NV (`HÖRER`)
(= Number of Values-Wertezuordnung)

`SCORED`: HV (STUDENT, PRÜFUNG)
(= have values)

Konstruktionshilfsmittel auf der Basis von PETRI-Netzen haben einige Nachteile:

- Einfache Konstruktionselemente (Verzweigungen) sind mit PETRI-Netzen häufig schwer zu realisieren. Zudem ist das Beschreibungsniveau zu niedrig; ein modularer Entwurf wird nicht unterstützt, so daß die Netze schnell sehr komplex werden. Oft müssen zur Abbildung der Realität künstliche Elemente eingefügt werden (z. B. künstliche Synchronisationspunkte).

3.2.2.4.2. Beispiel 2: Verbindung von Jackson System Development (JSD) und ERM

JSD ist eine Methode zur strukturierten Entwicklung von Informationssystemen (vgl. Jackson (1983)). Die Methode erlaubt die Erfassung dynamischer Abläufe durch Abbildung von Prozessen und Zuständen. Eine JSD Struktur bildet nicht eine Hierarchie von Prozeduren, sondern besteht aus einem Netz interdependenter Prozesse.

Den mächtigen Eigenschaften bei der Abbildung von Prozeßstrukturen stehen geringe Fähigkeiten zur Datenmodellierung gegenüber, so daß es sich anbietet, JSD und ERM miteinander zu verbinden (vgl. Robinson (1979), Mees/Put (1987)). Dabei werden Aktionen mit Hilfe sogenannter Sequenzdiagramme abgebildet, die den gesamten Lebenszyklus eines Datenobjektes erfassen. Diese bilden dann die Basis für das Entity-Relationship-Modell.

JSD geht in sechs Hauptschritten vor, die an dem Beispiel des Einschreibeprozesses an einer Fern-Universität verdeutlicht werden sollen (vgl. Mees/Put (1987)):

1. Schritt: Entity Action und Entity Structure Step

Ein Entity im JSD ist jedes Objekt der Realität, das Auslöser oder Adressat von Aktionen ist. Die betrachteten Aktionen werden definiert und mit ihren Parametern beschrieben

Beispiel:
Für einen Studenten existieren folgende relevante Aktionen:

Aktion	*Beispiel*	*Parameter*	*Initiator*
EINSCHREIBEN	*Ein Student schreibt sich definitiv durch Abgabe des entsprechenden Formblattes für eine Veranstaltung ein.*	*Studentenname, Adresse, Geburtsdatum, Telefonnr., Kursname, Einschreibedatum*	*Student*
ANFRAGE	*Ein Interessent bittet um die Zusendung einer Probe der Fernstudienbriefe*	*Interessentenname, Adresse, Versanddatum*	*Interessent*
UNTERLAGEN-VERSAND	*Der Student erhält die relevanten Fernstudienbriefe für die von ihm belegte Veranstaltung*	*Studentenname, Kursname, Studienbrief-Nr.*	*Student*
BEZAHLUNG	*Der Student bezahlt das Hörerentgelt für die von ihm beanspruchte Veranstaltung*	*Studentenname, Kursname, Betrag, Zahlungsdatum*	*Student*
MAHNUNG		*Studentenname, Kursname, Betrag, Mahndatum*	*Student*
EXMATRIKULATION	*Nach erfolgloser zweiter Mahnung wird ein Student aus dem spezifischen Kurs exmatrikuliert.*	*Studentenname, Kursname*	*Student*
HAUSARBEIT	*Der Student sendet die von ihm zu bearbeitenden Teile der Fernstudienbriefe zurück*	*Studentenname, Kursname, Studienbrief-Nr., Note*	*Student*
KLAUSUR	*Am Ende eines Kurses wird normalerweise eine Klausur geschrieben.*	*Studentenname, Kursname, Note*	*Student*
INTERESSENTEN-ERINNERUNG	*Interessenten, die eine Probe der Studienbriefe angefordert und erhalten haben, werden viermal an eine Einschreibung erinnert.*	*Interessentenname*	*Interessent*
INTERESSENTEN-LÖSCHUNG	*Interessenten, die sich nach vierfacher Erinnerung nicht einschreiben, werden gelöscht.*	*Interessentenname*	*Interessent*

Abb. 3.2.2/21: Entity Action und Entity Structure Step am Beispiel eines Studenten

Die erfaßten Aktionen gehören somit zu zwei Objekten: Student und Interessent. Zur Spezifizierung der Aktionen werden nun sogenannte Sequenzdiagramme verwendet. Diese

bestehen aus einem Baum, der die drei klassischen Programmkontrollelemente Sequenz, Iteration (durch Stern gekennzeichnet) und Auswahl (durch Kreis gekennzeichnet) umfaßt.

Für das Objekt Interessent ergibt sich das folgende Sequenzdiagramm.

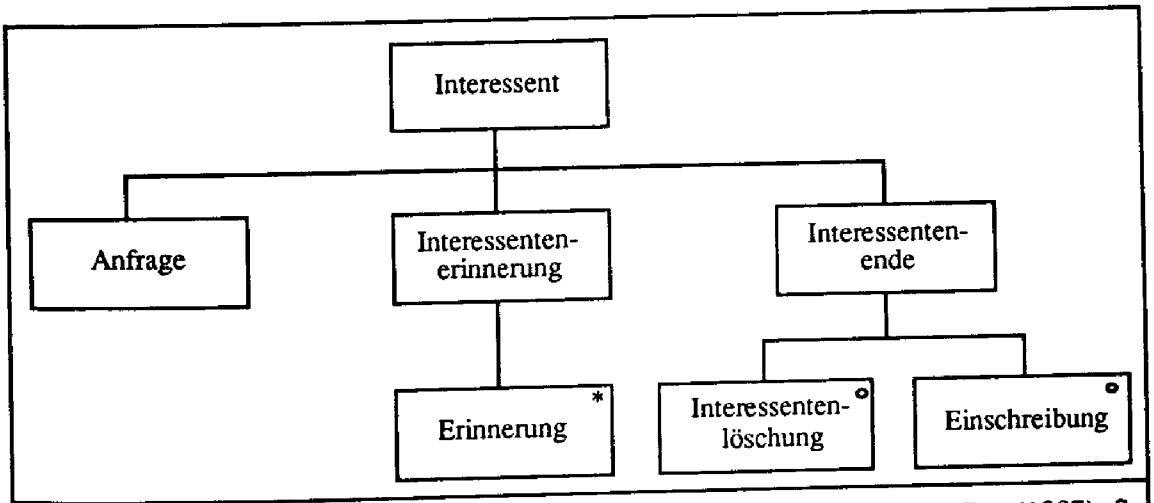


Abb. 3.2.2/22: Jackson Sequenz Diagramm eines Interessenten (vgl. Mees/Put (1987), S. 403)

Der Interessent fordert Probeunterlagen an, wird bis zu vier Mal zur Einschreibung aufgefordert (Iteration), schreibt sich als Student ein oder wird gelöscht (Auswahl).

Das Objekt Student ist komplexer: Die erforderlichen Aktionen sind unterschiedlich, ob der Student vorher Interessent oder Student anderer Kurse war oder nicht, da im ersten Fall die Stammdaten registriert werden müssen. Im Sequenzdiagramm werden die Bedingungen im oberen Teil des Rechtecks und die Aktionen im unteren Teil des Rechtecks angegeben.

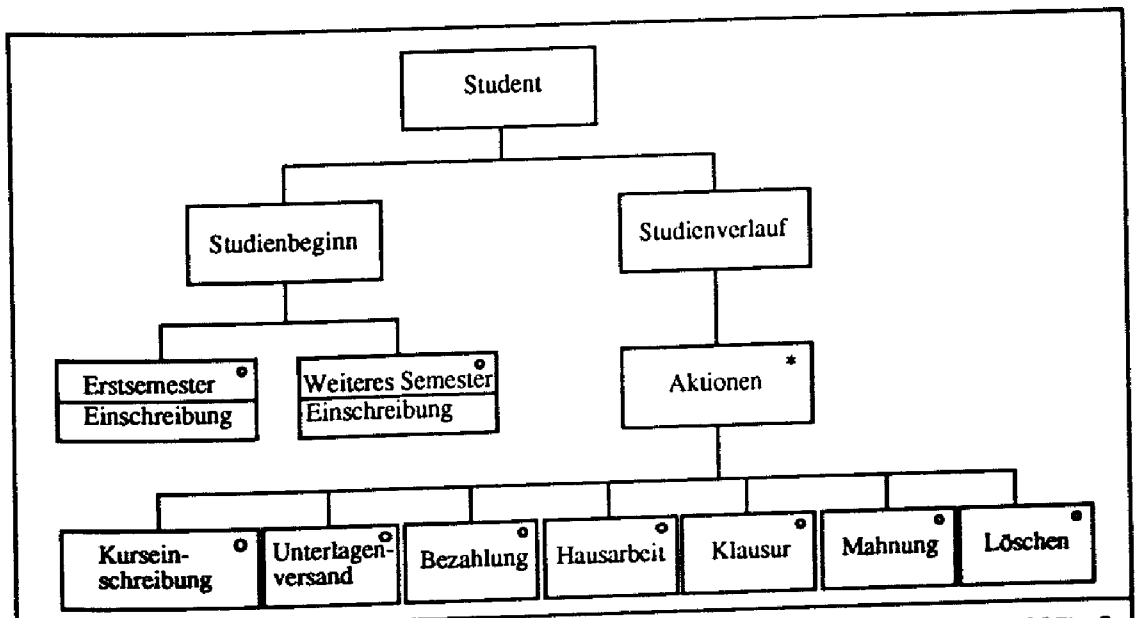


Abb. 3.2.2/23: Jackson Sequenz Diagramm eines Studenten (vgl. Mees/Put (1987), S. 403)

Die Objekte Interessenten und Studenten können zu einem übergeordneten Objekt Person abstrahiert werden.

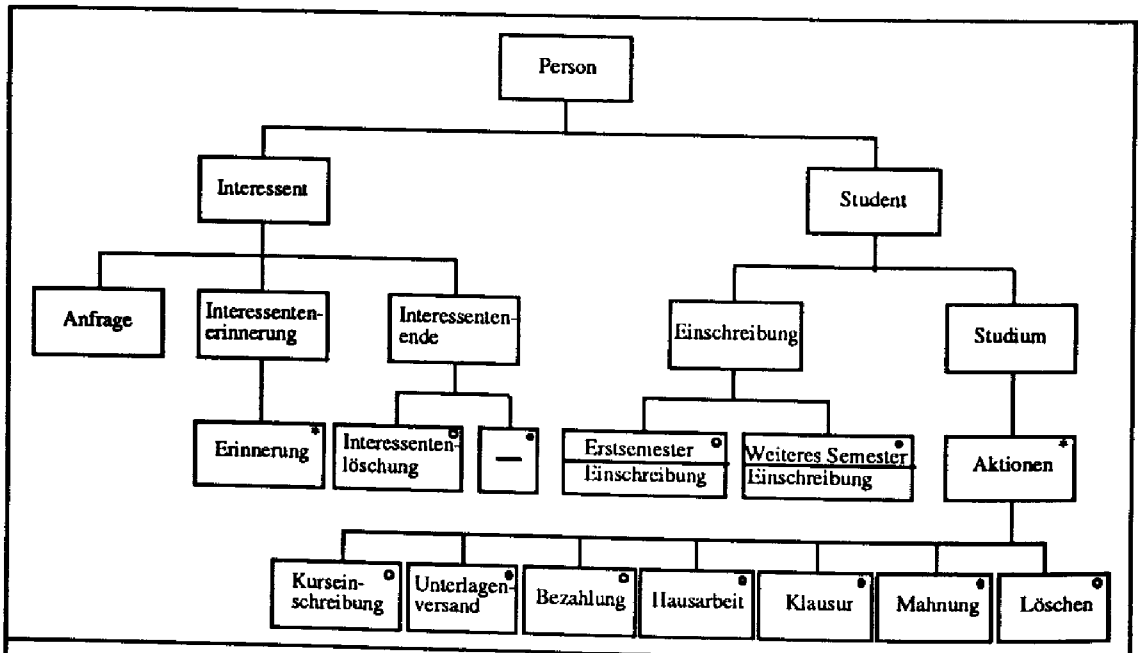


Abb. 3.2.2/24: Jackson Sequenz Diagramm für Obertyp "Person" (nach Mees/Put (1987), S. 406)

2. Schritt: Initital Model Step

Die Aktionen werden als eine Folge miteinander kommunizierender Prozesse definiert und beschrieben. Bei der Initialisierung werden extern und intern ausgelöste Aktionen unterschieden.

Beispiel:

Prozeß	*Intern ausgelöste Aktionen*	*Extern ausgelöste Aktionen*
Versenden	*UNTERLAGENVERSAND*	
Mahnen	*MAHNUNG, EXMATRIKULATION*	
Erinnern	*INTERESSENTEN-ERINNERUNG, INTERESSENTEN-LÖSCHUNG*	
PERSON-0		*EINSCHREIBUNG, ANFRAGE, BEZAHLUNG, HAUSARBEIT,*

Abb. 3.2.2/25: Beispiel zum Initial Model Step

Weiterhin werden Prozesse der Realität (hier eine reale Person, die sich einschreibt oder anfragt) und die im System modellierten Prozessen verbunden.

Die Verbindungen zwischen realen und modellierten, externen und internen Prozessen werden beschrieben mit Hilfe des System Specification Diagram (SSD). Sequentielle Prozesse werden entsprechend zum Sequenzdiagramm in Rechtecken ausgedrückt; Datenströme durch Kreise. Eine doppelt gestrichelte Linie kennzeichnet das mehrmalige Auftreten von Prozessen.

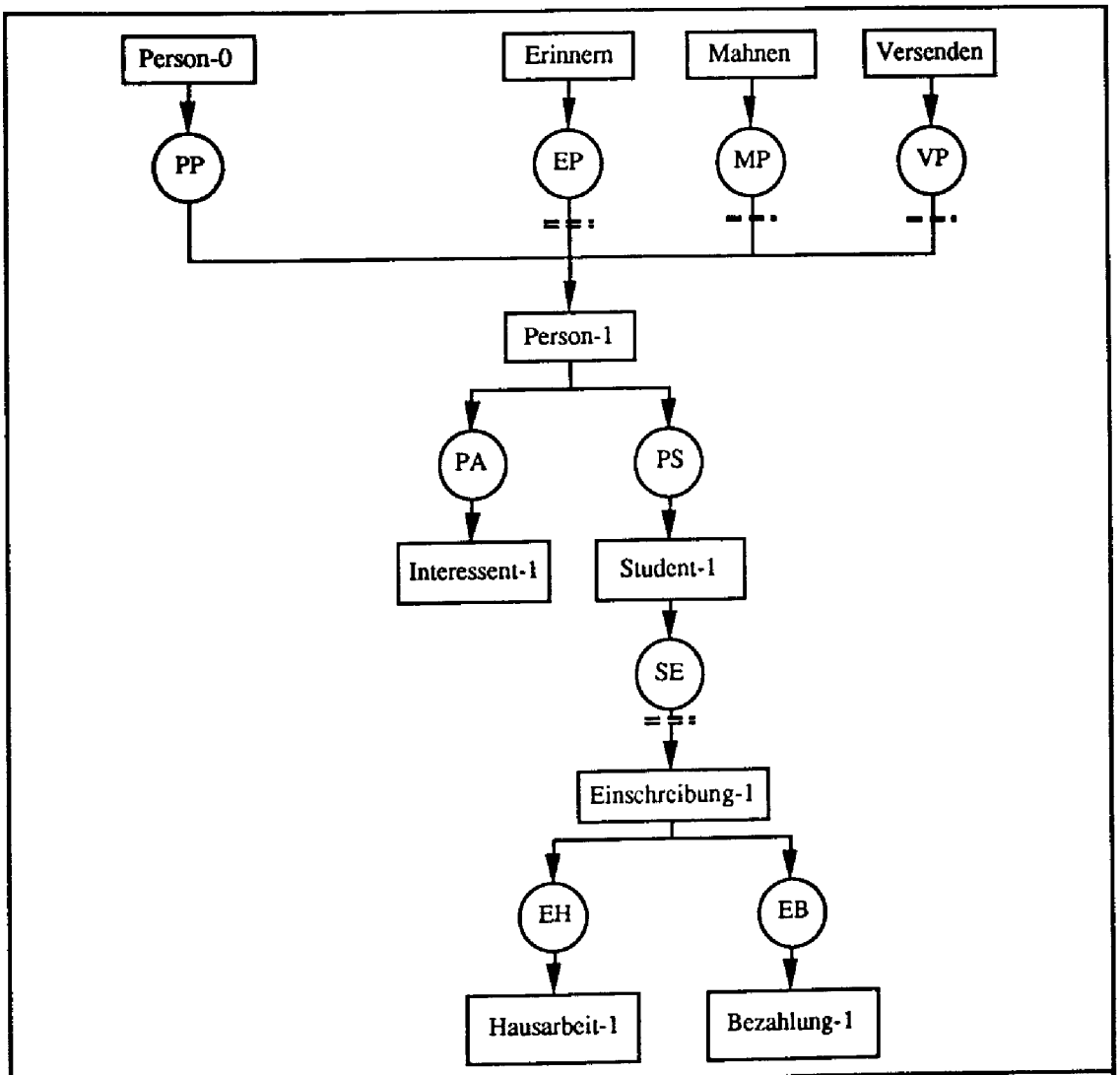


Abb. 3.2.2/26: System Specification Diagramm, (vgl. Mees/Put (1987), S. 407)

Diese Prozeßstruktur wird jetzt mit Hilfe eines ERM um die statische Datenstruktur ergänzt.

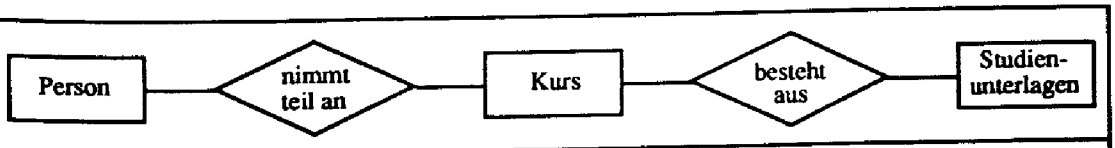


Abb. 3.2.2/27: Entity-Relationship Diagramm, (nach Mees/Put (1987), S. 408)

Das JSD liefert somit zu den Entities und Relationen den jeweiligen Lebenszyklus. Die Beziehung zwischen Person und Kurs durchläuft beispielsweise folgenden Lebenszyklus:

- fragt an,
- wird erinnert,
- schreibt sich ein,
- bezahlt nicht und wird gemahnt,
- bezahlt,
- gibt bearbeitete Studienunterlagen ab,

- schreibt Klausur.

Dieser Lebenszyklus "Teilnahme" (= nimmt teil an) wird in einem separaten Entity abgebildet, das aus den beiden Ursprungsentities "Person" und "Kurs" gebildet wurde und dessen Aktionen in einem Sequenzdiagramm definiert werden.

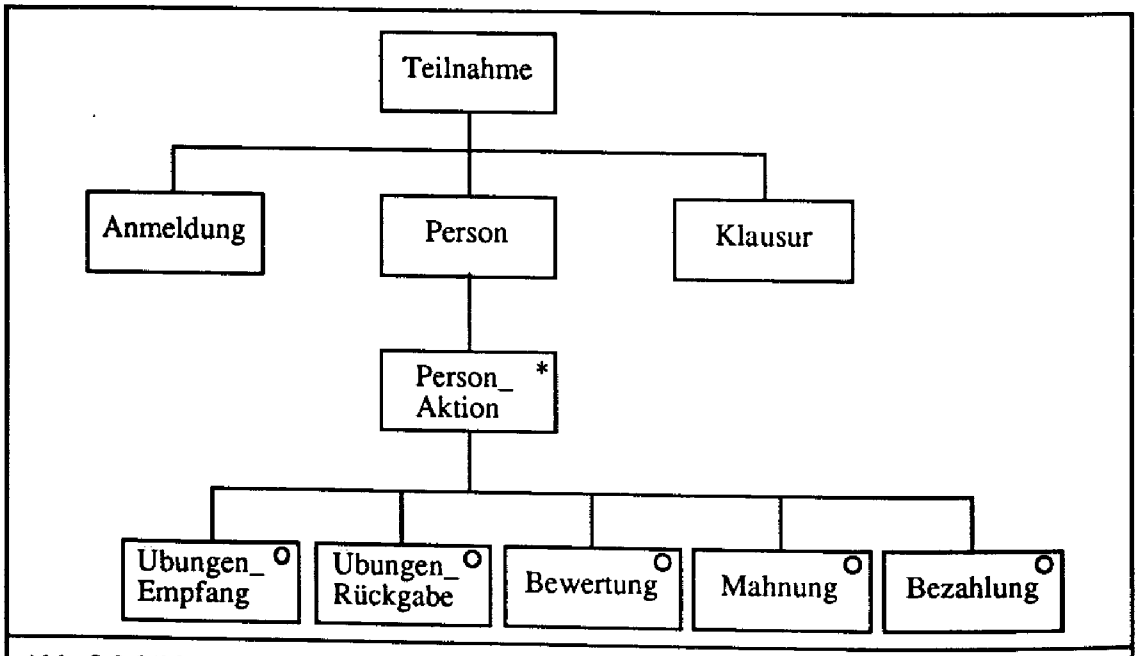


Abb. 3.2.2/28: Sequenzdiagramm (vgl. Put (1989), S.427)

Das Objekt Person kann erst dann existieren, wenn eine der initialisierenden Aktionen des JSD realisiert wurde.

Es folgen vier weitere Schritte der JSD-Methode, die für das grundsätzliche Vorgehen jedoch nicht entscheidend sind und daher hier nur erwähnt werden:

3. Schritt: Interactive Function Step
4. Schritt: Information Function Step
Dieser Schritt dient dazu, den notwendigen Qutput des System zu beschreiben und durch die Einfügung spezieller Prozesse zu definieren.
5. Schritt: System Timing Step

6. Schritt: Implementation Step

Hier wird die zeitliche Struktur (insbesondere die zeitliche Taktung und die Geschwindigkeit) des Systems und dabei speziell des zu generierenden Outputs festgelegt, da daraus Anforderung an die Zeitstrukturen der Input-Prozeduren ableitbar sind.

3.2.2.4.3. Beispiel 3: BIER - Behaviour Integrated Entity Relationship Approach

Das Konzept BIER baut auf dem ERM auf und ergänzt dieses um dynamische Verhaltensaspekte. Dazu wird ein zusätzlicher Entity-Typ eingeführt, der den Status der Grund-Entities angibt (vgl. Edet/Kappel/Tjoa/Wagner (1986), Kappel/Schreffl (1989)).

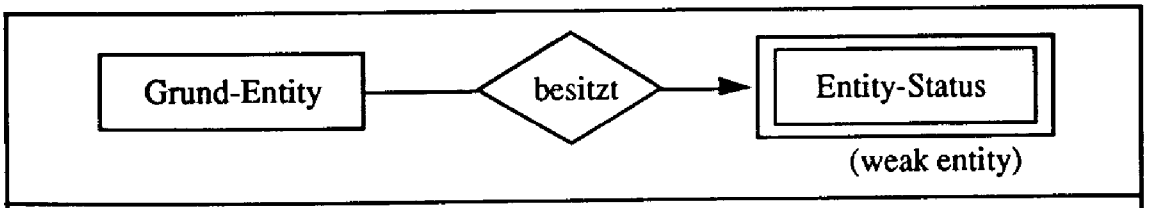


Abb. 3.2.2/29: Status der Grund-Entities

Der „Entity-Status" ist dabei ein schwacher Entity-Typ, d. h. seine Existenz hängt ab von einem entsprechenden Grund-Entity. Im Entity-Status werden jetzt folgende Typen von Attributen unterschieden:

- Schlüsselattribute, die den Bezug zum Grund-Entity herstellen,
- Statusattribute, die den Zustand eines Grund-Entities zu einem bestimmten Zeitpunkt widerspiegeln,
- Status-Zeitattribute, die den Zeitpunkt angeben, zu dem ein Entity seinen Status verändert hat.

Die Status-Attribute haben vordefinierte Ausprägungen: Beispielsweise kann der Familienstand eines Mitarbeiters erschöpfend wie folgt beschrieben werden:

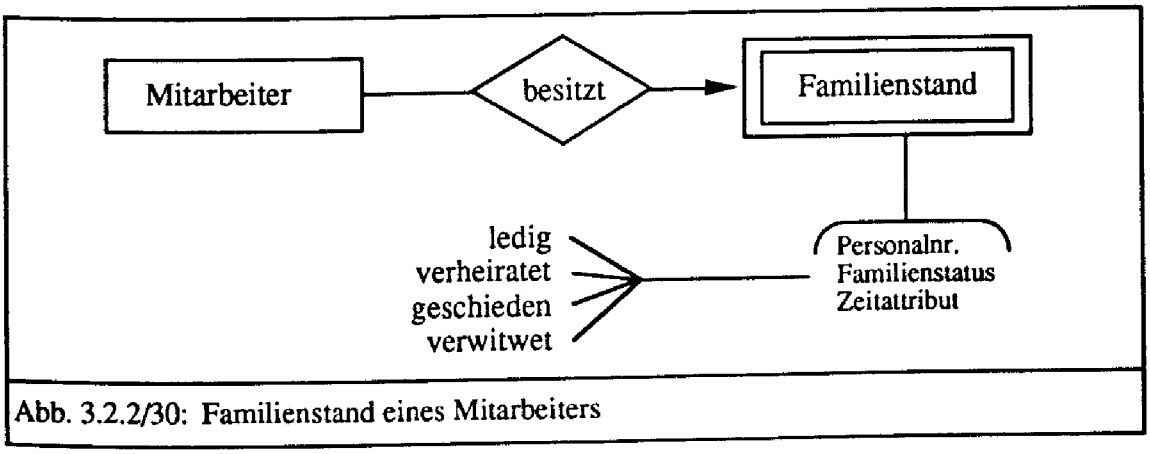


Abb. 3.2.2/30: Familienstand eines Mitarbeiters

Problematisch wird diese Modellierung, wenn das Status-Sub-Entity parallel mehrere Zustandsattribute umfaßt:

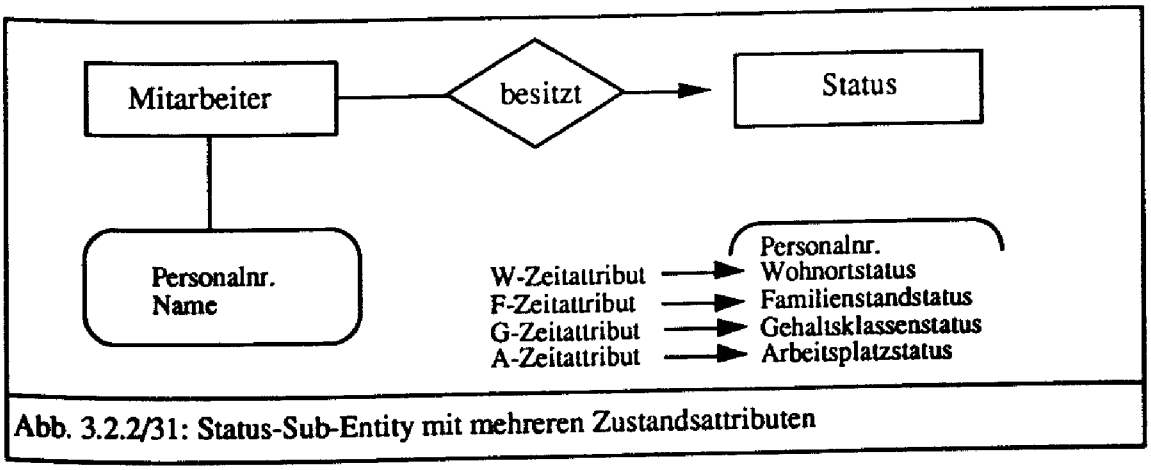


Abb. 3.2.2/31: Status-Sub-Entity mit mehreren Zustandsattributen

Bei bestimmten Status-Attributen läßt sich eine natürliche Statusfolge angeben (so z. B. beim Familienstand), die sich durch ein Petri-Netz-Konzept modellieren läßt:

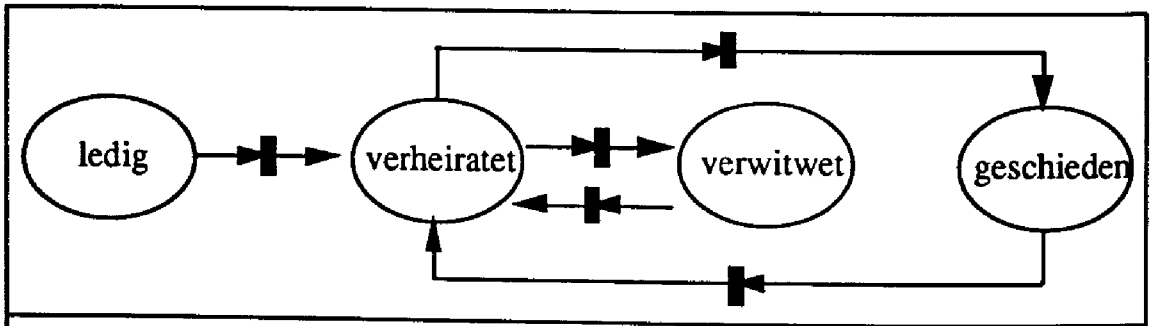


Abb. 3.2.2/32: Logische (natürliche) Reihenfolge von Zuständen nach dem BIER-Konzept (vgl. Eder/Kappel/Tjoa/Wagner (1987), S. 155 f.)

Bei anderen Status-Attributen läßt sich eine solche logische Folge nicht angeben, so daß offen bleibt, wie die Modellierung zu geschehen hat.

3.3.2.2.4. Beispiel 4: Extended Entity-Attribute-Relationship-Ansatz (EAR)

Iivari/Koskela schlagen ein Konstruktionshilfsmittel vor, in dem eindeutig zwischen der Abbildung der Realität in einem Objektsystem und dessen Abbildung in einem Informationssystem unterschieden wird (vgl. Iivari/Koskela (1983)).

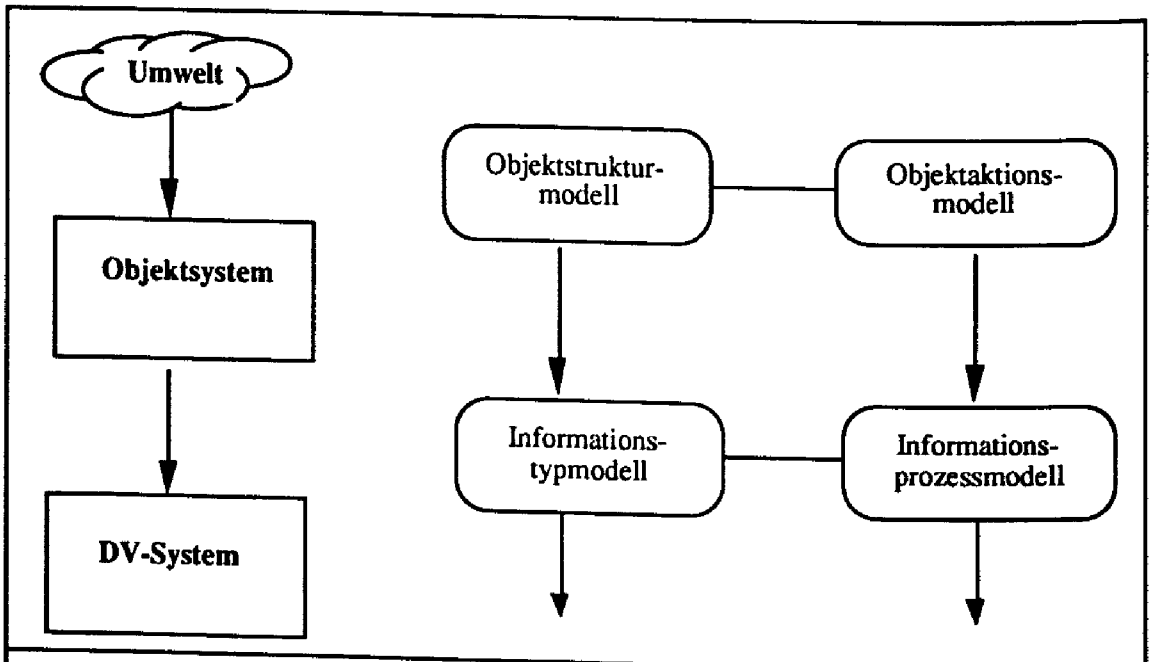


Abb. 3.2.2/33: Ebenen und Teilmodelle des Konstruktionshilfsmittels von Iivari/Koskela

Für die Abbildung in einem Objektsystem werden zwei Teilmodelle verwendet: Die Objekt-System-Struktur beschreibt das Zusammenspiel von Ereignissen und (statischen) Datenobjekten. Das Objektaktionsmodell (Object System Event Action) beschreibt das Verhalten des Objektsystems mit Hilfe von Aktionen und Triggern.

Objekt-System-Struktur

Für die Modellierung werden folgende Kategorien verwendet: Objekttypen, Assoziationstypen und die Zeit.

Objekttypen werden zerlegt in „Entity-Types", „Event-Types" und Zeitintervall-Typen.

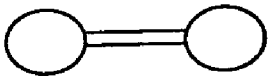
	Beschreibung	Graphisches Symbol	Beispiele
Objekt Typen (Entity - Types)	Objekte mit längerer Lebensdauer		Artikel, Kunde
Ereignis Typen (Event - Types)	Objekte mit kurzer, von vornherein begrenzter Lebensdauer		Auftrag, Lieferung
Zeit Intervall Typen (Time - Intervall-Types)	Zeitobjekte		Monat
Attribut Typen (Property association types)	Eigenschaften der Objekte, Ereignisse		Auftragsmenge, Lieferungszeit
Relationship Assoziation Types	Zusammenhänge zwischen Objekten		Buchung

Abb. 3.2.2/34: Objekt-System-Struktur

Entsprechend dem ERM werden Zusammenhänge zwischen Objekten mit Beziehungen (Relationsship Assoziation Types) dargestellt.

Objektsystem-Verhalten

Das Verhalten eines Objektsystems wird mit Hilfe von Aktionen und Triggern (= auslösende Bedingungen) beschrieben.

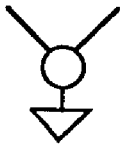
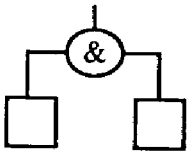
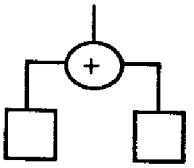
	Beschreibung	Graphisches Symbol
Ereignis Trigger	Auslösende Bedingungen für ein Ereignis	
Aktionen	Aktivität des Systems	
Aktionen Trigger mit logischem UND	Auslösende Bedingungen für zwei Aktionen	
Aktionen Trigger mit logischem ODER	Auslösende Bedingungen für entweder die eine oder die andere Aktion	

Abb. 3.2.2/35: Beschreibung des Objektsystem-Verhaltens

Das Objektsystem wird in einem nächsten Schritt in das Informations- und Datensystem überführt, das besteht aus

- einem Informationstypmodell, das die Informationstypen des Systems beschreibt
- einem Informationsprozessmodell, das die Informationsgewinnung der Initialinformationen und Input-Output-Prozeduren beschreibt
- einem Interaktionsmodell

Interessant ist an diesem Ansatz, daß er klar zwischen der Modellierung der realen Welt in einem konzeptuellen Schema einerseits und der Überführung des konzeptuellen Schemas in ein Informationssystem trennt.

Auch wenn dieses Hilfsmittel nicht sehr ausführlich und konkret in der Literatur beschriebenen ist, so zeigt es doch den Weg auf, den auch andere Verfasser bei der Suche nach semantisch mächtigeren Konstruktionshilfsmitteln gehen (vgl. Ferstl/Sinz (1990) und (1991)).

3.2.3. Dokumentation

Die Dokumentation soll die Aufgabe haben, die Entwurfsergebnisse aus den verschiedenen Phasen strukturiert und systematisch zu beschreiben. Ein Dokumentationssystem besteht aus inhaltlichen und organisatorischen Regelungen darüber, wann, von wem und in welcher Form die Elemente des externen, konzeptuellen und internen Schemas der Datenbank zu beschreiben sind. Schließlich wird noch ein technisches Hilfsmittel für die komfortable Verwaltung benötigt. Das Dokumentationssystem besteht aus:

(1) Regeln für das einzelne Datenelement (= Attribut), d. h. semantische sowie syntaktische Regeln mit dem Ziel einer Datenelement-Standardisierung

(2) Regeln für die Aufnahme der Datenelemente in ein Data Dictionary und für deren Selektion (Datenelement-Administration)
(3) dem Datenwörterbuch samt Datenverwaltungssystem (Data Dictionary)

Regeln für das einzelne Datenelement

Datenelemente sind die kleinsten organisatorischen Einheiten zur semantischen Strukturierung von Informationen.

Beispiel: *nicht* *DATUM*
nicht *LIEFERDATUM*
nicht *LIEFERDATUM AUSLAND*
sondern *LIEFERDATUM AUSLAND LUFTFRACHT*

Ziel ist es, mit einer begrenzten Zahl standardisierbarer Datenelemente eine

- überschaubare und vergleichbare
- wartbare und wiederverwendbare
- auf Konsistenz logisch prüfbare

Beschreibung aller Dateneingaben und Datenausgaben, Datenspeicherungen und Datentransformationen zu ermöglichen.

Datenelemente bilden die Attribute der Objekttypen und der Beziehungstypen, die dann so weit wie möglich standardisiert beschrieben werden. Ziel ist es, individuelle Attribute zu vermeiden und durch standardisierte Datenelemente zu ersetzen.

Dabei sind vielfältige Schwierigkeiten zu überwinden, die z. B. daraus resultieren,

daß unklar ist, wann von einem neuen Datenelement auszugehen ist und wann es sich nur um die Ausprägung eines alten handelt?

Beispiel: *AUFTRAG* *Auftrag, geplant*
Auftrag, freigegeben
Auftrag, angearbeitet
Auftrag, fertiggestellt

Lösung: Jedes Attribut eines Datenobjekts (hier Auftragsstatus) muß eine bestimmte Wertedomäne erhalten.

daß unklar ist, was im einzelnen als Meta-Datum zu begreifen ist?

Beispiel: *Schlüsseltyp* - *Schlüsselausprägung* - *Schlüsselinhalt*
(z. B. Länderkennziffer, Fehlernummer)
Meta-Meta-Datum *Meta-Datum* *Datum*

Lösung: Schlüsseltyp wird im Data Dictionary beschrieben mit allen Ausprägungen und Inhalten.

Organisatorische Regelungen der Datenadministration

DV-technische und organisatorische Regelungen für die Aufnahme von Datenelementen in die Datenbank sind für das gesamte Unternehmen wichtig, da integrierte Anwendungssysteme nur bei jederzeit vollständigen Datenbeständen funktionieren.

Organisatorisch ist festzulegen, ob die Datenadministration zentral oder dezentral erfolgen soll. Es ist eine Aufgabe, die dauernd und in sämtlichen Phasen des Software-Lebenszyklusses wahrzunehmen ist. Bei größeren Unternehmen ist daher neben einer zentralen Instanz mit methodischer Richtlinienkompetenz eine dezentrale Parallelorganisation einzurichten, die die Vollständigkeit der Datenbestände tagtäglich sichert.

Beispiel:
Einrichtung von neuen Artikelstammdaten in den Vertriebsgesellschaften, Veränderung von Attributen in den lokalen Personalstammdaten aufgrund lokaler Steuergesetze.

Im ANSI-SPARC-3 Schema-Modell werden organisatorisch jeweils den drei Schemata zugeordnete Administrationseinheiten unterschieden:

Schema	Organisatorische Einheit	Aufgaben
Konzeptuelles Schema	Unternehmensadministrator	- Untersuchung des Informationsbedarfs und des Informationsflusses in einer Organisation - Umsetzung in konzeptuelles Modell
Externes Schema	Anwendungsadministrator	- Untersuchung des Informationsbedarfs von Anwendungsprogrammen/Benutzern - Ableitung externer Schemata aus konzeptuellem Modell - Benutzerrechte vergeben
Internes Schema	Datenbankadministrator	- Untersuchung der Anforderungen des konzeptuellen Schemas an Speichertechnologie/organisation - Abbildung des konzeptuellen Modells mittels einer physischen Beschreibungssprache - Datenschutz und Datensicherung

Abb. 3.2.3/1: Organisatorische Administrationsebenen im 3-Schema-Modell (vgl. Lockemann/Dittrich (1987))

Der Unternehmensadministrator ist dem betriebswirtschaftlichen Management zuzurechnen. Nur in wenigen Unternehmen existieren heute dafür spezielle Organisationseinheiten; es bieten sich daher Stellen wie das Unternehmenscontrolling, die Unternehmensplänung oder auch Organisationsabteilungen für diese Aufgabe an.

Der Anwendungsadministrator befaßt sich auf der eher operativen Ebene mit den Schnittstellenanforderungen der Anwendungsprogramme und der erforderlichen Benutzeroberfläche für die Nutzer der Fachabteilungen. Es ist eine Aufgabe an der Grenze zwischen betriebswirtschaftlichem und DV-technischem Management.

Der Datenbankadministrator erfüllt innerhalb der DV-Organisation die unmittelbar mit der technischen Handhabung der Datenbanksysteme verbundenen Aufgaben.

Data Dictionary

Datenwörterbücher dienen der zusammenfassenden Dokumentation des konzeptuellen, des externen und später des internen Schemas (Datenobjekte, -beziehungen, -operatoren).

Ziel ist die Beschreibung und Verwaltung von Meta-Daten (d.h. Daten über Daten) und nicht der Inhalte der beschriebenen Daten.

Die Verwaltung der Meta-Daten in einem speziellen System ermöglicht einen logisch konsistenten Datenbestand, der in gleicher Struktur allen dezentralen Nutzern dargestellt wird.

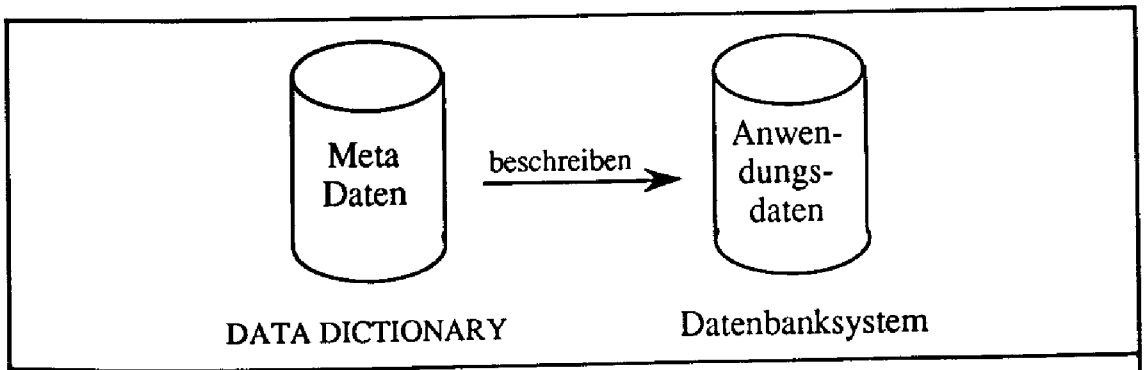


Abb. 3.2.3/2: Beziehung zwischen Meta-Daten und Anwendungsdaten (vgl. Lockemann/ Dittrich (1987))

Zu unterscheiden sind 3 Bereiche eines DATA DICTIONARIES:

* Das BASIS-Data Dictionary enthält alle Informationen über Datenbanktabellen, -spalten und -indizes, Datenschutz- und Integritätsregeln, Sichtendefinitionen und Datenstatistiken.
* Das erweiterte Data Dictionary enthält alle Informationen über die Anwendungsobjekte (Masken, Reports, Programmschnittstellen) und konzentriert sich somit auf das externe Schema einer Datenbank.
* Das globale Data Dictionary gibt einen Überblick über alle Datenbanken eines Rechnersystems, deren Namen, Eigner, Zuordnung zu Datenspeicherbereichen.

Inhalte	Betriebswirtschaftliche Nutzer			DV-Nutzer	
	Top Management	Linien-management	Informations-management	Datenbank-administration	System-entwickler
Datenarchitektur, global	●	○			
Fachbegriffe, semantische	○	●	○		
konzeptuelles Schema		○	●	●	○
Externes Schema			○	●	●
Internes Schema				●	●

Abb. 3.2.3/3: Nutzerprofil eines Data Dictionaries auf den Ebenen eines Unternehmens

Das DATA DICTIONARY ist ein Instrument für alle Ebenen des Managements und der Systementwicklung:

Das Top-Management übt damit die Kontrolle über die Datenbestände aus, die eine wesentliche Ressource der Unternehmung darstellen. Es kann dadurch sicherstellen, daß sich Änderungen in der Unternehmungspolitik in Änderungen der Datenarchitektur widerspiegeln.

Das Linien-Management nutzt vornehmlich den Überblick über das konzeptuelle Schema und die semantische Klärung wichtiger Fachbegriffe. Dadurch werden organisatorische Schnittstellen verfeinert, Unklarheiten und Doppelarbeit vermieden.

Das Informations-Management kann sicherstellen, daß die technologische, ökonomische und organisatorische Ebene der Daten-Architektur aus „einem Guß" gestaltet wird. Da sich die Systementwicklung von solchen Architekturen leicht über mehr als ein Jahr hinziehen kann, vermeidet es dadurch Integrationsprobleme.

Für die DV-Nutzer dient das Daten-Wörterbuch schließlich als Entwurfs-/Entwicklungsumgebung sowie als Instrument der Systemdokumentation und -verbesserung.

Für den Systementwickler dient das Data Dictionary (vgl. Ortner (1991), S. 424)

- der strukturierten Dokumentation der Ergebnisse des Entwicklungsprozesses,
- der Vergabe und Verwaltung von Entwicklungsstandards,
- der Darstellung von Querbeziehungen zwischen den verschiedenen Entwicklungsergebnissen (z. B. Masken, Datenbankrelationen, Dateistrukturen)

Das Data Dictionary enthält auch Fachbegriffe mit unternehmensspezifischen Definitionen, die die Informationsobjekte in ihrer Bedeutung und ihres Gebrauchs in den Fachabteilungen kennzeichnen.

Die einheitliche Definition von Fachbegriffen ist seit langem in den Unternehmen als Notwendigkeit begriffen worden. Traditionell finden sich entsprechende Angaben in Begriffswörterbüchern und Handbüchern, das Data Dictionary übernimmt deren Aufgaben.

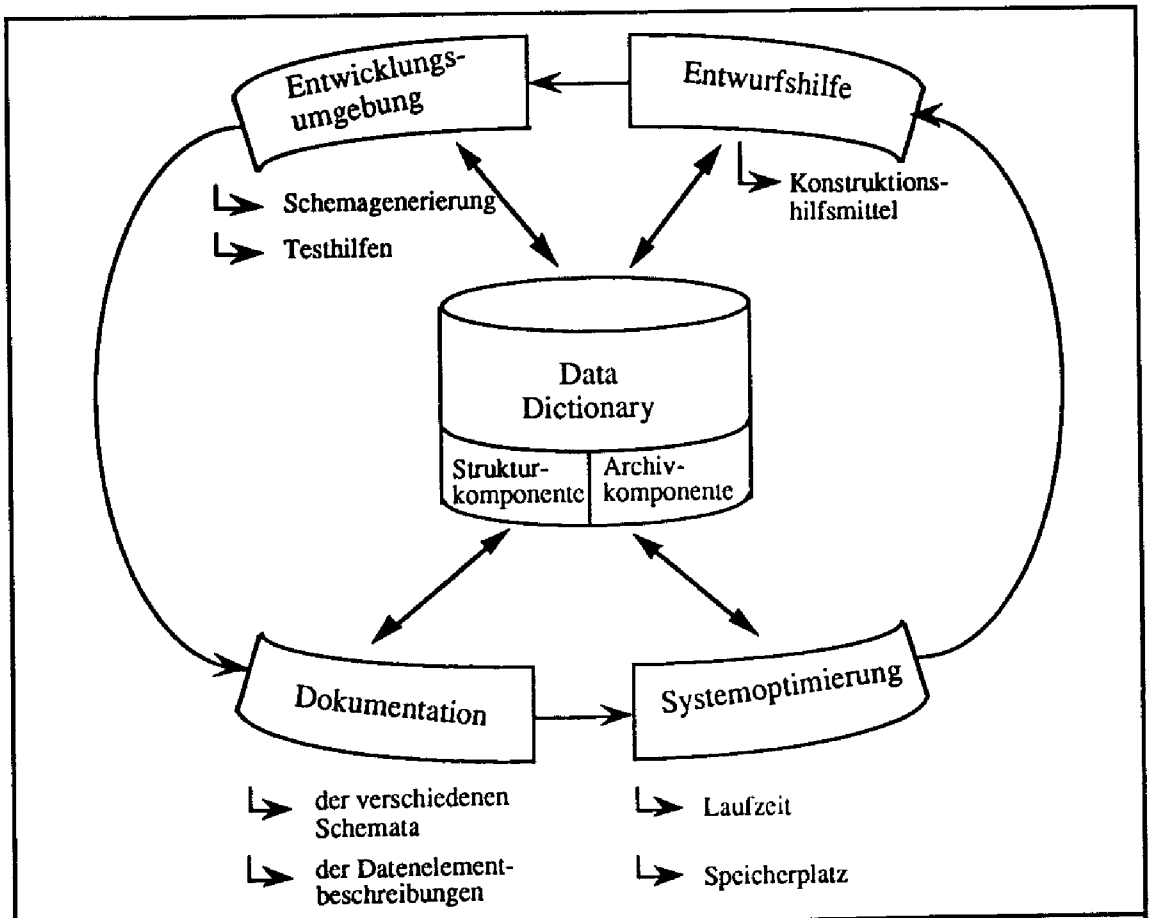


Abb. 3.2.3/4: Einsatz eines Data Dictionaries im Entwicklungszyklus eines Datenbanksystems

Das Data Dictionary (auch Data Repository) beschreibt alle Informationen, die im Entwicklungsprozeß der Datenbank und der darauf zugreifenden Funktions-Programmsysteme wichtig sind. Zum einen sind dies Strukturinformationen, mit deren Hilfe die verwendeten Entwicklungsobjekte, Schnittstellenstrukturen, Maskenstrukturen im Softwareentwicklungsprozeß beschrieben werden. Zum anderen werden Informationen über die vorhandenen Datenbestände archiviert. Dazu gehören:

-> Datenklassen und zugehörige Datenelemente
-> Informationsflüsse (Kontroll- und Nutzdatenflüsse)
-> Programmstrukturen der Anwendungssysteme
-> Schnittstellenbeschreibungen zwischen Datenbanken (hier auch Merkmale der Datenverteilung) und Anwendungssystemen.

Die globale DV-ARCHITEKTUR eines Unternehmens (Meta-Informationssystem) wird umfassend nicht nur mit den DV-Komponenten (Funktions-, Kommunikations- und Datenmodell) dokument. Anliegen ist auch, die technologischen Ressourcen und den organisatorischen Rahmen vollständig zu beschreiben.

	Konzeptionsdaten	**Implementierungsdaten**	**Betriebsdaten**
Konzeptuelles Schema	> Globale DV-Architektur > Semantisches Datenmodell > Semantische Begriffsklärungen	> Schema in ERM-Notationen > Objekt-/Struktur-/ Operatorendefinitionen	> Liste aller Namen > Organisationsanweisungen
Externes Schema	> Funktions-/Kommunikationsstrukturen der DV-Systemarchitektur	> Sichten in ERM-Notationen > Objekt-/Struktur-/Operatorendefinitionen	> Zugriffsrechte > Laufzeit-/ Speicherplatzverhalten
Internes Schema	> Technologieparameter der DV-Ressourcen	> Zugriffspfade > Systemtechnologie > Zuordnung zu Speichermedien	> Datenvolumen/ Speicherplatzbelegung > Datenbasisorganisation/-wiederherstellung

Abb. 3.2.3/5: Inhalte eines Data Dictionaries

Übliche Datenbanksysteme für formatierte Daten eignen sich nur begrenzt zur Implementierung eines DATA DICTIONARIES.

Anforderung	**Erläuterung**
Komplexe Datentypen	Im Software-Entwicklungszyklus werden zahlreiche, unterschiedlichste Dokumente erzeugt; die zum Teil aus anderen Dokumenten generiert werden. Komplexe Datentypen bedingen u.a. flexible Felder.
Rekursive Vernetzung	Die Dokumente spiegeln unterschiedliche Abstraktionsebenen wider und sind voneinander abhängig.
Komplexe Konsistenzbedingungen	Die Komplexität der Objekte und deren starke Vernetzung macht es notwendig, komplexe Konsistenzbedingungen zu überwachen.
Unterstützung der Versionenhaltung	In der Softwareentwicklung entstehen mehrere Versionen eines Dokuments, die nachvollziehbar verwaltet werden müssen.

Abb. 3.2.3/6: Anforderungen an ein DATA DICTIONARY (vgl. Wenner (1981), S. 35)

Da umfangreiche Dokumente mit sehr flexiblen Deskriptoren zu speichern sind, ist möglichst eine "non standard data base application" (d.h. Verwaltung von Graphiken, Texten und formatierten Daten) mit einer Status-/Versionenverwaltung (Abdeckung des gesamten Lebenszyklus vom Fachentwurf bis zum Rechenzentrums-Ablauf) anzustreben. Das Data Dictionary System sollte über eine Anbindung zu Entwicklungswerkzeugen (CASE-Tools) verfügen und heterogene Hard-/Softwarewelten unterstützen können, d.h. dort auch mit den erforderlichen Komponenten lauffähig sein.

Data-Dictionary	Basis-Datenbank	Anbieter
PREDICT Case	ADABAS	Software AG
DATA Repository	DB 2	IBM
DATAMANAGER	- spezialisiert	MSP
ROCHADE	- spezialisiert	R&O Software Technik GmbH
IDMS	IDMS	CA Computer Associates GmbH

Abb. 3.2.3/7: Beispiele für existierende Data Dictionary Systeme

3.3. Anwendungsbeispiele

3.3.1. Beispiel 1: Datenmodell der Bayrischen Motoren Werke AG (BMW), München

Die BMW AG verwendet ein modifiziertes Entity-Relationship-Modell, das auf einer vorgelagerten BSP-Studie aufbaut. In diesem Modell wird die Unternehmung in 4 semantischen Ebenen abgebildet, die jeweils aus einer Objekt- und einer Beziehungs-Schicht bestehen (vgl. Sneed/Gawron (1983), S. 717).

Die vier Ebenen sind:

1. Physische Organisation (organizational physical level)

Dort wird die reale Welt der Organisation charakterisiert. Sie umfaßt die Objekte: Organisationseinheiten (z. B. Sparten, Bereiche, Abteilungen, Gruppen, Mitarbeiter => Aufbauorganisation) und die zugehörigen Attribute, wie Bezeichnung, Größe, Ort, Qualifikationen sowie die Aktivitäten: (z. B. Ziele, Verantwortungsbereiche, Aufgabenbereiche, Aktivitäten => Ablauforganisation) samt den zugehörigen Attributen, wie Aktivitätsart, Bezeichnung, Dauer und Häufigkeit, Auslöser.

2. Logische Organisation (organizational logical level)

Diese Ebene beschreibt die Anwendungssysteme, die wiederum zerfallen in

-> die (hierarchisch abgestuften) Datenobjekte mit deren Attributen (Art, Name, Größe, Auftreten)
-> die Aktivitäten : Anwendungssysteme mit ihrer Funktions- und Programmstruktur als Attribute

3. Logische Technikebene (technical logical level)

Hiermit wird die logische Struktur der Datenverarbeitungssysteme gekennzeichnet, die hierarchisch in ein Funktionsmodell und ein Datenmodell zerlegt wird.

4. Physische Technikebene (physical technical level)

Diese beschreibt die Hardware-Objekte, die zur Speicherung von Datenobjekten vorhanden sind und die Softwaresysteme, die die Datenobjekte aktiv bearbeiten können.

Ebene	Objekte	Funktionen (actions)
1. Organisation (physical organizational schema)	- Unternehmung - Sparte - Abteilung - Gruppe - Mitarbeiter	- Ziele - Verantwortungsbereiche - Aufgaben - Projekte - Individuelle Aktivitäten
2. Anwendungssysteme (organizational logical schema)	- Anwendungsressourcen - Datenressourcen - Datenobjekte - Datengruppen - Datenelemente	- Zweck - Anwendung - Prozeß - Funktionsgruppen - Elementarfunktion
3. DV-Systeme (technical logical schema)	- Datenbestände - Datenbanken - Datensätze - Datenmodule - Datenfelder	- Programmarchitektur - Programmsystem - Programme
4. Technik (physical technical schema)	- Datenspeicher - Filestrukturen	- Code-Module - Code-Blöcke

Abb. 3.3.1/1: Semantische Ebenen bei BMW (vgl. Sneed/Gawron (1983))

Die Objekte aller vier Ebenen werden durch vier Arten von Beziehungen miteinander verbunden:

- Hierarchische Beziehungen (hierarchical relationships)

Hierarchische Beziehungen sind vertikale Verbindungen zwischen Objekten oder Aktivitäten auf einer semantischen Ebene, d.h. ein übergeordnetes Objekt besteht aus n untergeordneten Objekten während umgekehrt das untergeordnete Objekt Bestandteil des übergeordneten Objekts ist.

- Prozeß-Beziehungen (processing relationsships)

Prozeß-Beziehungen beschreiben horizontale Input-Output-Zusammenhänge auf dem gleichen semantischen Niveau zwischen Objekten und Aktivitäten. Eine Aktivität schafft, verändert oder verwendet ein Objekt; während andererseits das Datenobjekt durch die Aktivität passiv geschaffen, verändert oder genutzt wird. Wichtig ist dabei, das solche Prozessbeziehungen nur auf einem hierarchischen Niveau zulässig sind.

- Logische Beziehungen (associative relationsships)

Logische Beziehungen beschreiben logische oder zeitliche Verbindungen auf der gleichen hierarchischen und semantischen Ebene zwischen verschiedenen Objekten oder Aktivitäten, etwa

- zu einem Kunden gehört ein Auftrag, ein Kundenkonto etc.
- einem Auftrag folgt eine Lieferung, die Rechnungsstellung usw.

- Abbildungsbeziehungen (mapping relationships)

Abbildungsbeziehungen stellen diagonale Abstraktions- oder Konkretisierungsverbindungen zwischen Objekten oder Funktionen auf dem gleichen semantischen Niveau und korrespon-

dierenden Objekten bzw. Funktionen auf dem nächst höheren oder nächst niedrigeren Niveau dar.

Den Datenobjekten auf jeder der vier Ebenen werden innerhalb des Datenmodells der BMW AG Attribute zugeordnet. Diese Attributstrukturen sind standardisiert und werden einheitlich verbal und graphisch beschrieben. BMW verwendet dabei ein von IBM entwickeltes Schema.

3.3.2. *Beispiel 2: Raffineriemodell aus Kuwait*

Das "Petroleum Technology Department" des "Kuwait Institute for Scientific Research" betreibt ein wissenschaftliches Labor, das Rohöltests und -analysen sowie statistische Auswertungen für unterschiedliche Einrichtungen und Unternehmen durchführt (z. B. für Ölgesellschaften oder die Regierung von Kuwait) (vgl. Al-Fedaghi (1983)).

Dabei fallen Datenmengen an, die computergestützt bearbeitet werden. Für die Datenmodellierung wird das Entity-Relationship-Modell verwendet. Folgende Gründe waren ausschlaggebend:

- Einfachheit	Einfach zu verstehendes Modell (auch für Erdölingenieure, Chemiefachleute)
- Vollständigkeit	Mit Hilfe einfacher Objekte lassen sich viele Anforderungen erfüllen.
- Flexibilität	Das Modell läßt sich flexibel auf unterschiedliche Problemstellungen erweitern.

Die Testphase des Rohöls läßt sich folgendermaßen darstellen:

Das Labor entnimmt das Rohöl den angelieferten Behältern. Diesen Behältern werden Proben entnommen, denen das Wasser entzogen wird.

In der anschließenden Destillation wird die Probe in verschiedene Bestandteile zerlegt. Dabei wird nach "Atmospheric-Pressure-Fractions" und "Low-Pressure-Fractions" unterschieden. Die wichtigsten Attribute der Bestandteile sind das Gewicht, die Dichte und das Volumen. Die einzelnen Bestandteile können nicht als ein Entity dargestellt werden, weil sich die Attribute der Bestandteile unterscheiden.

Mit den Bestandteilen werden Versuche durchgeführt, wobei diese wiederholt vorgenommen werden. Die Versuche werden für Auswertungszwecke dokumentiert. Wichtige Attribute dabei sind der Mitarbeiter, der den Versuch durchgeführt hat und das Datum.

Die Rückstände werden weiteren Destillationsvorgängen ausgesetzt. Die dabei gewonnenen Bestandteile werden nach unterschiedlichen Merkmalen klassifiziert.

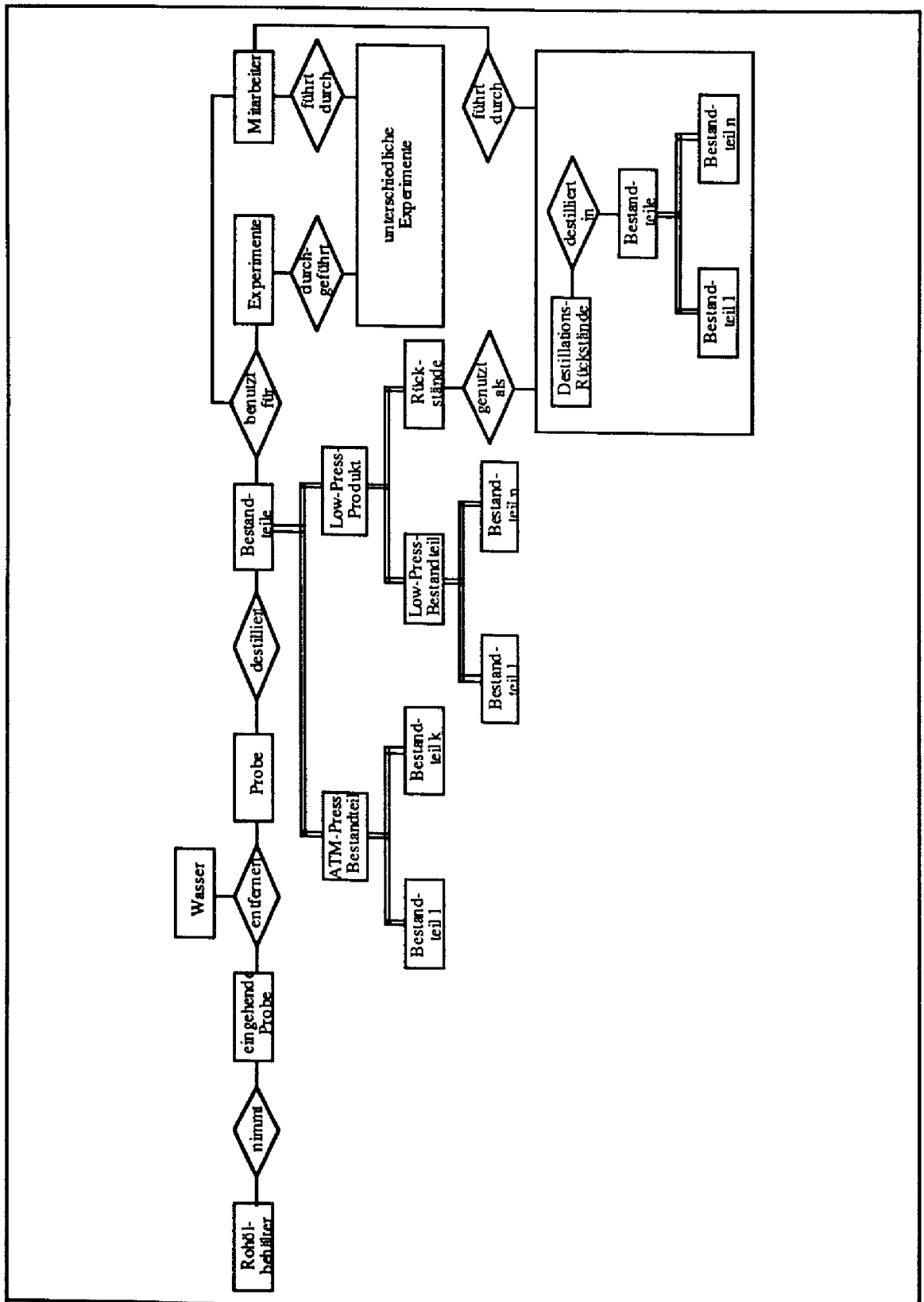


Abb. 3.3.2/1: Strukturen des Raffineriemodells (vgl. Al Fedaghi (1983), Seite 767)

Für das Raffinerie-Modell werden folgende Erweiterungen vorgeschlagen:

- Direkte oder berechenbare Attribute

Es werden zwei Arten von Attributen unterschieden:

- direkte (measured) Attribute: Attributausprägungen lassen sich direkt der Realwelt entnehmen.
- berechenbare (calculated) Attribute: Attributausprägungen lassen sich aus anderen Attributausprägungen bestimmen, d. h. es existieren Beziehungen zwischen den Attributen.

Beispiel für berechenbare Attribute: Zu dem Entity LUFT gehört das Attribut LUFT-DICHTE, das aus anderen Attributen berechnet wird. Aus Redundanzgründen werden berechenbare Attribute nicht in das Datenbanksystem aufgenommen. Im ERM werden sie für den Systementwickler mit aufgenommen.

Durch diese zusätzlichen Attributbeziehungen bilden Entities und Attribute eine komplexe Struktur.

- Entity-Relationship-Attribut - Konflikt

Dieser Konflikt tritt auf, wenn bei einem Objekt nicht eindeutig zwischen Entity, Relationship, Attribut oder Ausprägung unterschieden werden kann.

Beispiel Attribut GEWICHT:

GEWICHT läßt sich durch den Typ des Materials (z. B. Messing, Stahl) und die physikalische Dichte des Gewichts beschreiben. Im Entity STICHPROBE kommt Gewicht als Attribut vor, im Entity GEWICHT kommen Typ und Dichte vor, aus denen sich das Gewicht berechnen läßt. Durch die Beziehung GEWOGEN-VON werden quasi zwei gleiche Attribute in Beziehung gesetzt.

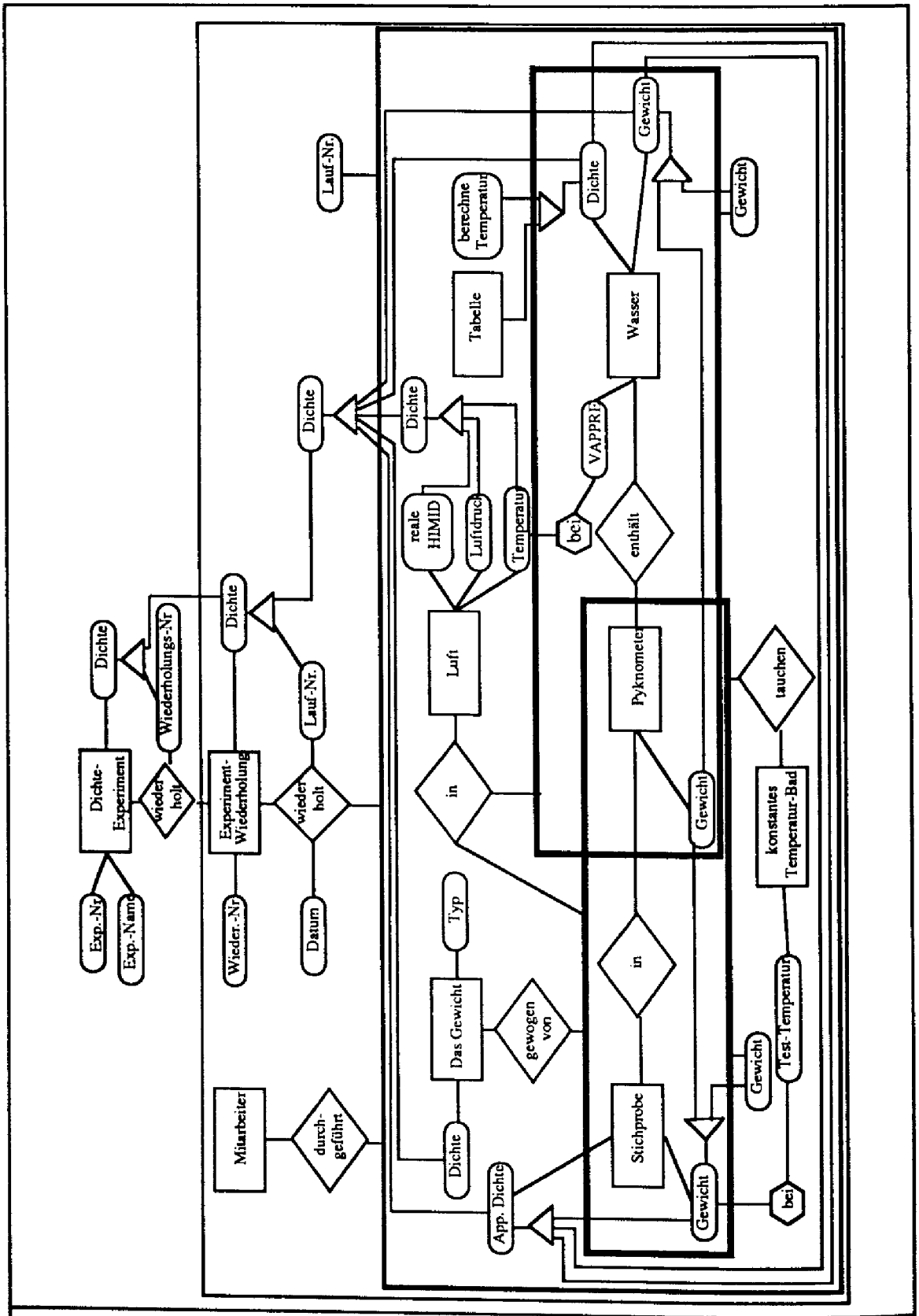


Abb. 3.3.2/2: Beispiel für Attributkonstrukte und für den Entity-Relationship-Attribut - Konflikt (vgl. Al-Fedaghi(1983), S. 772)

- Selektive Unterdrückung

Bei der selektiven Unterdrückung wird ein Teil des Modells mit Alternativen versehen.

Beispiel:
Das Entity STICHPROBE wird durch das Attribut VOLUMEN gekennzeichnet, das sich aus den Attributen DICHTE und GEWICHT berechnen läßt. Wird ein solcher Sachverhalt mit gestrichelten Linien und einem Knoten mit dem Operator OR gekennzeichnet, so kann das VOLUMEN direkt eingegeben werden oder aus DICHTE und GEWICHT berechnet werden.

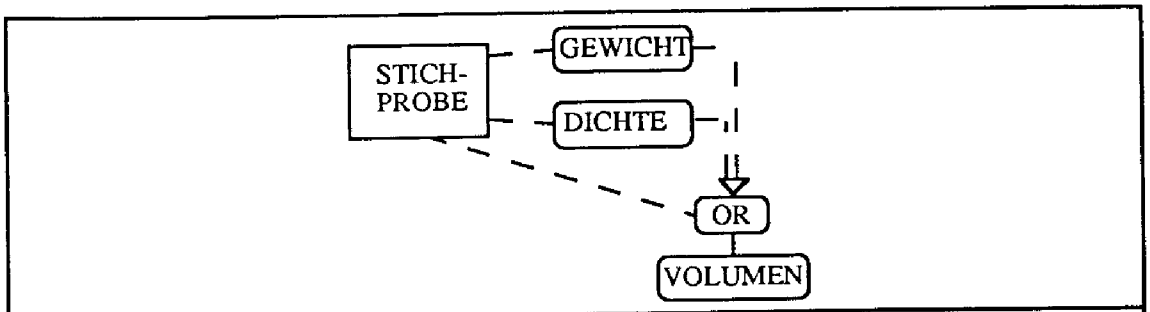


Abb. 3.3.2/3: Beispiel für selektive Unterdrückung im Raffineriemodell

- Partielle Generalisierung

Es werden Attribute zwischen Objekten vererbt, was durch doppelte Linien darstellt wird.

Beispiel: Rohöl-BESTANDTEILE
Die Erdöl-Rückstände sind ein Bestandteil des Rohöls, das bei der Destillation entsteht. Es existieren Attributausprägungen, die sich aus vererbbaren Attributen ermitteln lassen. Problem ist, daß das ERM dadurch sehr redundant ist. Das Gewicht aller Rückstände ergibt das Gesamtgewicht. Dazu wird die Attribut-Gleichheits (AE=Attribute Equality)-Beziehung benutzt, die eine konstante Beziehung ist und die Gleichheit von Attributen bzw. Attributausprägungen darstellt.

Nichtvererbbare Attribute wie z. B. Bestandteilnr. werden durch doppelte Kreise dargestellt.

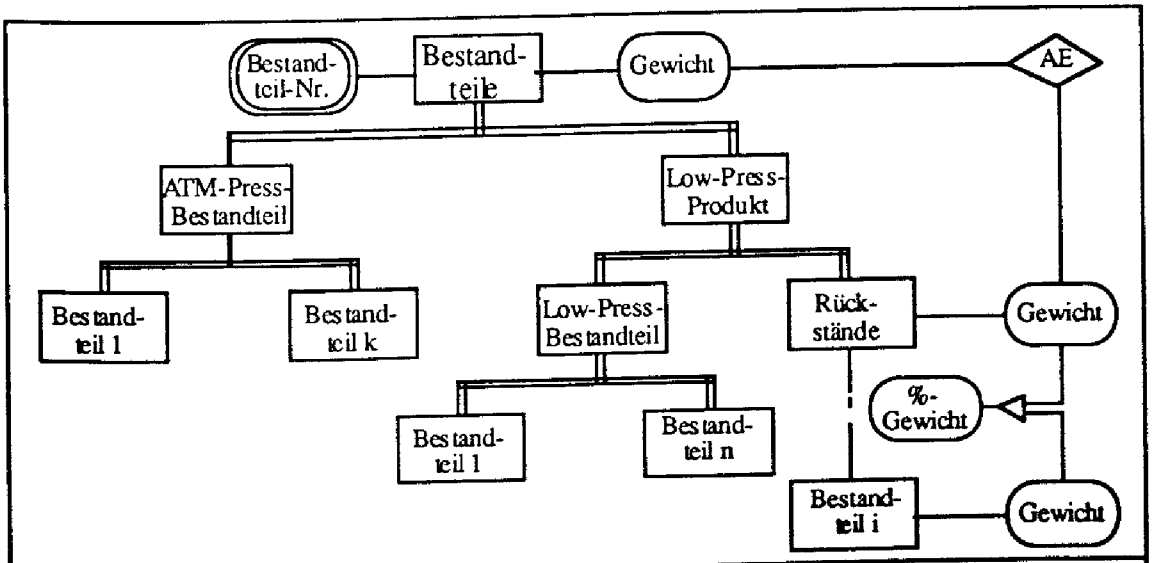


Abb. 3.3.2/4: Beispiel für partielle Generalisierung (vgl. Al-Fedaghi (1983), S. 775)

Nebenbedingungen

Nebenbedingungen sichern die Struktur, die Integrität und die operationalen Abhängigkeiten des Modells. Um Abhängigkeiten zwischen Attributen modellieren zu können, werden Hexagone eingeführt.

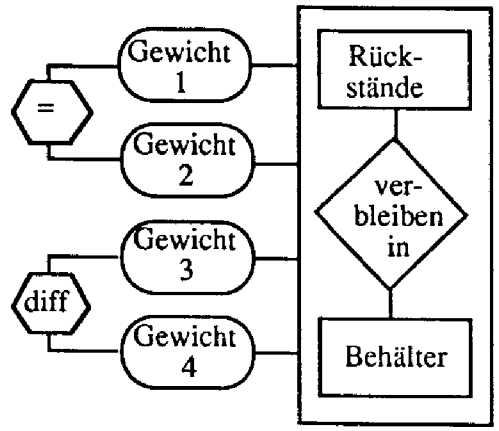


Abb. 3.3.2/5: Formulierung von gleichen bzw. ungleichen Attributausprägungen

Die Anwendung zeigt, daß im konkreten Fall Anforderungen entstehen können, die das semantische Leistungsvermögen des Entity-Relationship-Modell übersteigen.

4. Datenmodellierung (= logische Datenmodellierung)

4.1. Kennzeichnung

Ziel ist es, die fachliche Datenstruktur in DV-gerechter Form in einer Datenbankstruktur zu beschreiben. Dieses datenbankgerechte Abbild der fachlichen Datenstruktur sei als Datenmodell bezeichnet (vgl. Wedekind (1981), S.49)

Der Unterschied zwischen semantischer Modellierung und logischer Modellierung muß auch in den verwendeten Begriffen deutlich werden.

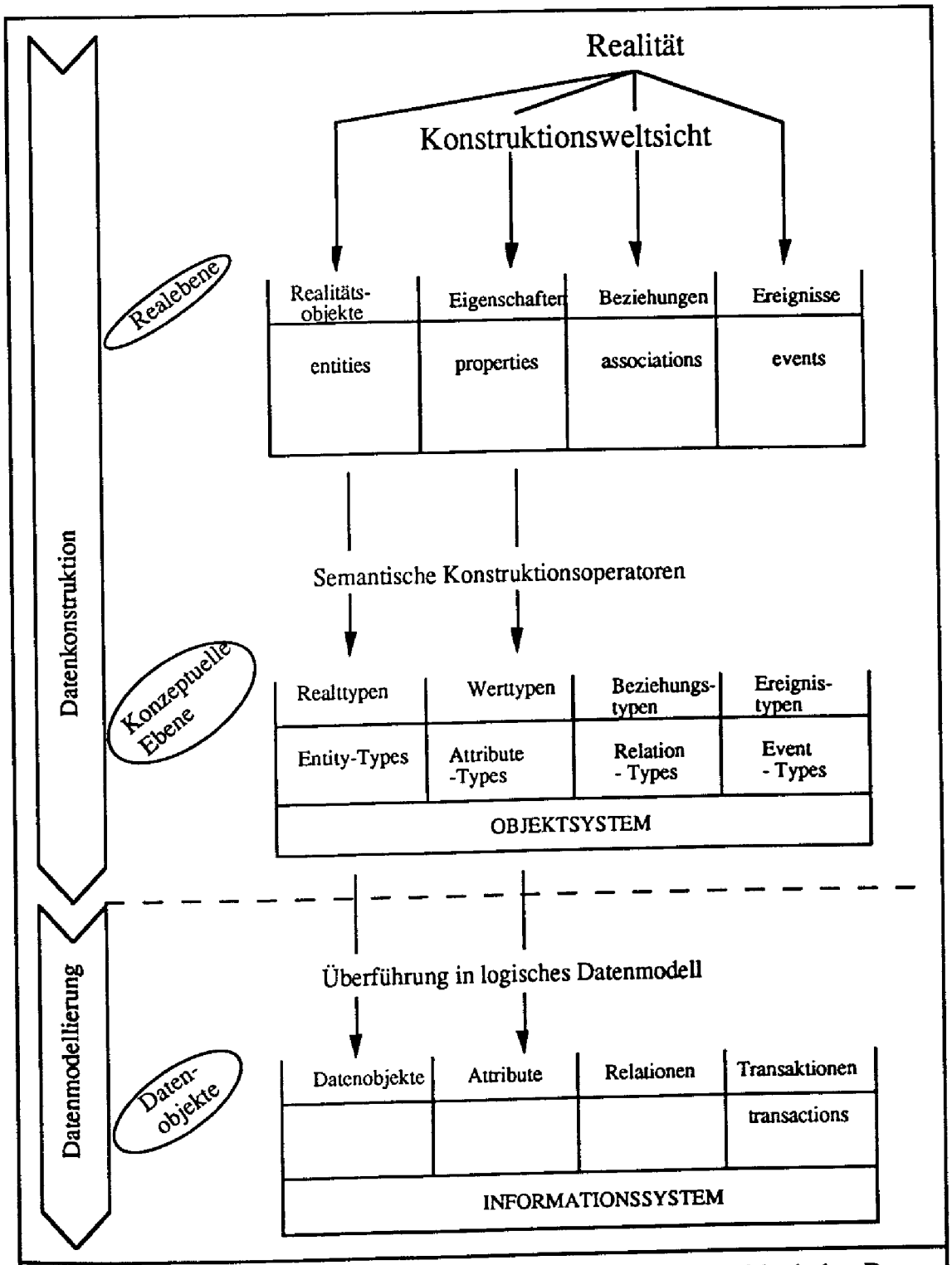


Abb. 4.1/1: Begriffliche Unterscheidung zwischen semantischer und logischer Datenmodellierung

Es wird das im Konstruktionsprozeß entwickelte abstrakte Modell der Daten- und Prozeßstrukturen in einem Informationssystem fixiert. Dazu sind

(1) die abstrakten Begriffe mit eindeutigen, DV-gerechten Begriffen zu belegen,
(2) die Datenstrukturen der einzelnen Objekte und Beziehungen in Form von Attributstrukturen (Datentypen, Feldlängen etc.) zu definieren,
(3) die zulässigen Eigenschaften von Objekten, Relationen und zugehörigen Attributen zu fixieren,
(4) zeitliche und inhaltliche Integritätsbedingungen zu beschreiben,
(5) die Prozeduren zu beschreiben, die interne oder externe Ereignisse des abstrakten Datenmodells abbilden sollen,
(6) die Zeitbegriffe zu definieren, die das Daten- und Funktionsmodell erfassen soll.

Besonders die Daten- und Zeitstrukturen sind abhängig vom gewünschten externen Modell der Datenbank, d.h. welche Abfragen und auch Eingaben diese durch Nutzer und Programme ermöglichen soll.

Die logische Modellierung ist der abschließende Schritt der fachlichen Anforderungsanalyse und mündet in den Systementwurf und in einen Anforderungskatalog an das zu wählende Datenbanksystem; d.h. dieser Schritt bildet die Schnittstelle zwischen betriebswirtschaftlich-fachlicher Analyse und DV-Systementwicklung.

Aufgrund der verfügbaren Datenbanksysteme und deren Leistungsvermögen ist es dabei unter Umständen notwendig, das Ergebnis des Konstruktionsprozesses (= semantisches Datenmodell) zu modifizieren, um das jeweilige Hard-/Softwaresystem sinnvoll nutzen zu können.

4.2. Aufgaben

Auf der Ebene der logischen Modellierung soll wiederum unterschieden werden zwischen dem konzeptuellen Schema und dem externen Schema.

Das konzeptuelle Schema setzt das semantische Modell in Bezug zu einem konkreten, logischen Datenmodelltyp (etwa dem relationalen Datenmodell). Dazu wird geprüft, inwieweit die Struktur der Objekte und Beziehungen sich in einem bestimmten Datenmodell abbilden läßt. Falls keine Abbildung des semantischen Datenmodells auf logischer Ebene möglich ist, ist ein Rücksprung in die Konstruktionsphase notwendig.

Während das konzeptuelle Schema neutral ist, repräsentiert das externe Schema die Sichten durch Benutzer und Programme sowie die hard- und softwaremäßige Realisierbarkeit.

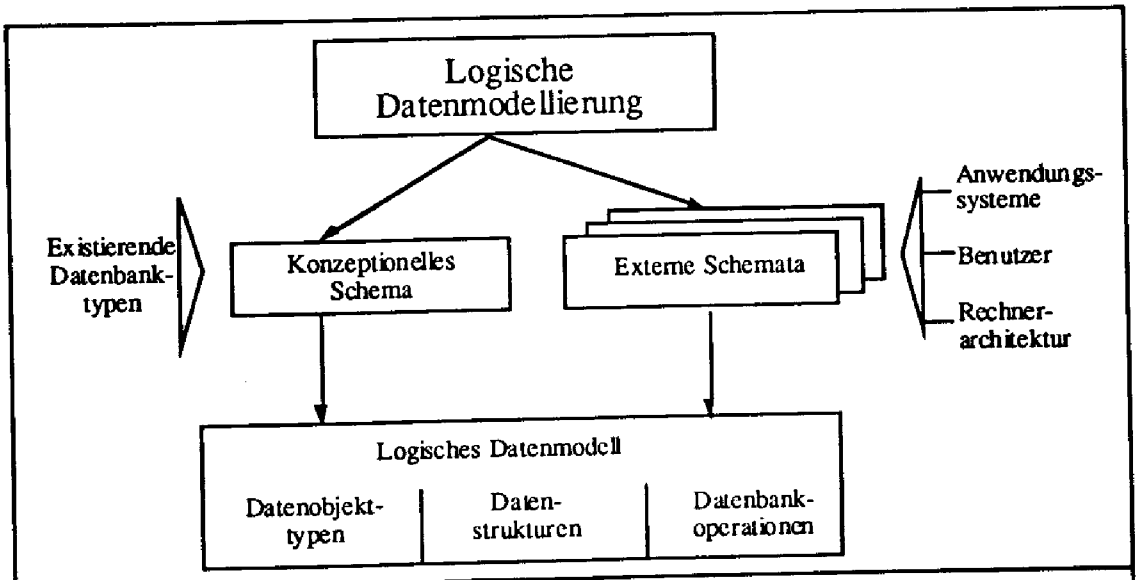


Abb. 4.2/1: Prozeß der logischen Datenmodellierung

4.2.1. Modellierung des konzeptionellen Schemas

Das konzeptionelle Schema umfaßt das statische und das dynamische Datenmodell.

Es ist darauf zu achten, daß zwar alle relevanten statischen und dynamischen Aspekte beschrieben werden, doch sollte stets von Gesichtspunkten der späteren Implementierung abstrahiert werden (z. B. von der Benutzersicht, Maschineneffizienz, physischen Datenorganisation).

4.2.1.1. Statische Modellierung

Datenobjekte

Bei den Datenobjekten wird versucht, alle Attribute in einem zusammengehörigen Element des logischen Datenmodells (etwa einer Relation) abzubilden (vgl. Mayr/Dittrich/Lockemann (1990), S. 518). Da eine Identifizierung der Datenobjekte durch deren semantische Eigenschaften möglich ist, entsprechende Operatoren jedoch bei den üblichen logischen Datenmodellen fehlen, werden die Datenobjekte mit identifizierenden Attributtypen (sogenannten Schlüsselattributen) versehen. Diese Schlüsselattribute können natürlich vorhanden sein, müssen jedoch in der Regel durch Nummernsysteme künstlich eingeführt werden.

Attribute der Datenobjekte

Bei den Attributen sind mögliche Ausprägungen (z. B. Zeichen, Graphiken, Prozeduren, Sub-Datenmodelle) und deren Wertedomänen zu analysieren.

Während für mögliche Werteausprägungen theoretisch kaum Grenzen gesetzt sind, bieten die heute verfügbaren Datenbanksysteme lediglich eingeschränkte Möglichkeiten: Zumeist sind nur klassische Datentypen zugelassen, z. B. numerische oder alphanumerische Zeichenketten mit fester Maximallänge.

Im logischen Modell ist es daher teilweise notwendig, die Attribute des semantischen Datenmodells um zusätzliche („künstliche") Attribute zu erweitern, um komplexe Datentypen beispielsweise über eine Zerlegung in mehrere Attribute abbilden zu können. Scheitert diese

Möglichkeit, weil sich die Anforderungen (z. B. bei Graphiken) mit klassischen Datentypen nicht erfüllen lassen, sind unter Umständen bestimmte Elemente des semantischen Datenmodells außerhalb der Datenbank in separaten Dateistrukturen abzuspeichern.

Beziehungen zwischen Datenobjekten

Beziehungen zwischen den Datenobjekten sind im logischen Datenmodell ebenfalls mit eindeutig identifizierenden Attributen zu versehen. Dieses Schlüsselattribut kann sich aus denen der beteiligten Datenobjekte ergeben.

4.2.1.2. Dynamische Modellierung

Bei der dynamischen Datenmodellierung sind die zulässigen Operationen zu definieren und dadurch die zulässigen (dynamischen) Entwicklungen des Datenbestandes von einem (statischen) Zustand zum nächsten zu beschreiben (vgl. Brodie/Ridjanowic (1984)).

Die Operationen werden durch die Operatoren bestimmt, die in einem logischen Datenmodell verfügbar sind. Sie bestimmen drei Momente einer Datenbank:

(1) Wie entwickelt sich eine Datenbank von einem (statischen) Zustand zum nächsten Zustand? (Zustandsübergänge)
(2) Welche Zustände sind zulässig und wie wird die Zulässigkeit gesichert? (Integritätsbedingungen)
(3) Wie nachvollziehbar ist die zeitliche Zustandsgeschichte einer Datenbank?

Bei den **Zustandsübergängen** sind unter anderem zu beschreiben

- die zulässigen Operationen auf Datenobjekte (Transaktionen, Zugriffe, Ausgaben)
- die möglichen Eigenschaften von Operationen (insbesondere Wechselwirkungen zwischen Operationen),
- die Behandlung von Ausnahmesituationen (z. B. Fehlerhandhabung).

Diese Komponenten werden in einem TRANSAKTIONSMODELL festgehalten, das die zulässigen Operationen beschreibt.

Beispiel:
Eine Hotelreservierung betrifft die Datenelemente
- *Hotelzimmer,*
- *Hotelkunde,*
- *Reservierungstag.*

Operationen sind die einzig zulässigen Aktionen, um den Zustand von Attributen, Entities oder Relationen zu ändern. Damit kann das Verhalten eines Datenobjekts vollständig durch die zulässigen Operationen erklärt werden. Operationen können bestehen aus:

-> Einfügeaktionen,
-> Löschaktionen,
-> Änderungsaktionen,
-> Suchaktionen.

Sie setzen sich aus diesen Aktionen durch parallele oder sequentielle Abfolgen, durch Wiederholungen oder durch Entscheidungen zusammen. Ziel ist es, unzulässige Zustände des Datenbestandes (sogenannte Anomalien) zu vermeiden.

Aktionen	Mögliche Anomalien	Beschreibung	Beispiel
EINFÜGEN	Einfügeanomalien	Einfügen eines Entities (Satzes) mit einer unbekannten Identifikation des korrespondierenden Entities	- Einfügen eines Prüfungsergebnisses für eine Matrikelnr., für die der entsprechende Student dem System noch nicht bekannt ist
LÖSCHEN	Entfernungsanomalien	Entfernen eines Entities mit einer Identifikation, die noch von korrespondierenden Entities benötigt wird	- Löschung eines Studenten, obwohl noch Prüfungen unter seiner Matrikelnr. geführt werden
ÄNDERN	Änderungsanomalien	Änderung eines Attributs, ohne daß abhängige Attribute ebenfalls geändert werden	- Änderung einer Prüfungsnote, ohne daß gleichzeitig die Durchschnitts-Note des Studenten sich ändert

Abb. 4.2.1/1: Arten von Anomalien

Anomalien können durch syntaktische oder semantische Operatoren vermieden werden. Syntaktische Operationen kennzeichnen zum Beispiel die Attribute, die immer gleichzeitig geändert werden müssen und die gleiche Zeichenfolge aufweisen müssen. Semantische Operatoren überprüfen hingegen, ob der Bedeutungsgehalt der von einer Einfügung betroffenen Entities und Relationen stimmig ist.

Integritätsbedingungen lassen sich unterscheiden in **statische** Bedingungen, die die zulässigen Zustände sichern, und **dynamische** Bedingungen, die die zulässigen Zustandsübergänge beschreiben. Sie werden beschrieben mit Restriktionen hinsichtlich der Eigenschaften der Objekte und Beziehungen, die in Programmen dargestellt werden müssen.

Beispiele:
- keinen Kunden aus Stammdaten streichen, für den noch offene Posten existieren
- keinen Stoff einrichten, für den noch keine Stückliste und Teilestammdaten existieren.

Implizite Restriktionen folgen aus logischen Fernwirkungen anderer Objekte und Beziehungen.

	Explizite	Implizite
Dynamische Integritätsbedingung	> Die Beziehung "wird berechnet" (= Rechnung) kann erst dann eingerichtet werden, wenn die Beziehung "wird geliefert" (= Lieferschein) existiert	> Die Beziehung "wird berechnet" beeinflußt die umsatzabhängigen Boni des Kunden
Statische Integritätsbedingungen	> keinen Kunden aus Stammdaten streichen, für den noch offene Posten existieren	> keinen Stoff einrichten, für den noch keine Stückliste/-Teilestammdaten existieren

Abb. 4.2.1/2: Beispiele für Integritätsbedingungen

Zur Abbildung der **zeitlichen Historie einer Datenbank** existieren spezielle zeitorientierte Datenmodelle (vgl. weiter unten). Auch im Rahmen normaler Datenmodelle ist sicherzustellen, daß die zeitliche Entwicklung von Datenobjekten und Datenbeziehungen nachzuvollziehen ist. In normalen Datenbanken (sogenannten Snap-shot-databases) wird dieses üblicherweise über die Speicherung von Datenversionen gelöst.

4.2.2. Modellierung des Externen Schemas

Die Modellierung des externen Datenbankschematas zielt auf die Benutzersichten (Views), die von manuellen Benutzern, von vorgelagerten Programmsystemen des gleichen Rechners oder in Rechnerhierarchien benötigt werden. Es bildet den Informationsbedarf von vorgelagerten Systemen und indirekt von Teilbereichen des Unternehmens ab (Abteilungen, Mitarbeiter etc.). Das externe Schema ist ein Ausschnitt des konzeptuellen Schemas, der sich beispielsweise durch Benennungen unterscheidet. Da der Bedarf aller Nutzer letztlich das konzeptionelle Schema bestimmt, lassen sich beide Schemata nur in Abhängigkeit voneinander konstruieren.

Um das externe Schema aus dem konzeptuellen Schema abzuleiten, sind Entscheidungen zur Auswahl (Selektion), Zusammenfassung (Aggregation) und Verteilung (Fragmentierung) sowie der Benennung der Datenelemente zu treffen.

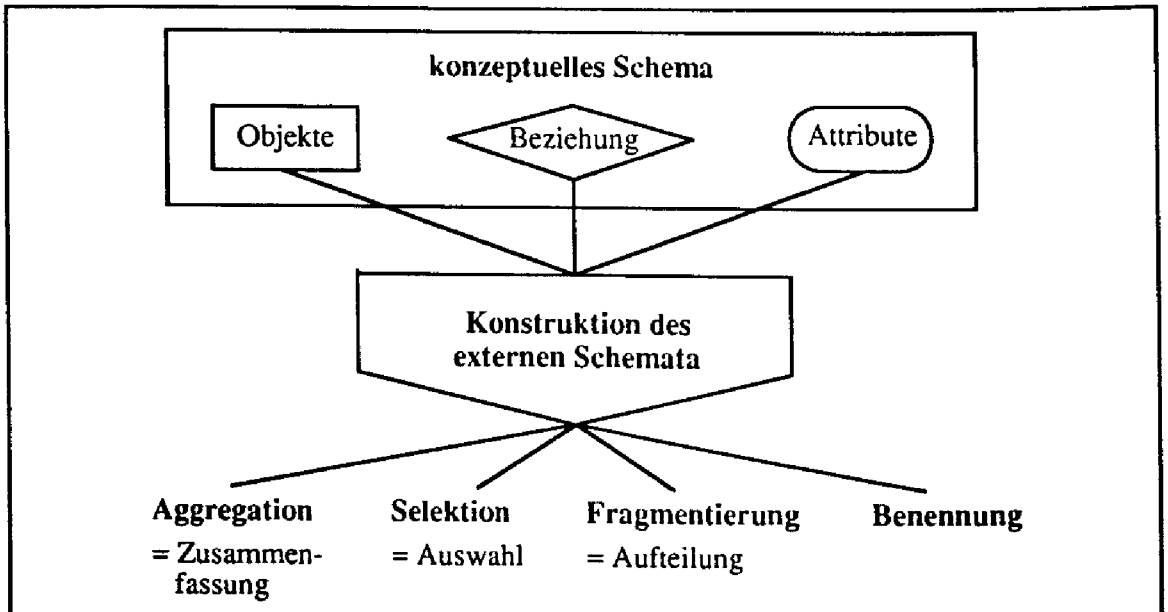


Abb. 4.2.2/1: Modellierungsentscheidungen bei der Ableitung der externen Schemata aus dem konzeptuellen Schema

Das externe Schema kann sich in der Detaillierung, der Benennung sowie in der Zusammenstellung von Attributmengen zu Objekten unterscheiden (vgl. Ling, (1988)).

Es bestehen enge logische Beziehungen zwischen dem konzeptuellen und dem externen Schema, die sich speziell bei Einfüge- und Änderungsoperationen bemerkbar machen. Probleme entstehen beispielsweise,

- wenn eine externe Sicht nur einen Teil der Attribute des Datenobjekts bzw. der Beziehungen des konzeptuellen Schemas enthält. Dann müssen bei anderen Views die für das konzeptuelle Schema fehlenden Attribute nachgefordert werden.

- wenn ein View synthetisierte Werte aufgrund von Aggregationsoperationen enthält. Wie soll ein solcher zusammengesetzter Attributwert beim Einfügen in die atomaren Attributstrukturen des konzeptuellen Schemas zerlegt werden? (vgl. Mayr/Dittrich/Lockemann (1990), S. 540))

In dynamischer Hinsicht können in den externen Schemata zusätzliche Integritätsbedingungen vorgesehen werden, um die Nutzer- oder Programmschnittstellenanforderungen zu erfüllen.

Das externe Schema wird durch Zugriffe der Datenbanksprachen realisiert.

4.3. Datenmodelle

4.3.1. Typisierung von Datenmodellen

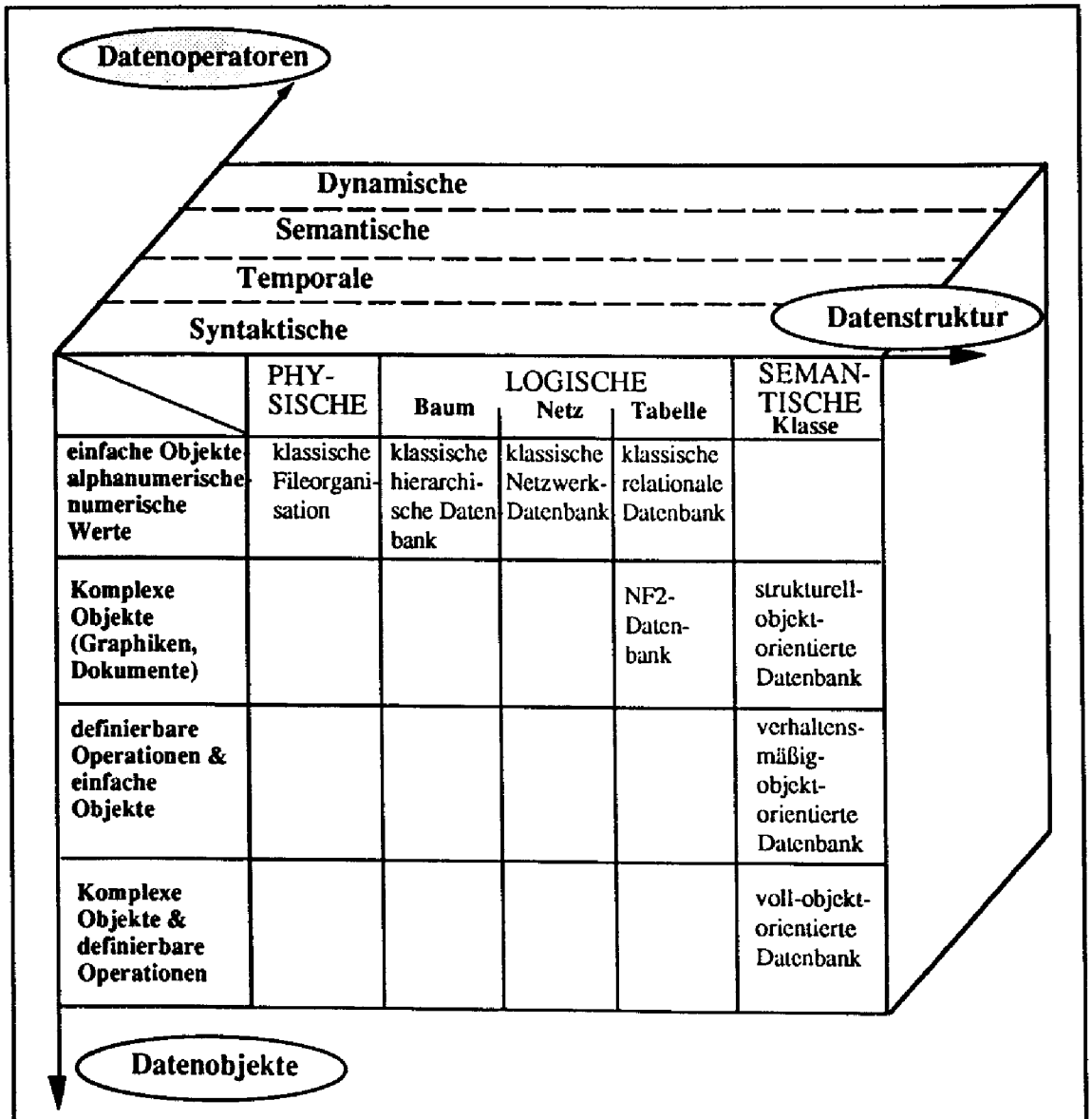
Datenmodelle sollen es ermöglichen, große Datenbestände strukturiert und effizient abzuspeichern und darauf mit möglichst mächtigen Operationen zuzugreifen.

Logische Datenmodelle unterscheiden sich darin, welche Datenobjekte erlaubt sind und welche logischen Datenstrukturen zur Organisation verwendet werden. Die Datenoperatoren als drittes Element kennzeichnen die zulässigen Aktivitäten auf den Datenbestand. Datenmodelle sollen im folgenden anhand dieser drei Merkmale unterschieden werden:

Merkmal 1: **Datenstruktur**
- beschreibt die strukturellen Zusammenhänge der Datenobjekte

Merkmal 2: **Datenobjekt**
- beschreibt die syntaktische Struktur des einzelnen Datenobjektes

Merkmal 3: **Datenoperator**
- beschreibt die definierten Operatoren innerhalb des Datenmodells



	PHY-SISCHE	LOGISCHE Baum	LOGISCHE Netz	LOGISCHE Tabelle	SEMAN-TISCHE Klasse
einfache Objekte alphanumerische numerische Werte	klassische Fileorganisation	klassische hierarchische Datenbank	klassische Netzwerk-Datenbank	klassische relationale Datenbank	
Komplexe Objekte (Graphiken, Dokumente)				NF2-Datenbank	strukturell-objekt-orientierte Datenbank
definierbare Operationen & einfache Objekte					verhaltens-mäßig-objekt-orientierte Datenbank
Komplexe Objekte & definierbare Operationen					voll-objekt-orientierte Datenbank

Abb. 4.3.1/1: Datenmodelle in den Dimensionen Datenobjekte - Datenstrukturen - Datenoperatoren

4.3.1.1. Merkmal Datenstruktur

Datenstrukturmodelle lassen sich in Generationen einteilen. Mit den Modellen der 0. Generation wurde das Ziel verfolgt, für definierte Anwendungsprogramme die Daten speicherplatz- und zugriffseffizient zu halten. Man spricht von primitiven oder physischen Datenmodellen.

Mit Modellen der 1. (Datenbank-)Generation wurde demgegenüber eine programmunabhängige Datenspeicherung durch separate Datenverwaltungssysteme angestrebt. Man spricht von klassischen oder logischen Datenbankmodellen. Aktuell werden Modelle der 2. Generation (sogenannte semantische Datenmodelle) diskutiert, die den inhaltlichen Gehalt der Daten bei der Speicherung nutzen sollen.

	Physische Datenstrukturmodelle	**Logische Datenstrukturmodelle**	**Semantische Datenstrukturmodelle**
Beispiele	ISAM - Dateien VSAM - Dateien	Hierarchisches Datenmodell Netzwerk Datenmodell Relationales Datenmodell	Objekt - Beziehungs-Datenmodell Klassen - Datenmodell
Anliegen	- Physische Effizienz	logische Effizienz (= Vermeidung von Redundanz) Physische Effizienz	Logische Effektivität (= Semantische Ausdrucksmächtigkeit)
Kennzeichen	- Keine Operatoren, d. h. Operatoren, sind in Programmen und Betriebssystemen verankert	- Strukturierung von Zeichenmengen - Flache Datenstrukturen (Satzorientierung) - Standardisierte Operatoren	- Strukturierung von Satzmengen - Tiefe Datenstrukturen (Klassenorientierung)
Bewertung	- Programmabhängige Datenstrukturen	- Strukturierung erfolgt weitgehend unabhängig vom Bedeutungsgehalt (-> Informationen gehen verloren) - Komplexe Operationen (Join) setzen aus Datenmengen wieder Informationen zusammen - Konsistenz von Informationen (logische Integrität) muß in Anwendungsprogrammen gesichert werden	+ Semantischer Bezugsrahmen (z. B. Klassenkonzept) zur Datenstrukturierung + Konsistenz von Informationen wird durch logische Integritätsbedingungen gesichert

Abb. 4.3.1/2: Vergleich physischer, logischer und semantischer Datenstrukturmodelle

Differenziert man etwas genauer, so können folgende Typen unterschieden werden:
(vgl. Schönthaler, Zehnder (1989), S.105 ff.)

Primitive Datei-Strukturmodelle sind physisch orientiert, d. h. sie sind abhängig von den physischen Speicherstrukturen und den logischen Anforderungen der Programme. Sie verfügen über keine eigenständigen Operatoren, d. h. Operatoren sind in Programmen und Betriebssystemen verankert. Die Datenstrukturen umfassen Sätze/Segmente/Speicherplätze. Beispiele sind die ISAM oder VSAM-Dateistrukturen.

Klassische Daten-Strukturmodelle sind syntaktisch-orientiert, d. h. sie strukturieren Zeichenmengen und betrachten nicht deren Bedeutung. Eigenschaften sind

* Daten- und Programmunabhängigkeit, d. h. es existieren eigene Datenverwaltungsprogramme,
* die Datenstrukturierung erfolgt empirisch-intuitiv, d. h. es gibt kein formales Regelwerk zum Aufbau der Datenbestände,
* Satzorientierung, d. h. primitive physische Datenstrukturen (Sätze, Segmente, Speicherplatz), auf die die Operatoren (einfügen, auffinden, lesen, ändern, löschen) angewendet werden,
* Wertorientierung, d. h. die Identität von Datenobjekten wird anhand von numerischen oder alphanumerischen Werten bestimmt.

Komplexe Datenobjekte müssen in mehrere Sätze zerlegt werden (1:n-Beziehung), daraus resultiert eine ineffiziente logische (und physische) Verwaltung, da dadurch der inhaltliche Zusammenhang zwischen den Objekten verlorengehen kann. Beispiele sind das hierarchische Datenmodell und das Netzwerk-Datenmodell.

Erweiterte klassische Datenstrukturmodelle kennen einfache semantische Unterscheidungen (Schlüssel, Nicht-Schlüssel-Attribute). Ziel ist die Unabhängigkeit von Strukturen der Datenspeicherung und der Programme. Eigenschaften sind

* eine logisch-mathematisch fundierte Datenstrukturierung, d. h. es existiert ein formales Regelwerk
* Attributorientierung, d. h. die Operatoren erlauben den direkten Zugriff auf einzelne Attribute
* Wertorientierung, d. h. in flachen Datenobjekten werden numerische oder alphanumerische Werteausprägungen gespeichert.

Beispiele sind das relationale Datenmodell, das auf einer formalen mathematischen Logik basiert, und bestimmte konstruktiv-orientierte Datenmodelle, die ohne formale Logik durch pragmatische Ausnutzung von Dateistrukturen ähnliche Eigenschaften aufweisen (z. B. das System ADABAS der Software AG).

Semantische Datenmodelle wollen einen Ausschnitt der Realität möglichst genau abbilden. Dazu wird angestrebt, die Objekte der Realität, deren Eigenschaften, Beziehungen und Verhalten durch entsprechende Modellierungshilfen (logische Restriktionen, statische Zustände und dynamische Prozesse, Integritätsbedingungen) im Datenmodell einheitlich strukturiert beschreiben zu können. **Entity-Relationsship-Ansätze** verwenden die mit dem Entity-Relationship-Modell (ERM) populär verbreitete Weltsicht von Datenobjekten mit zugehörigen Datenattributen; Beziehungstypen zwischen den zugehörigen Datenobjekten und entsprechenden Beziehungsattributen sowie implizit vorgesehenen statischen Integritätsbedingungen. **Klassenansätze** (in der Literatur häufig auch Objektmodelle genannt; (vgl. Ferstl/Sinz (1990)) betrachten explizit das Verhältnis der Datenobjekte zueinander. Dabei wird jedes Objekt als Instanz einer Klasse (Objekttyp) verstanden. Eine Klasse wird als abstrakter Datentyp verstanden, der durch Variablen (Attribute) und Methoden (Operatoren) beschrieben wird und dabei an andere Klassen Attribute oder Operatoren vererben bzw. selbst erben kann. Weiterhin werden die möglichen Operationen auf jedes Objekt beschrieben und damit die dynamische Integritätsprüfungen integriert.

	Datenmodell	Daten-syntax	Daten-semantik	Daten-struktur	Operatoren
1. Primitive Daten-modelle		Zeichen, Feld, Satz	Keine Differen-zierung		Keine eigen-ständigen Operatoren
2. Klassische Daten-modelle	**2.1. hierarchisch**	Zeichen, Feld, Satz	Keine Differen-zierung	Baum	syntaktische Operatoren
	2.2. Netzwerk			Netz	
3. Erweiterte klassische Daten-modelle	**3.1. relationales Modell**	Attribut, Satz, Relation	Schlüssel-attribute, Nichtschlüs-selattribute	Tabelle	Mengen-operatoren
	3.2. Konstruktiv orientierte Modelle	Attribut, Satz,			
4. Seman-tische Daten-modelle	**4.1. Entity-Relations-ship-Ansätze**	Zeichen, Feld, Satz	Datenobjekt, Objektattribut, Datenbezie-hung, Beziehungs-attribut	Netz	semantische Operatoren
	4.2. Klassen-ansätze	Variable, Methoden	Superklasse, Subklasse	Baum, Netz	

Abb. 4.3.1/3: Klassifikation von Datenmodellen im Überblick

4.3.1.2. Merkmal Datenobjekt

Datenobjekte beschreiben die zulässigen Wertedomänen der Attribute von Objekten, Beziehungen und Ereignissen (vgl. Dadam/Linnemann (1989), Adiba (1987)). Die möglichen Strukturen von Datenobjekten lassen sich in einem morphologischen Kasten klassifizieren:

Datenobjekt-verhalten	passiv	intern aktiv (= definierbare Datenobjekt-operationen)		extern-aktiv (= definierbare Algorithmen)
Datenobjekt-strukturen	einfach (= vordefinierte Strukturen, z. B. Sätze)		komplex (= flexible, nicht vor-definierte Strukturen)	definierbare Operationen
Datensyntax	atomar (= vor-definierter Zeichensatz)		benutzerspezifizierbar (= Definitionsvorschriften zur Beschreibung von Zeichensätzen durch den Benutzer)	
Daten-Zeichensatz	numerisch/ alphanumerisch	erweitert vordefiniert		benutzerspezi-fizierbar
Datenobjekt-zeitbezug	keine Zeitdimension	eine Zeitdimension		mehrere Zeit-dimensionen
Datenobjekt-integrität	keine Überprüfung	statische Integritäts-überprüfung		dynamische Integritätsüber prüfung

Abb. 4.3.1/4: Morphologischer Kasten zu den Merkmalen von Datenobjekten

Die klassischen Datenobjekt-Modelle sind datenorientiert oder auch „satzorientiert". Sie enthalten nur die Möglichkeit, numerische oder alphanumerische Werte abzuspeichern und eignen sich damit für die typischen betriebswirtschaftlichen Anwendungen.

In objektorientierten Datenobjekt-Modellen wird versucht, diese Beschränkungen aufzuheben und sowohl komplexe Objekte als auch bestimmte Operationen im Datenobjekt-Modell abzubilden. Oft wird in diesem Zusammenhang auch von abstrakten Datentypen (Abstract Data Types-ADT) gesprochen.

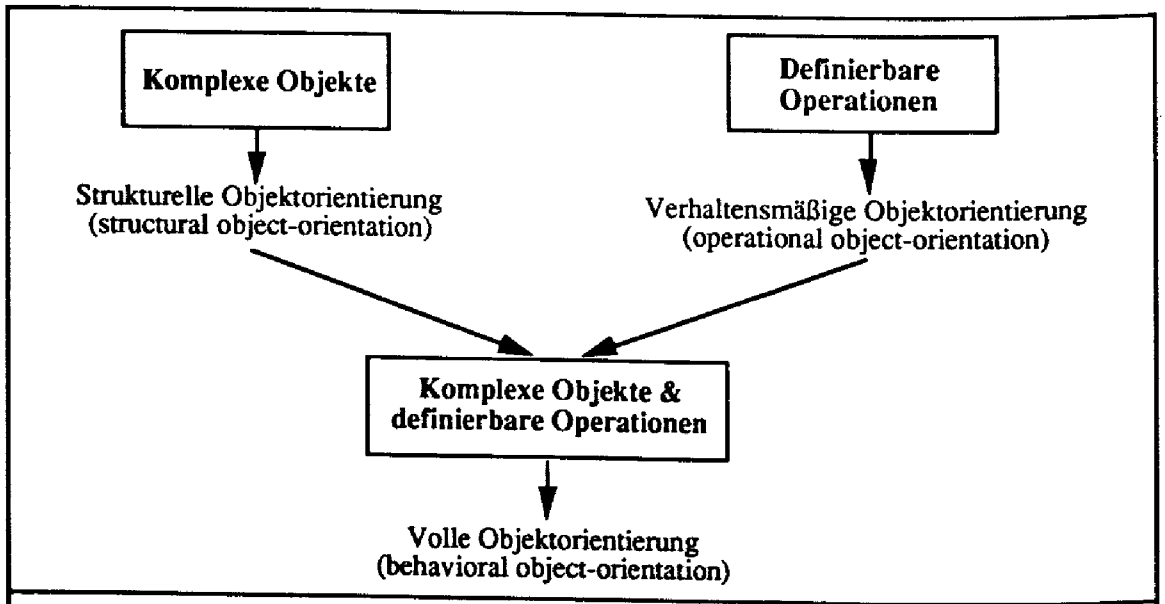


Abb. 4.3.1/5: Arten objektorientierter Datenbanken, (vgl. Dittrich (1986, 1989, 1990))

Objektorientierte Datenobjekt-Modelle werden speziell bei der Abspeicherung

-> von Grafiken/Landkarten etc. (z. B. in CAD-Systemen, bei militärischen Anwendungen)
-> von Dokumenten (z. B. in Bürokommunikationsanwendungen)
-> von logischen Strukturen und Programmen

verwendet. Komplexe Datenobjekt-Modelle werden auch für Multi-Media-Databases benötigt die neben vordefinierten Daten (Grafik, Schrift, Bilder, Töne) vollkommen freie Strukturen verwenden. HYPERTEXT-Systeme sind ein erstes Beispiel für solche Multi-Media-Databases.

4.3.1.3. Merkmal Datenbankoperator

Datenoperatoren beschreiben
- welche Operationen zur Datenmanipulation und zur Recherche in der Datenbank möglich sind,
- wie die Unverletzlichkeit der Datenbasis (Integritätsbedingungen) gesichert wird.

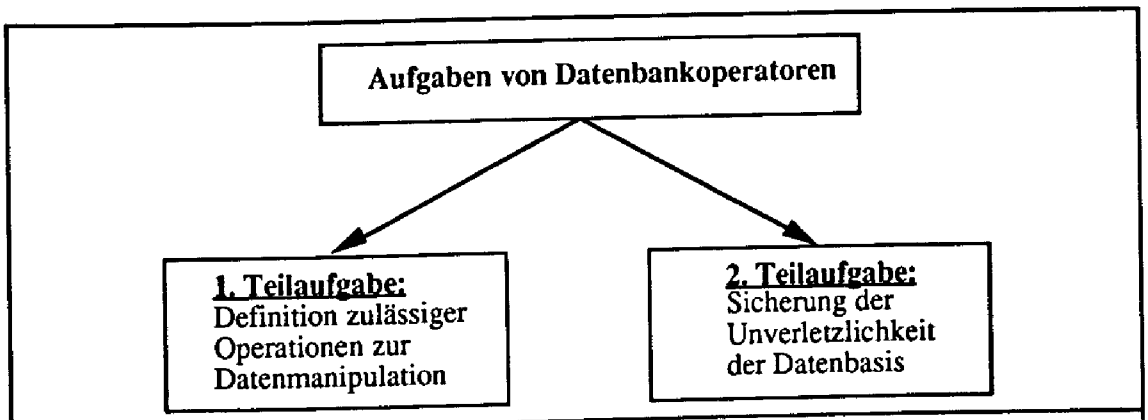


Abb. 4.3.1/6: Aufgaben von Datenbankoperatoren

Klassische Datenbankoperatoren sind syntaktisch orientiert und umfassen die üblichen Operationen bezogen auf Zeichenketten

- zum Einfügen neuer Daten (INSERT-Operator)
- zum Löschen von Daten (DELETE-Operator)
- zum Ändern vorhandener Daten (UPDATE-Operator)
- zum Auffinden vorhandener Daten (RETRIEVE-Operator)

Diese Grundoperatoren bilden in den üblichen Datenbanksystemen die sogenannte „Datenmanipulationssprache" (DML).

Die zweite Aufgabe von Datenbankoperatoren liegt darin, Programm- und Eingabefelder zu verhindern, die die Datenbasis zerstören oder logisch inkorrekt gestalten. In der sogenannten „Transaktionsverwaltung" wird versucht, die logische Konsistenz und Integrität des Datenbestandes zu sichern. Dazu müssen für jede Transaktion folgende Punkte sichergestellt sein (vgl. Lockemann/Dittrich (1987), S. 91)):

1. Atomarität: Die Transaktion hat nur als „geschlossenes Ganzes" eine Wirkung nach außen, d. h. bis zu ihrem vollständigen Abschluß hinterläßt sie überhaupt keine Wirkung, nach ihrem Abschluß ist ihre Wirkung sichtbar und verbindlich.
2. Konsistenz: Garantiert die Transaktion isoliert für sich keine logische Konsistenz der Datenbasis, so wird diese durch die Transaktionsverwaltung sichergestellt.

3. Persistenz: Die Wirkung einer Transaktion geht nicht mehr verloren, außer sie wird durch weitere Transaktionen ausdrücklich widerrufen.

Typ der Integrität	Arten	Kennzeichen	Beispiel
Feldintegrität (field integrity)	Wertintegrität (value integrity)	Der Operator prüft, ob bestimmte Attribute korrekte Ausprägungen haben.	Ein Schlüsselfeld muß immer einen Wert definierter Syntax annehmen.
	Mehr-Wert-Integrität (multi value integrity)	Der Operator prüft, ob bestimmte Ausprägungen unterschiedlicher Attribute sich zueinander korrekt verhalten.	Geschlecht: WEIBLICH ↑ Stand: MUTTER
Bedeutungsintegrität (semantic integrity)		Der Operator prüft die inhaltliche Bedeutung von Datenobjekten.	Mitarbeiter müssen ein bestimmtes Mindestalter haben.
Beziehungsintegrität (referential integrity)		Der Operator prüft die Beziehungen zwischen Schlüsselattributen unterschiedlicher Datensätze.	Auftrag kann so lange nicht gelöscht werden, wie noch eine AUFTRAGSPOSITION existiert.

Abb. 4.3.1/7: Formen statischer Integrität

Temporale Datenbankoperatoren beschreiben die Geschichte (Historie) eines Datenbestandes. Sie stellen sicher, daß z. B. trotz Einfüge- und Löschoperationen immer nachvollzogen werden kann, wie sich ein Datensatz im Zeitablauf entwickelt hat. Rollback-Datenbanken halten dazu nebeneinander, jedoch nicht integriert, mehrere zeitliche Versionen eines Datenbestandes. Historische Datenbanken erlauben es demgegenüber, die zeitliche Geschichte eines Datenbestandes stufenlos nachzuvollziehen.

Beispiel:
Temporale Operatoren erlauben es jederzeit, die Wohnorte einer Person im Laufe seines Lebens aus der Datenbank abzufragen (von .... bis....)

Semantische Datenbankoperatoren beschreiben die zulässige inhaltliche Struktur oder das zulässige Verhältnis der Datenobjekte zueinander. Während die objektorientierten Datenbanken die Möglichkeit bieten, bestimmte Integritätsprüfungen zu programmieren, prüfen semantische Datenbankoperatoren die Einhaltung bestimmter Integritätsbedingungen „quasi automatisch". Semantische Datenbanken müssen infolgedessen Abfragesprachen anbieten, die die semantische Struktur der Datenbank dem Nutzer deutlich machen.

Beispiel:
Der gleichzeitige Eintrag des Vornamens "Norbert" und des Geschlechts "weiblich" ist nicht zulässig.

Zu unterscheiden sind in der aktuellen Diskussion zwei Konzepte: Klassenkonzepte verwenden Klassenhierarchien mit Vererbungsstrukturen für Datenstrukturen und -operatoren. Objektkonzepte betrachten semantische Netzstrukturen, in denen Datenobjekte über semantische Beziehungen definierte Nachrichtentypen austauschen.

Dynamische Datenbankoperatoren betrachten die (zulässige) Entwicklung der Datenobjekte im Zeitablauf (dynamische Integritätsbedingungen). Aktive Datenbankoperatoren führen beim Zugriff auf bestimmte Felder oder Prozeduren automatisch Folgeprozeduren

durch (inferencing procedures), die nicht nur die syntaktische, sondern auch die semantische Integrität der Datenbank im Zeitablauf sichern.

Beispiel:
Das Attribut Familienstand kann nur eine bestimmte zeitliche Folge (ledig, verheiratet, geschieden etc.) durchlaufen.

Operatortyp	Beschreibung	Konzepte
Syntaktische	Operatoren beschreiben die Einfüge- und Lösch-operationen von Zeichenstrukturen	
Temporale	Operatoren beschreiben die zeitliche Entwick-lung (Geschichte) von Zeichenstrukturen	Roll-back-Database Historische Datenbank
Semantische	Operatoren beschreiben die inhaltliche Bedeu-tung von Datenobjekten und deren inhaltliches Verhältnis zu anderen Objekten	Klassenkonzept Entity-Relationsship - Konzept
Dynamische	Operatoren beschreiben die zulässigen Zustände von Datenobjekten und den Prozeß der Objekt-veränderung	Dynamische Integritätsbedingungen

Abb. 4.3.1/8: Gegenüberstellung der Operatortypen

4.3.2. Kennzeichnung ausgewählter Datenmodelle

4.3.2.1. Strukturorientierte Modelle

4.3.2.1.1. Hierarchisches Modell

Die grundlegende Datenstruktur des hierarchischen Datenmodells ist ein (heterogener) Baum zur Darstellung von 1:n-Beziehungen zwischen den Datenelementen. Es existieren Baum-strukturen

* mit einstufiger Hierarchie (ein Vater - mehrere Söhne)
  Beispiel:
  Prüfung-Prüflinge
* mit mehrstufiger Hierarchie (ein Großvater - mehrere Väter - mehrere Söhne je Vater).
  Beispiel:
  Uni-Fachbereich-Prüfung-Prüflinge

Die Datenobjekte (Entity-Typen) im Baum werden auch Knoten genannt. Das oberste Datenelement in der Hierarchie wird als Wurzelknoten (root) bezeichnet; jeder weitere Knoten (children) hat nur einen Eltern-Knoten (parent).

Beim hierarchischen Modell ist es sinnvoll, zwischen physischen und logischen Bäumen zu unterscheiden. Physisch besteht eine hierarchische Datenbank aus einer Kollektion von Bäumen, die aus Segmenten gebildet sind. Jeder Baum hat auf den Datenspeichereinheiten ein bestimmtes physikalisches Format, dessen Struktur durch die Zugriffsmethoden der Datenbanksprache definiert wird (vgl. Martin (1988), S. 194 ff.).

Aus der physischen Baumstruktur werden logische Baumstrukturen erzeugt. Dabei werden Zeiger (logical pointer) benutzt, um das gleiche physische Segment nicht in verschiedenen Bäumen doppelt speichern zu müssen und um beliebige logische Strukturen erzeugen zu können. Somit kann die logische Struktur einer hierarchischen Datenbank von jeder der existierenden physischen Baumstrukturen verschieden sein. Logische Zeiger können nicht

nur zwischen physischen Segmenten bestehen, sondern auch zwischen korrespondierenden Elementen verschiedener Datenbanken. Weiterhin können die logischen Verknüpfungen auch sogenannte „Schnittpunktdaten" (intersection data) umfassen, d. h. es sind Daten, die Beziehungen zwischen Segmenten beschreiben (z. B. „Noten" in der logischen Beziehungen zwischen „Studenten" und „Vorlesung").

Bei natürlichen Hierarchien wie Stücklisten (Artikel - Baugruppen - Teile), Schlagwort-Katalogen (Sachgruppen - Schlagworte - Titel), Personaldateien (Firma, Abteilung, Gruppe, Mitarbeiter) oder Melderegistern (Gemeinde, Familie, Kinder) sind hierarchische Datenmodelle normalerweise sehr effizient, da die Abfragen dieser natürlichen Struktur folgen. Anders bei künstlichen Hierarchien, die meist an den dominierenden Abfragestrukturen orientiert werden. Dabei richtet man sich nach den häufigsten Zugriffspfaden, der effizienten Speicherplatznutzung und Datenreorganisation. Zwar ist es bei den existierenden hierarchischen Systemen möglich, fast beliebige logische Strukturen zu realisieren, doch führen Unterschiede zwischen logischer und physischer Baumstruktur zu weniger effizienten Lösungen.

Modellierung mit dem hierarchischen Datenbankschema

Wurde im Zuge der Konstruktion das Entity-Relationsship-Modell verwendet, so ist dieses wie folgt in das hierarchische Datenmodell zu überführen:

- Je "Entity-Typ" ist ein Knoten (= segment) zu bilden,
- je "Relation" ohne Attribute ist ein Verweis und
- je "Relation" mit Attributen ein Knoten oder ein Schnittpunktdatum zu definieren.

In der Folge soll ein einfaches Beispiel zur Illustration der Datenmodelle genutzt werden:

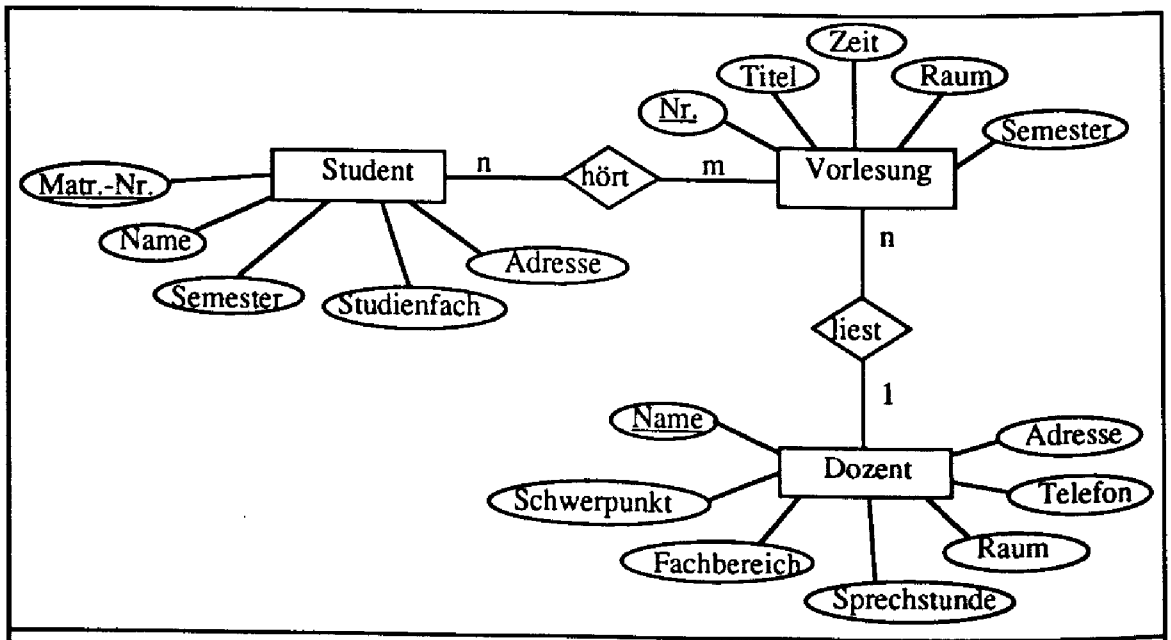


Abb. 4.3.2/1: Entity-Relationship-Modell für nachfolgende Beispiele

Es werden zunächst drei physikalische Bäume gebildet, um das Beispiel in dem hierarchischen Datenmodell abzubilden.

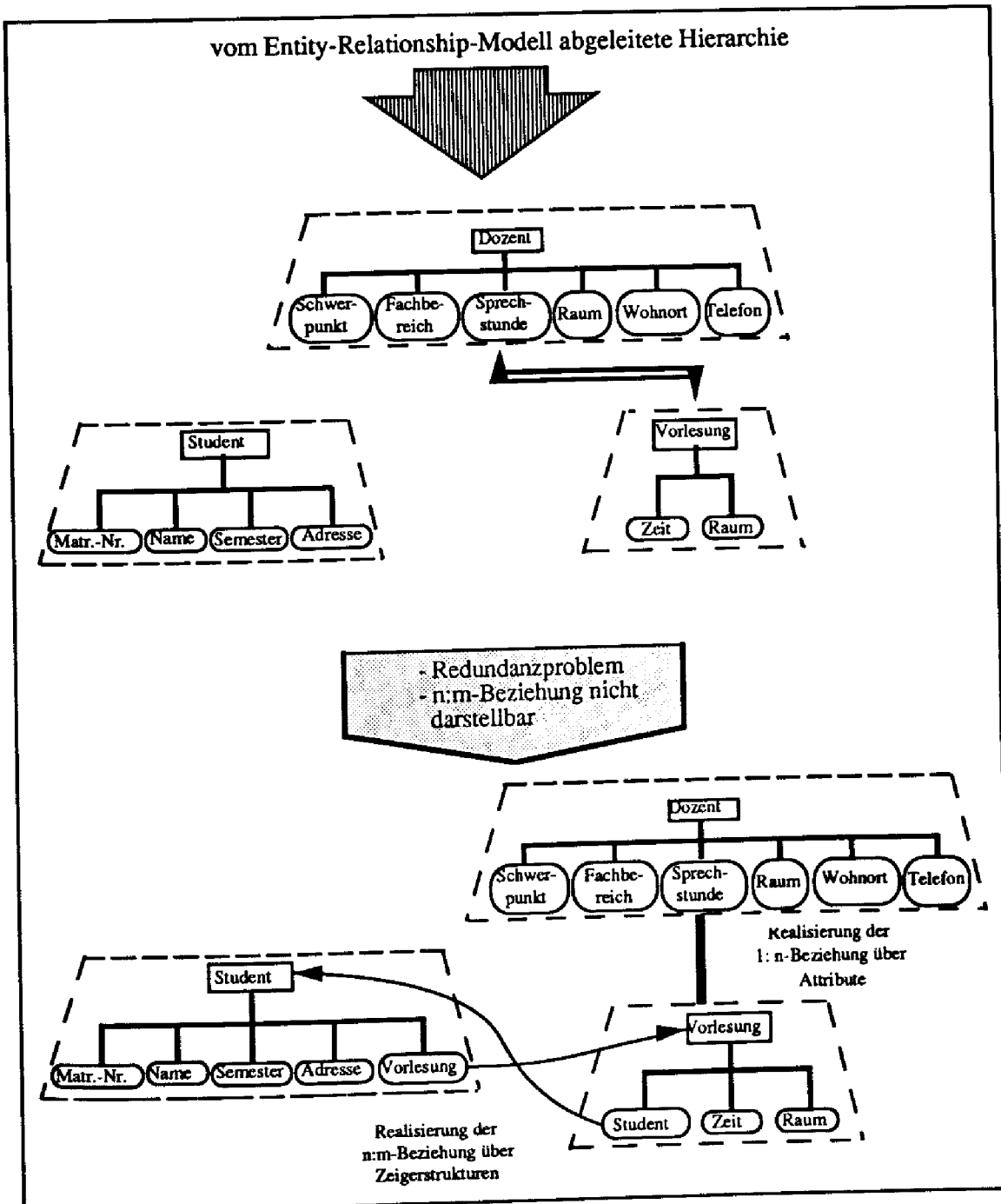


Abb. 4.3.2/2: Hierarchisches Modell abgeleitet vom Entity-Relationship-Modell

Sollen bestimmte Segmente in der Datenbank physisch nicht mehrfach abgespeichert werden (z. B. in jeder Vorlesung die zugehörigen Studenten), so ist es sinnvoll, die „reine Baumstruktur" durch logische Zeiger zu durchbrechen. Dadurch lassen sich n:m-Beziehungen darstellen, und es wird die erwähnte Redundanz vermieden.

Auch im hierarchischen Modell sollten sachlogische Datenstrukturen in DV-effiziente Datenstrukturen überführt werden. Dabei können die weiter oben erläuterten Methoden der Schemabildung (Strukturzerlegung und -synthese) so genutzt werden, daß

- "natürliche Hierarchien" möglichst ausgenutzt werden,
- häufige Zugriffsoperationen unterstützt werden und
- daß möglichst wenige Datenduplizierungen notwendig sind.

Operationen im hierarchischen Datenmodell beziehen sich auf

- den direkten Zugriff auf ein Wurzel-Entity über dessen Primärschlüsselwert,
- den Zugriff auf ein Entity ensprechend der hierarchischen (physischen) Ordnung auf die Kinder-Entities
- den Zugriff über logische Zeiger auf verknüpfte Entities.

Normalerweise erlauben die Operatoren auch die prozedurale Suche nach bestimmten Attributausprägungen in einem physischen oder logischen Baum.

Vorteile:

Das hierarchische Datenbankmodell kann einfache physische Speicherstrukturen (z. B. sequentielle Dateien) verwenden und bietet daher große verarbeitungstechnische Vorteile. Besonders effizient ist es bei Vorliegen natürlicher Hierarchien und eindimensionaler Abfragen (z. B. Stücklisten).

Vorteile		Nachteile	
DV-Effizienz	- einfache sequentielle Speicherstrukturen verwendbar	- Änderungen relativ aufwendig	**Flexibilität**
Anwendungseffizienz	- gut geeignet für natürliche Hierarchien und rekursive Strukturen	- Daten werden doppelt gespeichert, außer es werden logische Zeiger verwendet	**Redundanz**
		- Nutzer muß die Zugriffspfade kennen	**Zugriffsoperationen**
		- Bäume realistischer Größe sind unübersichtlich	**Transparenz**

Abb. 4.3.2/3: Vor-/ Nachteile des hierarchischen Modells

Nachteile:

Änderungen sind im hierarchischen Modell relativ schwierig. Wurde die hierarchische Datenbank mit den üblichen sequentiellen Strukturen realisiert, so bedingt die Einführung von Segmenten oder Attributen entweder die Neuorganisation der Datei oder aber die Verwendung logischer Zeiger (für die physisch ebenfalls Platz vorgesehen werden muß).

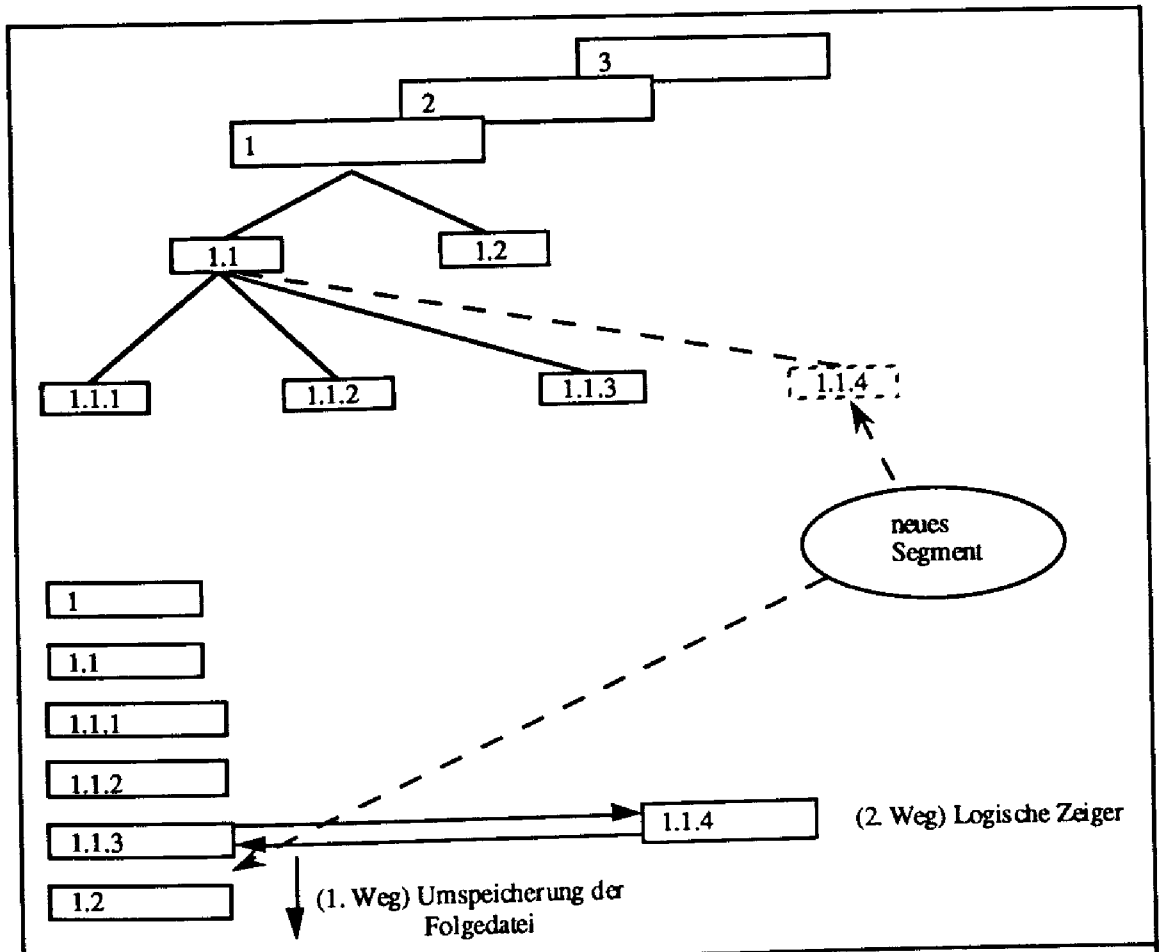


Abb. 4.3.2/4: Sequentielle Speicherung einer Baumstruktur und Einfügen eines neuen Segments durch (1) Umspeicherung der Folgedatei oder (2) logische Zeiger

Die folgende Tabelle zeigt die Auswirkungen von Änderungen im hierarchischen Modell auf:

Änderungstyp	Maßnahmen	Beispiel
Wertänderung	keine Strukturänderung	Adreßänderung
zusätzliches Entity (Insert)	Einfügung eines Datensatzes mit einer Beziehung	Geburt eines Kindes
Änderung einer Zuordnung	Umstellung (Löschen eines Adreßverweises u. Einfügen eines neuen Adreßverweises)	Adoption eines Kindes
Zusätzlicher Entity-Typ	Einfügen aller entsprechenden Datensätze	Fahrzeuge der Kinder mit je einer Beziehung

Abb. 4.3.2/5: Änderungen im hierarchischen Datenmodell

4.3.2.1.2. Netzwerk-Modell

Wenn ein Datenobjekt (Entity-Typ) mehr als ein übergeordnetes Datenelement hat, kann diese Beziehung ohne Redundanz nicht mit einem hierarchischen Modell beschrieben werden. Formal entsteht diese Struktur dadurch, daß ein Strukturelement (Datensatz, Entitität) gleichzeitig zu mehreren Hierarchiegruppen gehören kann (vgl. Vetter (1987)).

Bei den Netzstrukturen (teilweise auch Plexstrukturen genannt, vgl. Martin (1988)) sind zwei Fälle zu unterscheiden: Bei einfachen Netzstrukturen ist die Eltern-Kind-Beziehung komplex (d. h. die Eltern können mehrere Kinder haben), doch die Kind-Eltern-Beziehung ist einfach (d. h. die Kinder haben jeweils nur eine Mutter bzw. einen Vater). Bei komplexen Netzstrukturen sind beide Beziehungen komplex.

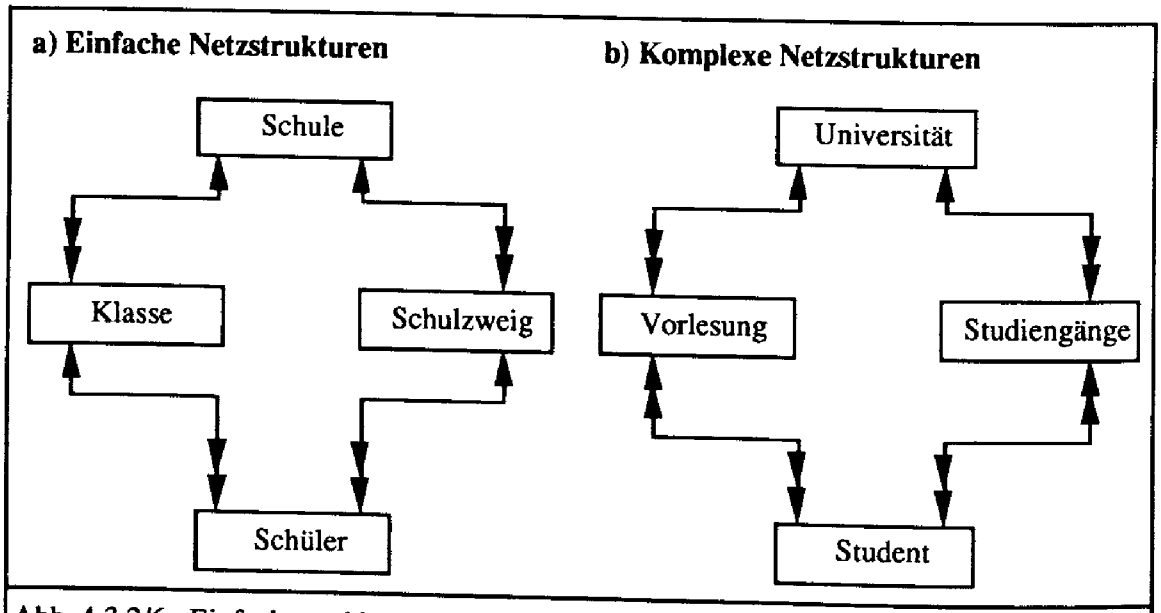
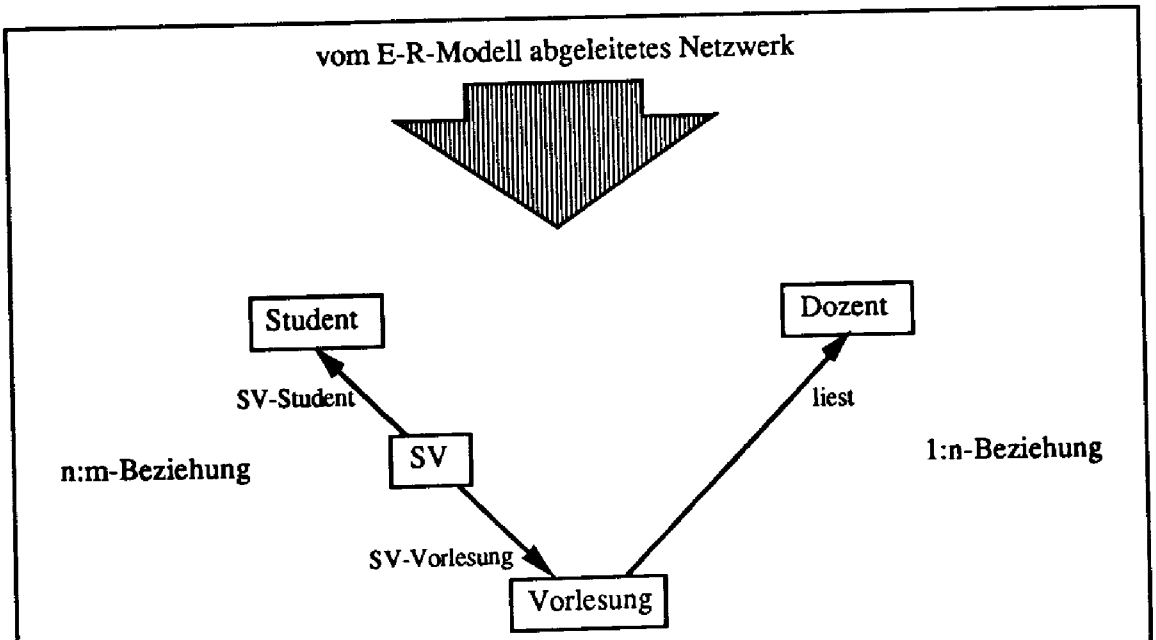


Abb. 4.3.2/6: Einfache und komplexe Netzstrukturen (vgl. Martin (1988))

In der Schule gehört jeder Schüler zu einem Schulzweig und zu einer Klasse und diese ist wiederum eindeutig einem Schulzweig zugewiesen (einfache Struktur), während in einer Universität ein Student an vielen Vorlesungen teilnimmt und auch zu mehreren Studiengängen gehören kann. Eine Vorlesung kann ebenfalls für mehrere Studiengänge verbindlich sein (komplexe Struktur).

In Netzwerk-Modellen werden die Beziehungen zwischen einzelnen Datensätzen (Individualbetrachtung), nicht die Beziehungen zwischen gleichartigen Mengen abgebildet. Das allgemeine Netzwerk-Modell mit einfachen und komplexen Beziehungen ist speicherungstechnisch schwer zu realisieren. Daher werden in den verbreiteten Datenbanksystemen komplexe n:m-Beziehungen in einfache Beziehungen zerlegt. Bei der Beschreibung des Netzwerkmodells orientieren wir uns an dem Vorschlag der CODASYL-Arbeitsgruppe (Conference on Data Languages) aus dem Jahre 1973.

In diesem speziellen Netzwerkmodell gibt es nur spezielle Beziehungstypen, nämlich 1:n-Beziehungen. Vernetzte Strukturen aus m:n-Beziehungen werden bei der Schemabildung zu "künstlichen 1:n-Beziehungen durch Zwischenschaltung eines Entity-Typs" zerlegt - hier "SV". Es entstehen künstliche Entities mit dem Attribut "SV-ID" (Student-Vorlesung-Identity).



Logische Record-Typen:
Student (Matr.-Nr., Name, Semester, Studienfach, Adresse)
Dozent (Name, Schwerpunkt, Fachbereich, Sprechstunde, Raum, Adresse)
Vorlesung (Nr., Titel, Zeit, Raum, Semester)

Set-Typen:
SV-Student: SV -> Student
SV-Vorlesung: SV -> Vorlesung
liest: Vorlesung -> Dozent

Abb. 4.3.2/7:Netzwerkmodell mit seinen logischen Record-Typen und Set-Typen (Notation nach Schlageter/Stucky (1983))

Das Netzwerk-Modell kennt folgende Strukturelemente:

RECORD Typen

Entity-Typen (= Datenobjekte mit gleichen Attributen) werden im Netzwerkmodell als Record-Typ und einzelne Entities dementsprechend als Records bezeichnet. Die Records bestehen aus Items (Attributen) und Item-Wertebereichen (Attribut-Domänen). Zu unterscheiden sind natürliche Record-Typen und künstliche Record-Typen, die zur Aufspaltung von m:n - Beziehungen in 1:n - Beziehungen benötigt werden. Einer der Sätze bildet den sogenannten Ankersatz (owner record); die übrigen Sätze sind logisch dem Ankersatz zugeordnet und heißen „Gliedsätze"(membership records). Jeder Satz ist entweder über seinen Typ oder über die Sets, denen er angehört, zugänglich.

SET-Typen

Die 1:n-Beziehungen werden als Set-Beziehungen, der Beziehungstyp als Set-Typ bezeichnet. Jeder Set-Typ verbindet zwei bestimmte Record-Typen; er hat einen Owner- und einen Member-Record-Typ. Ein Set hat genau einen Owner-Record ("Vater") und mehrere Member-Records ("Söhne"), die (im CODASYL-Ansatz) über automatische Member Record-Set-Beziehungen die Beziehungsintegrität sicherstellen können sowie zwingende

von optionalen Beziehungen unterscheiden. Member-Records dürfen auch fehlen (empty set).

Modellierung mit Netzwerkmodell

Ergibt sich aus der Konstruktion ein ERM, so ist dieses nach folgenden Regeln zu überführen (vgl. Mayr/ Dittrich/Lockemann (1990), S. 521 f):

Regel 1: Zerlege die Attributtypen des semantischen Datenmodells so, daß sie sich 1:1 in den Attributtypen eines logischen Netzwerkmodells abbilden lassen.

Regel 2: Ergänze die Attributtypen jedes Gegenstandsobjektes des semantischen Datenmodells um einen eindeutig identifizierbaren Attributtyp (Schlüssel).

Regel 3: Für jedes Gegenstandsobjekt des semantischen Datenmodells wird ein Satztyp (record) definiert, der das Schlüsselattribut und alle weiteren Attribute des Datenobjekts aufnimmt.

Regel 4: Für jeden Beziehungstyp des semantischen Datenmodells ohne Attribute wird ein Bindungstyp (set) definiert, der einen der beiden Gegenstandsobjekte als Ankertyp und den anderen als Gliedsatz (member record) einrichtet.

Regel 5: Für jeden Beziehungstyp des semantischen Datenmodells, der Attribute umfaßt, wird ein Satztyp (Record-Typ) definiert, der die Attribute der Beziehung aufnimmt. Dieser Record-Typ wird über Relationen-Typen (Set-Types) mit den zu assoziierten Objekten (= records) verbunden.

Operationen im Netzwerkmodell beschreiben den logischen Weg vom Einstiegspunkt über eine Folge von Sets zum gewünschten Record. Der Benutzer spezifiziert somit den Zugriffspfad für jeden Satz; er "navigiert" in der Datenbank und vergleicht dabei seine Suchkriterien sequentiell mit den Attributen der Sätze.

Vorteile:

Das Netzwerkmodell erlaubt die Realisierung von vermaschten und rekursiven Strukturen und Mehrfachbeziehungen. Im Codasyl-Ansatz sind zudem eine Reihen von Prüfbedingungen realisierbar, die die Integrität der Datenbestände sichern helfen. Der Zugriff kann im Netzwerkmodell sowohl über Ankersätze (owner records) als auch über Memberbeziehungen sowie direkt über Schlüssel erfolgen. Der Benutzer muß dazu den Zugriffspfad als Sequenz von Sets anlegen (Navigieren durch die Datenbank).

Vorteile		Nachteile	
Anwendungs-effizienz	- hohe Ausdrucksmächtigkeit für semantische Erfordernisse	- Struktur ist durch logische Zeiger statisch; Änderungen bei diesen Zeigern einfach	**Flexibilität**
Redundanz-freiheit	- Logische Zeiger vermeiden Redundanz	- Verkettete Speicherstrukturen - Künstliche Sets beanspruchen Speicherplatz	**DV-Effizienz**
		- Nutzer muß die Zugriffspfade kennen	**Zugriffs-operationen**
		- Netze realistischer Größe sind unübersichtlich	**Transparenz**

Abb. 4.3.2/8: Vor-/ Nachteile des Netzwerk-Modells

Nachteile:

In den üblichen Systemen müssen die komplexen n:m-Netzbeziehungen durch künstliche Entities auf hierarchische Beziehungen 1:n reduziert werden. Das Netzmodell kann nur einseitige Existenzabhängigkeiten abbilden, Beziehungen mit wechselseitigen Existenzabhängigkeiten, ohne Existenzabhängigkeiten, mit mehreren Ownern oder mit begrenzter Anzahl von Membern sind nicht darstellbar.

Änderungstyp	Maßnahme	Beispiel
Wertänderung	keine strukturelle Änderung	Familiennamensänderung
Zusätzliches Entity	Einfügen in alle Beziehungen	Zusätzliches Kind
Änderung der Zuordnung	Verknüpfung ändern	Adoption
Zusätzlicher Entity-Typ	Einfügen aller entsprechenden Daten	

Abb. 4.3.2/9: Änderungen im Netzwerkmodell

4.3.2.1.3. Relationales Modell

Die Besonderheiten des relationalen Datenmodells liegen darin, daß alle nach der Konstruktion für den Benutzer relevanten (statischen) Informationen ausschließlich durch Datenwerte ausgedrückt werden; Verweise über Indextabellen oder Verkettungen sind nicht notwendig; sie werden durch identifizierende Attributkombinationen ("Schlüssel") ersetzt. Damit ist die Verarbeitung der Daten unabhängig von deren Speicherung.

Das relationale Modell basiert auf Tabellen, die sowohl "Entities" als auch "Relations" in Tabellenzeilen und deren Attribute in Tabellenspalten darstellen. Die Tabellen (auch Relationen genannt) besitzen eine feste Anzahl von Spalten (= Attributen) und einer variablen Zahl von Sätzen (= Tupel).

Beispiel: Student	*Matrikelnummer*	*Name*	*Vorname*	*Alter*
	4711	*Koch*	*Uwe*	*28*
	4712	*Dreyer*	*Holger*	*29*
	4713	*Werner*	*Andreas*	*31*
	usw.			

Das relationale Schema "Student" besteht aus vier Spalten, die semantisch als die Attribute "Matrikelnummer", "Name", "Vorname" und "Alter" interpretiert werden. Zulässige Wertebereiche für die Attribute ergeben sich aus dem Nummerkreis der Matrikelnummer, zulässigen Namen und einem sinnvollen Altersbereich (etwa 18 bis 85 Jahre). Die Matrikelnummer bildet den (künstlichen) Schlüssel dieser Relation.

Man sagt,
- die Anzahl der Attribute bestimmt den "Grad einer Relation",
- die Anzahl der Sätze bestimmt die "Kardinalität einer Relation",
- die Wertebereiche eines Attributs bestimmen dessen "Domäne".

Die Beziehungen zwischen Relationen werden mittels der Schlüssel hergestellt. Dabei ist es nicht erlaubt, daß ein Attribut eines Satzes mehrere Werte annimmt, in einem solchem Fall müssen dann mehrere Sätze gebildet werden.

Beispiel: Veranstaltungsbesuch	*Veranstaltungsnummer*	*Matrikelnummer*
	1	*4711*
	1	*4712*
	1	*4713*
	2	*4712*
	2	*4713*
	usw.	*usw.*

Die Veranstaltungen werden von mehreren Studenten besucht, infolgedessen existieren für jede Veranstaltungsnummer mehrere Sätze.

Ein Schlüssel übernimmt in jeder Relation die Funktion des "Primärschlüssels", d. h. er bildet die Untermenge an Attributen, die zur eindeutigen Identifizierung eines Satzes ausreicht und verwendet wird. Fremdschlüssel sind Primärschlüssel anderer Relationen, die als Attribut aufgenommen werden, um Beziehungen zwischen Relationen herzustellen.

Mittels sogenannter Integritätsregeln wird die logische Zulässigkeit des Datenmodells gesichert. Entitätsintegrität bedeutet, daß jedem Satz ein bekannter Primärschlüssel zugeordnet werden muß. Referentielle Integrität bedeutet, daß bei der Zuweisung eines Satzes mit einem Fremdschlüssel darauf geachtet wird, daß dieser in der zugehörigen Relation als Primärschlüssel existiert.

Art	Bedeutung
Entitätsintegrität	Es kann kein Datenobjekt ohne Schlüssel eingetragen werden.
Domänenintegrität	Gleichartige Attribute in allen Attributen haben die gleiche Werteintegrität
Referentielle Integrität	Wird ein Datenobjekt mit einem Attribut eingetragen, das in einer anderen Relation Primärschlüssel ist, so muß dessen Existenz geprüft werden.

Abb. 4.3.2/10: Integrität in einer relationalen Datenbank

Beispiel:
In der Relation "Veranstaltungsbesuch" ist Veranstaltungsnummer der Primärschlüssel und "Matrikelnummer" der Fremdschlüssel. Würde eine unbekannte Matrikelnummer eingetragen werden, so würde die referentielle Integrität des Datenmodells verletzt werden.

Auf dem Markt werden viele Datenbanksysteme angeboten, denen die Hersteller aus Marketing-Gründen das Merkmal „relational" verleihen. Um den Begriff „relationales Datenbanksystem" eindeutig abzugrenzen, hat Codd die folgenden zwölf Regeln aufgestellt (Codd (1985), Ziekursch (1989)).

Regel 0: „Ein relationales Datenbankmanagementsystem (RDBMS) muß Datenbanken allein durch seine relationalen Fähigkeiten vollständig verwalten können"; das heißt Bearbeitung aller Daten mit mengenorientierten Operationen (GRUNDREGEL).

Regel 1: „Die gesamte Information in einer relationalen Datenbank wird explizit auf der logischen Ebene und genau auf eine Weise dargestellt durch Werte in Tabellen" (INFORMATIONSREGEL), das heißt eine einheitliche logische Sichtweise unabhängig von der physikalischen Speicherung.

Regel 2: „Auf jeden einzelnen (atomaren) Wert in einer relationalen Datenbank kann auf der logischen Ebene mit einer Kombination aus Tabellenname, Primärschlüsselwert und Spaltenname garantiert zugegriffen werden." (GARANTIERTER ZUGRIFF), das heißt es kann assoziativ über den Dateninhalt selektiert werden.

Regel 3: „Nullwerte sind weder leere Zeichenketten oder Ketten von Leerzeichen, noch Null oder irgendeine andere Zahl. Nullwerte werden in einer vollständig relationalen RDBMS systematisch und datentypunabhängig zur Darstellung unbekannter und nicht zutreffender Informationen unterstützt (SYSTEMATISCHE BEHANDLUNG VON NULLWERTEN).

Regel 4: „Die Datenbankbeschreibung wird auf der logischen Ebene auf die gleiche Weise wie gewöhnliche Daten dargestellt, so daß autorisierte Benutzer sie mit derselben relationalen Sprache abfragen können, die sie auch auf reguläre Daten anwenden", d. h., die Meta-Daten im Data Dictionary sind ebenfalls entsprechend der GRUNDREGEL und der Informationsregel zu behandeln. (DYNAMISCHER UND AKTIVER KATALOG IM RELATIONALEN MODELL).

Regel 5: „Ein relationales Modell darf mehrere Sprachen und verschiedene Terminal-Modi (z. B. Formular-Ausfüll-Modus) unterstützen. Es muß jedoch mindestens eine Sprache geben, deren Anweisungen einer wohldefinierten Syntax entsprechend, als Zeichenketten ausgedrückt werden können. Sie muß die folgenden Punkte berücksichtigen

- Datendefinition (CREATE TABLE)
- Datensichtdefinitionen (CREATE VIEW)
- Datenmanipulationen (INSERT, SELECT, UPDATE, DELETE)
- Integritätsbedingungen (CREATE INTEGRITY)
- Authorisierung (GRANT)
- Transaktionsgrenzen (BEGIN, COMMIT, ROLLBACK) (UMFASSENDE DATEN-SUBSPRACHE)

Regel 6: „Alle Datensichten (Views), die theoretisch Datenänderungen (Update) zulassen, können auch durch das System geändert werden", d. h. nicht eindeutige Datensichten mit unerwünschten Nebeneffekten werden ausgeschlossen (DATENÄNDERUNGEN durch DATENSICHTEN).

Regel 7: „Die Fähigkeit, eine Basisrelation oder eine abgeleitete Relation oder einen Operanden zu handhaben, ist nicht nur zum Wiederauffinden von Daten, sondern auch zum Einfügen, Ändern und Löschen von Daten anwendbar", d. h. umfassende mengenorientierte Operationen (MENGENORIENTIERTES EINFÜGEN, ÄNDERN, LÖSCHEN).

Regel 8: „Anwendungsprogramme und Terminalaktivitäten bleiben zu jedem Zeitpunkt logisch unbeeinträchtigt, wenn Veränderungen an den Speicherstrukturen oder an den Zugriffsmethoden vorgenommen werden," d. h. vollkommene Unabhängigkeit des logischen „Was" vom logischen „Wie" (PHYSISCHE DATENUNABHÄNGIGKEIT).

Regel 9: „Anwendungsprogramme und Terminalaktivitäten bleiben logisch unbeeinträchtigt von jeglichen informationserhaltenden Veränderungen, die theoretisch ohne Beeinträchtigungen an den Basistabellen vorgenommen werden können", d. h. vollkommene Unabhängigkeit der Anwendungsprogramme von eventuellen Designänderungen und von dem darunterliegenden logischen Datenmodell (LOGISCHE DATENUNABHÄNGIGKEIT).

Regel 10: „Integritätsbedingungen, die für bestimmte relationale Datenbanken spezifisch sind, können in der relationalen Daten-Subsprache definiert und im Katalog, nicht in den Anwendungsprogrammen, gespeichert werden, d. h. kein Anwender muß Integritätsbedingungen in seinen Programmen berücksichtigen und er kann sie auch nicht umgehen. (INTEGRITÄTSUNABHÄNGIGKEIT).

Regel 11: „Ein relationales RDBMS besitzt Verteilungsunabhängigkeit" (VERTEILUNGSUNABHÄNGIGKEIT).

Regel 12: „Falls ein relationales System über eine niedere (satzorientierte) Sprache verfügt, kann diese nicht benutzt werden, um die Integritätsregeln und Bedingungen zu verletzen oder zu umgehen, die in der höheren (mengenorientierten) Sprache formuliert worden sind. (UNTERWANDERUNGSVERBOT).

Die heute angebotenen „relationalen Datenbanksysteme" realisieren die zwölf Grundanforderungen nur teilweise. Echt - relationale Datenbankmodelle erfordern Assoziativspeicher (= Speicher, die den tabellenartigen Zugriff unterstützen). Unechte - relationale Datenbankmodelle "gaukeln" dem Benutzer die relationale Sicht der Daten vor, indem sie zum Beispiel nur Teilmengen der Daten in den Arbeitsspeicher holen und diese Teilmengen relational im Zugriff haben.

Relational-orientierte Datenbankmodelle verletzen die Grundregeln für relationale Datenbanksysteme, insbesondere die Regel 0, nach der Zugriffe nur über Mengenoperationen zulässig sind. Sie benötigen für Zugriffsoperationen den Suchpfad, und können direkt keine Mengenoperationen unterstützen (Regel 6 wird verletzt).

Beispiel:
dBASE muß mengenorientierte Anfragen (z. B. über SQL) immer in prozedurorientierte Anfragen umsetzen, die aus mehreren Befehlen bestehen.

Modellierung mit einem relationalen Datenmodell

Ein ERM läßt sich nach folgenden Regeln in ein relationales Modell überführen: (vgl. Mayr/Dittrich/Lockemann (1990), S. 518)

Regel 1: Zerlege die Attributtypen des semantischen Datenmodells so, daß die zerlegten Attribute sich 1:1 in den Attributtypen des logischen, relationalen Datenmodells abbilden lassen.

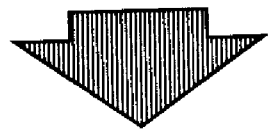
Falls sich bestimmte Attributtypen einer Abbildung in den beschränkten Datentypen der üblichen relationalen Datenbanksysteme entziehen, so sind sie entweder zu streichen oder mit einem Verweis außerhalb der Datenbank zu speichern.

Regel 2: Ergänze die Attributtypen jedes Datenobjektes und jeder Beziehung um eindeutig identifizierende Attribute (Schlüsselattribute).

Regel 3: Fasse Attributtypen eines Datenobjektes oder einer Beziehung des semantischen Datenmodells in einer Relation des logischen Datenmodells zusammen.

Regel 4: Eine Beziehung des semantischen Datenmodells wird so in einer Relation abgebildet, daß jeweils die Schlüsselattribute aller an der Beziehung beteiligten Datentypen und die eigenen Attributtypen der Beziehung in der Relation enthalten sind.

vom E-R-Modell abgeleitete Relationen



Student	Matr.-Nr.	Name	Semester	Studienfach	Adresse
	3617680	Schmidt	11	Wirtschaftswissenschaften	Paderborn
	3255666	Bauer	7	Wirtschaftsingenieur	Paderborn
	3172222	Müller	9	Wirtschaftswissenschaften	Detmold
	3627343	Fischer	5	Informatik	Soest

Vorlesung	Nr.	Schwerpunkt	Zeit	Raum
	05325	Kostenrechnung	Di. 9 - 11	C1
	05328	Datenbanken	Do.14 - 16	H5.324
	05336	Marketing	Mi. 9 - 11	H5.324

Dozent	Name	Schwerpunkt	Fachbereich	Sprechstunde	Raum	Telefon	Adresse
	Meier	Kostenrechnung	5	Fr. 9 - 11	C4.224	2303	Paderborn
	Meier	Buchführung	5	Fr. 9 - 11	C4.224	2303	Paderborn
	Rubens	Anwendungssysteme	5	Do. 10 - 11	H5.208	3814	Bielefeld
	Schneider	Datenbanken	5	Do. 14 - 15	B3.332	2807	Paderborn
	Schneider	Kommunikation	5	Do. 14 - 15	B3.332	2807	Paderborn
	Willms	Marketing	5	Mi. 9 - 10	C4.201	2504	Detmold

liest	Name	Nr.
	Meier	05325
	Schneider	05325
	Willms	05336

hört	Matr.-Nr.	Nr.
	3255666	05325
	3255666	05336
	3172222	05328
	3272222	05325
	3627343	05328

Abb. 4.3.2/11: Relationales Modell mit Ausprägungen

Der Vorteil relationaler Datenbanken liegt in den einfachen Operationen, mit denen Relationen zerteilt und zusammengefügt werden können. Operationen auf relationale Datenbanken generieren wieder Relationen (= Tabelle). In der relationalen Algebra werden zwei Typen von Operationen unterschieden: Operationen, die auf eine Relation wirken (unitary operations) und solche, die zwei Relationen mit gleichen Spalten miteinander kombinieren (binary operations).

Zu den unitären Operationen gehören:

-> der Selektionsoperator, der aus einer gegebenen Relation eine Untermenge von Sätzen entsprechend einer zu spezifizierenden Bedingung wählt,
Beispiel: Alle Studenten mit einem Alter >30 Jahre
-> der Projektionsoperator, der aus einem relationalen Schema eine Untermenge von Attributen wählt.
Beispiel: Namen und Matrikelnummern aller Studenten

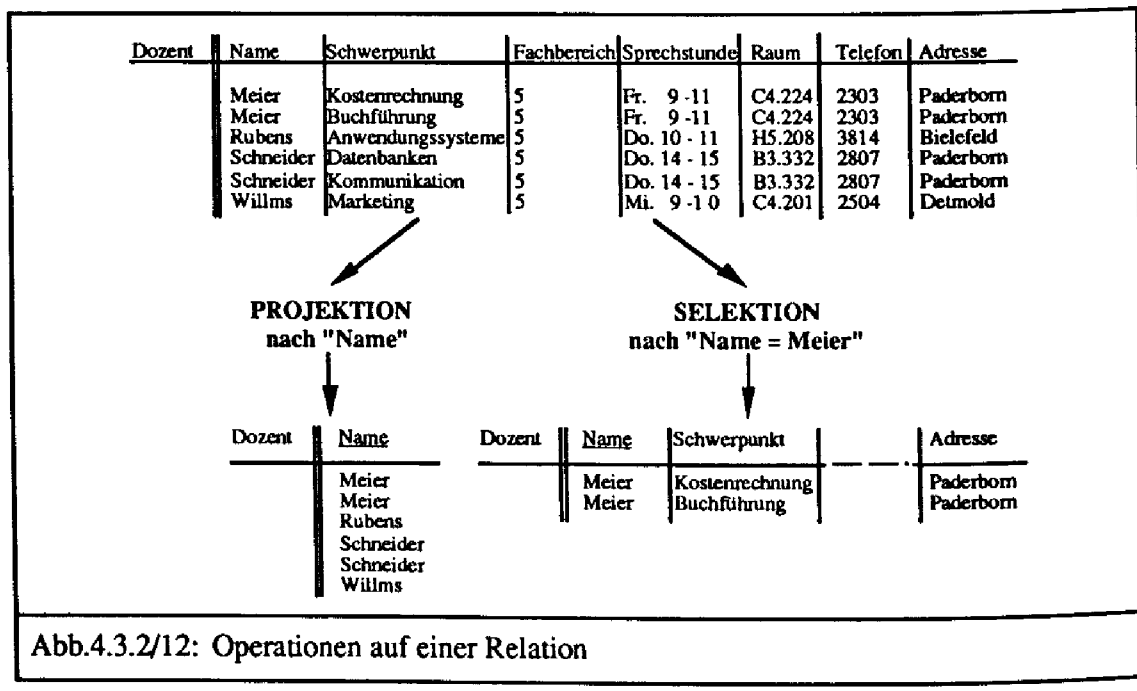


Abb.4.3.2/12: Operationen auf einer Relation

Operationen bezogen auf zwei Relationen mit gleichen Spalten (binary operations) sind:

-> der Vereinigungsoperator (union) bildet eine Vereinigungsmenge über zwei Relationen und sucht alle Sätze aus beiden Relationen, die bestimmten Merkmalen genügen;
-> der Schnittoperator (intersection) bildet eine Schnittmenge über zwei Relationen und selektiert all die Sätze, die sowohl in der ersten als auch in der zweiten Relation vorkommen,
-> der Differenzoperator bezeichnet die Differenzmenge zweier Relationen, also die Menge an Sätzen, die zwar Bestandteil von Relation 1 nicht aber von Relation 2 sind.

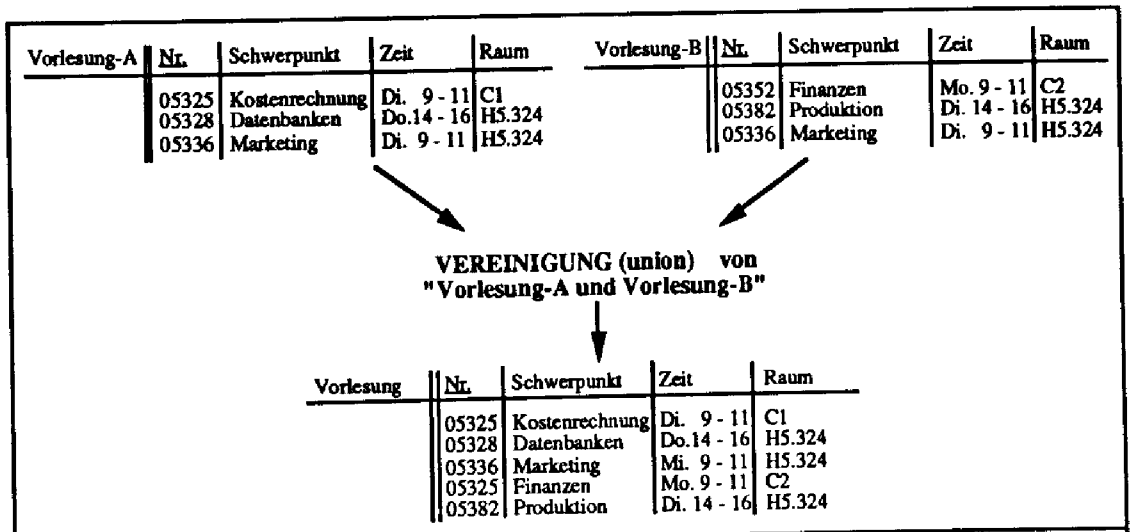


Abb. 4.3.2/13: Vereinigung von Relationen

Durch einen weiteren Operatortyp wird das relationale Modell erweitert. Der Verbundoperator (join) verkettet die Sätze (tupel) zweier Relationen und setzt eine neue Tabelle aus den Spaltenwerten zusammen, die sich auf dasselbe Schlüsselattribut beziehen.

Es gibt mehrere Varianten des Verbundoperators. Der natürliche Verbundoperator (natural join) vergleicht Attributausprägungen zweier Relationen miteinander, sucht die Menge aller Sätze, deren Werte hinsichtlich eines bestimmten Attributs übereinstimmen und streicht redundante Attribute. Weitere Typen erhalten die Redundanz oder lassen neben der Gleichheitsbeziehung weitere Operatoren zu (>, < etc.) (vgl. Schlageter/Stucky (1983), S. 140).

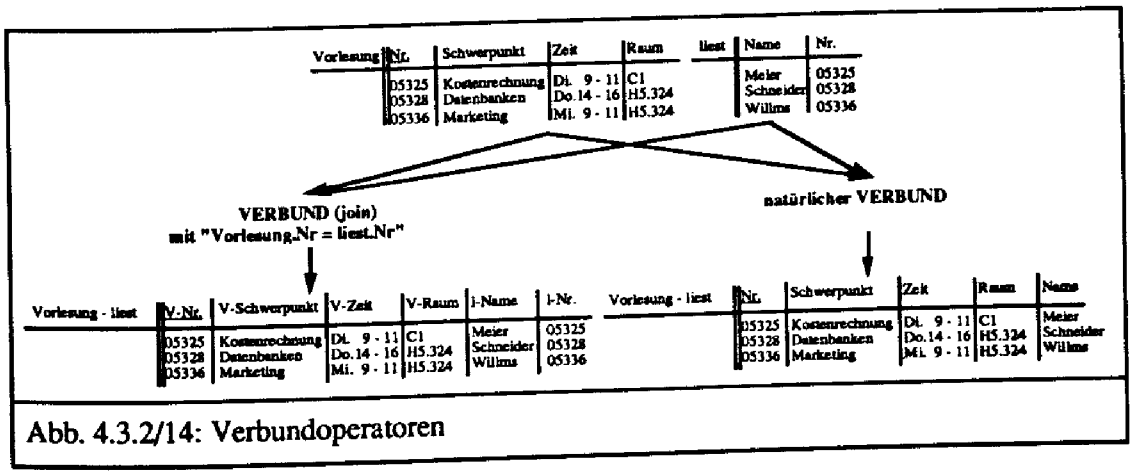


Abb. 4.3.2/14: Verbundoperatoren

Verbundoperatoren sind vorsichtig zu gebrauchen, da es möglich ist, „join-Operation" auf Relationen anzuwenden, die logisch nicht verbunden werden sollen.

Mit dem relationalen Modell ist die Schemabildung nach der sogenannten „Normalformenlehre nach Codd" verbunden. Dieses Verfahren der Strukturzerlegung stellt sicher, daß die Attribute komplexer Objekte (entities) in verschiedenen Tabellen (und damit logisch und physisch verteilt) abgelegt werden, um die Redundanz zu minimieren und physisch Speicherplatz zu sparen. Die Attribute der Objekte werden dann über die Schlüssel in der Anwendung wieder zusammengeführt.

Vorteile:

Relationale Datenmodelle basieren auf einer einsichtigen geschlossenen formalen Logik. Durch die Tabellenstruktur sind sie für den Benutzer einfach zu handhaben. Die Zugriffsoperationen erfolgen deskriptiv, d. h. der Benutzer beschreibt einfach die Attribute und Attributausprägungen, nach denen er sucht. Das Modell besitzt eine hohe Flexibilität, da jederzeit neue Attribute und Sätze und durch zusätzliche Schlüsselkombinationen auch neue Beziehungen eingefügt werden können. Es werden rein logische, rechnerunabhängige Implementierungen einer Datenbank ermöglicht.

Die mit dem relationalen Modell verbundene Normalformenlehre ermöglicht eine redundanzfreie Speicherung (zumindestens für die reinen Nutzinformationen) und vermeidet Zugriffs- und Speicheranomalien, die beim Zugriff und beim Löschen sowie bei der Modifikation Probleme bereiten können.

Vorteile		Nachteile	
Flexibilität	- Änderungen der Struktur und der Schemata einfach	- DV-technische Realisierung mit herkömmlichen Speichertechnologien schwierig - Normalformen senken u. U. Zugriffseffizienz	**DV-Effizienz**
Redundanzfreiheit	- Normalformenlehre vermeidet Redundanz	- Ausdrucksmächtigkeit bei rekursiven Datenbeziehungen und bei mehrwertigen Attributen beschränkt	**Anwendungseffizienz**
Zugriffsoperationen	- Deskriptiver Zugriff mit Operationen auf Attributmengen		
Transparenz	- Tabellenstrukturen und deren Verknüpfung über Schlüssel sind klar und einsichtig		

Abb. 4.3.2/15: Vor-/ Nachteile des relationalen Modells

Nachteile:

Die Satzorientierung relationaler Modelle bedingt, daß komplexe Objekte zersplittert werden (Preis für die Normalisierung), d. h. die gewünschten Informationen müssen aus mehreren Tabellen zusammengesucht werden. Die Normalformen sind syntaktisch orientiert und berücksichtigen nicht den semantischen Zusammenhang der Attribute oder die Häufigkeit bestimmter Operationen der Datenbank.

Die Orientierung an der Mengenlehre verleiht dem relationalen Modell zwar formale Eleganz, doch ist es logisch komplex bei großen Anwendungen nur aufwendig zu entwickeln. In Standardsoftware, die relationale Datenbanken verwendet, wird daher häufig auf bestimmte formal-logische Eigenschaften (z. B. die strikte Normalisierung) verzichtet, um im konkreten Fall eine höhere Performance zu erreichen.

Wird das relationale Modell bei einer Anwendung konsequent umgesetzt, so wird zwar die Redundanz minimiert, doch benötigt es sehr hohe Rechnerleistung (inklusiv Pufferspeicher) für die Vielzahl der Zugriffsoperationen bei der Zusammenführung der Attribute eines Objektes aus verschiedenen Tabellen (substitutiert Speicherplatz durch Rechnerleistung).

Das relationale Modell ist logisch nicht in der Lage, rekursive Beziehungen (Tupel - Subtupel - Relationen) abzubilden. Diese können nur umständlich über universelle Programmiersprachen erfaßt werden. Wichtig sind rekursive Anfragen z. B. bei Stücklistenstrukturen oder Organisationshierarchien. Das relationale Modell kann logisch nur einfache Integritätsbedingungen erfassen und bedingt daher, daß der gleiche Satz mit Bedingungen an verschiedenen Stellen der Anwendung überprüft wird (-> Code-Redundanzen, Semantische Interpretationsspielräume). Datenbanksysteme auf der Basis des relationalen Modells sind zur Zeit bei kleinen und mittleren Anwendungen sehr verbreitet, da in diesen Fällen die Flexibilitätsanforderung hoch sind und angesichts heutiger Rechnerleistungen die Performance-Anforderungen weniger gewichtig sind. Bei großen kommerziellen Anwendungen sind zur Zeit oft die Performance-Nachteile noch hinderlich. Zudem benötigen solche Anwendungen häufig rekursive Strukturen (z. B. in Produktionsdatenbanken), für die das relationale Modell wenig geeignet ist.

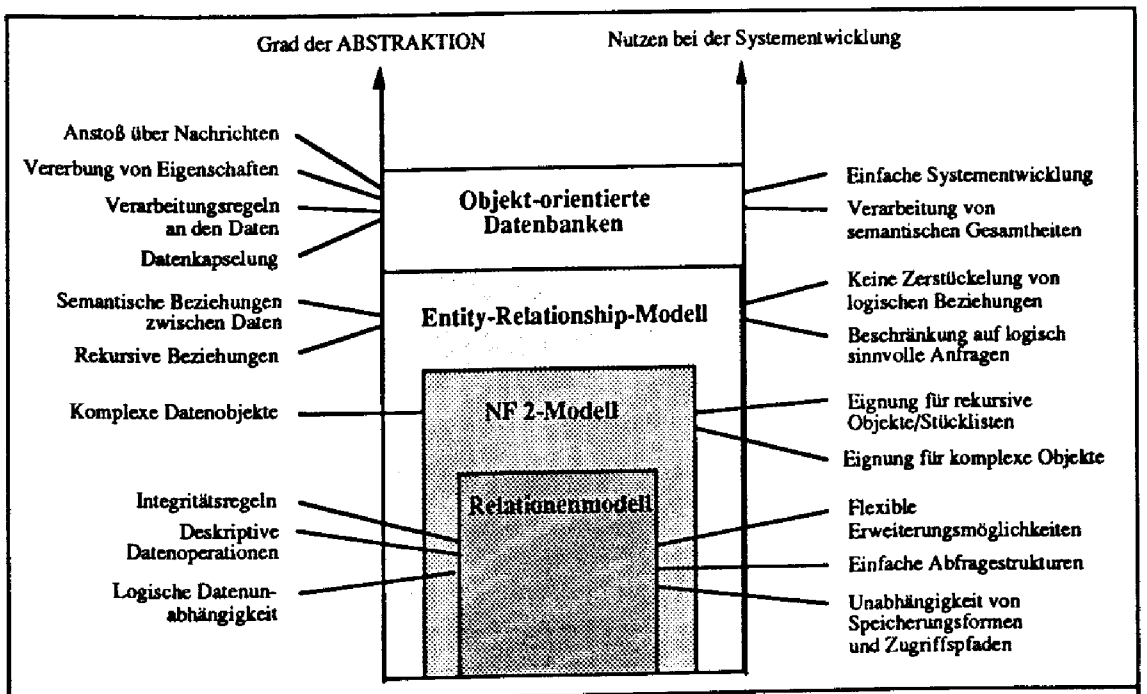


Abb. 4.3.2/16: Unterschiede zwischen modernen Datenbankkonzepten

4.3.2.1.4. Konstruktiv orientierte Modelle

Diese Modelle orientieren sich pragmatisch an der physischen Speicherorganisation und technisch-effizienten Möglichkeiten. Beispiele sind die Systeme ADABAS der SOFTWARE AG oder SESAM der SIEMENS AG. Diese Systeme verwenden die klassischen physischen Speicherungsformen wie indexsequentielle und invertierte Dateien. Dabei werden alle Sätze einer Datei nach den Werten eines Attributs oder mehrerer Attribute in Tabellen geordnet. Die Tabellen (invertierte Listen) enthalten neben den Attributwerten nur noch Adreßhinweise (vgl. Zehnder (1989), S. 107ff).

Beispiel: ADABAS der SOFTWARE AG

ADABAS arbeitet mit invertierten Listen für jeden Datenbankdeskriptor, die neben der Ausprägung des Deskriptors die Anzahl der zugehörigen Datensätze und eine logische Adresse des physikalischen Satzes enthält.

Ausprägung	Anzahl	Logische Adresse (ISN=internal sequence number)
Meier	2	3, 23
Rubens	1	8
Schneider	2	2, 6
Wilms	1	4, 7

Abb. 4.3.2/17: Eine invertierte Liste für die Dozenten

Über einen Adreßkonverter wird dann die physische Adresse eines Datensatzes bestimmt. Sucht man etwa den zweiten Datensatz zu Schneider, so hat dieser die ISN = 6, im Adreßkonverter steht seine physische Adresse an sechster Stelle.

Speicherblock							
1	1	1	2	12	2	12	2

Der Adreßkonverter gibt den physischen Speicherblock an.

Im Block 2 des physischen Speicherraums stehen somit folgende Sätze:

	DOZENTEN.NR.			VERANST.NR.
4	4711	WILMS	Marketing	921128
6	4795	SCHNEIDER	Wirtschaftsinformatik	921117
8	4723	RUBENS	Finanzierung	914322
etc.	etc.			

ADABAS speichert die logischen Sätze in der Reihenfolge ihres Auftretens in physischen Speicherblöcken ab.

Speicherraum				
Logischer Satz 1	Logischer Satz 2	...	Logischer Satz n	Erweiterungsraum
4711 Wilms	4795 Schneider			

ADABAS teilt somit den physischen Speicherplatz in Blöcke ein. Diese Speicherblöcke umfassen einen "Erweiterungsraum", der Ausdehnungen der Satzgröße erlaubt. Datenkomprimierungstechniken werden zur Senkung des Speicherplatzbedarfs genutzt.

Die Abfrage-/Datentransaktionsmöglichkeiten der ursprünglichen ADABAS reichen nicht an das relationale Modell heran. So benötigt man z. B. für die Suche nach den Modalitäten einer bestimmten Veranstaltung eines Dozenten 2 Befehle,

FIND DOZENTEN mit DOZNR
FIND VERANSTALTUNGEN mit VERANSTNR

während die gleiche Abfrage in einer relationalen Datenbank nur einen Befehl benötigte. Jedoch wird dieser Nachteil durch die sehr effiziente Speicherorganisation im praktischen Betrieb aufgehoben. ADABAS verfügt über gewisse Eigenschaften von NF-2-Datenbanken, indem es pro Attribut mehrere Ausprägungen zuläßt.

Vorteile		Nachteile	
DV-Effizienz	- Direkter Bezug zu verbreiteten physischen Speicherungsformen	- Für Zugriffseffizienz sind umfangreiche Hilfsdaten (invertierte Dateien, Indextabellen) notwendig	**Redundanz**
Anwendungseffizienz	- Datensätze können von variabler Länge sein (Möglichkeit mehrwertiger Attribute)	- durch Abhängigkeit von physischen Strukturen eingeschränkt	**Flexibilität**
Transparenz	- Gute Überschaubarkeit auf der logischen und physischen Ebene	- erlaubt nur beschränkt deskriptive Operationen	**Zugriffsoperationen**

Abb. 4.3.2/18: Vor-/ Nachteile des konstruktiven Modells

4.3.2.2. Semantische Datenmodelle

4.3.2.2.1. Objekt-Beziehungs-Modelle

Zwei Forschungsrichtungen versuchen zur Zeit auf der Grundlage der semantischen Datenmodellierung funktionierende Datenbanksysteme zu entwickeln:

- Ansätze zur Erweiterung des Entity-Relationship-Modells
- Ansätze mit objektorientierten Modellen

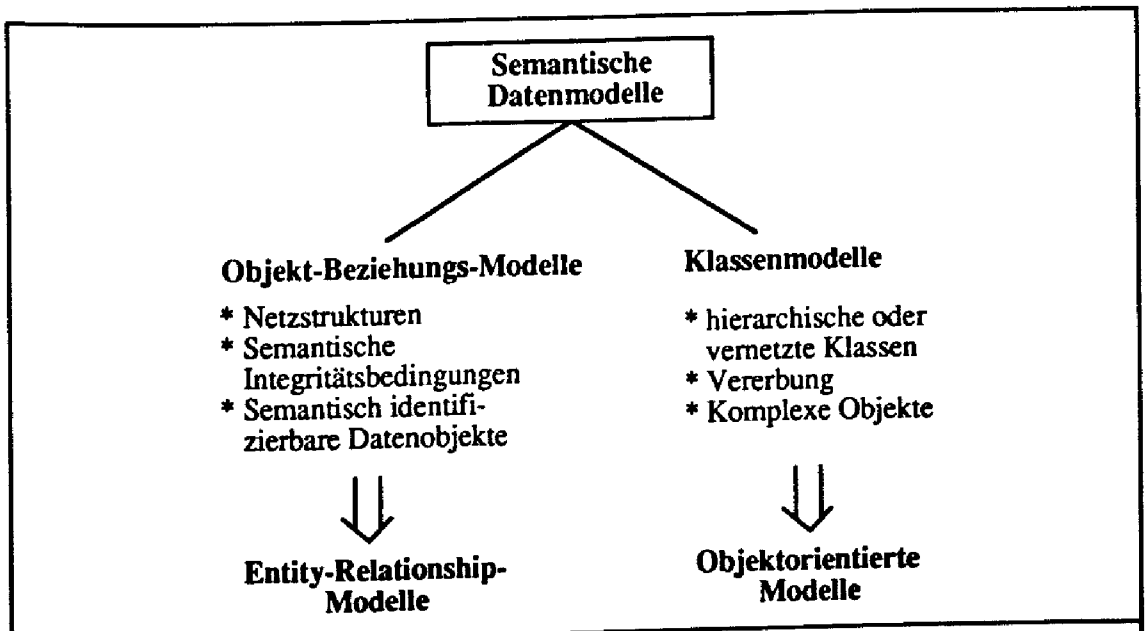


Abb. 4.3.2/19: Ansätze semantischer Datenmodelle

Objekt-Beziehungsmodelle bauen auf den Entity-Relationship-Modellen nach Chen oder Bachman auf. Diese besitzen gewisse semantische Eigenschaften. Dazu gehören:

- Eindeutige Unterscheidungen zwischen Datenobjekten und Datenbeziehungen sowie der jeweils zugehörigen Attribute,
- Unterstützung von semantischen Konstruktionsoperatoren (Konnexion, Generalisierung, Abstraktion, Identifizierung),
- Definierbare Typen von Relationen (1:1, 1:n, n:m) und die Möglichkeit, daraus bestimmte Integritätsbedingungen zu definieren.

Diese Vorteile haben zu Bestrebungen geführt, das Konstruktionshilfsmittel ERM durch ein entsprechendes Datenmodell und eine formale Datenbanksprache zu ergänzen. Zwar besteht eine große Abwärtskompatibilität des ERM mit dem Relationalen Modell, doch wollen Software-Hersteller (u. a. SOFTWARE AG mit ADABAS ENTIRE) dadurch u. a. eine höhere semantische Mächtigkeit erreichen.

Eine entsprechende Datenbanksprache müßte folgende Anforderungen erfüllen:

- Definitionssprache für abstrakte Datenobjekttypen
- Zugriff und Manipulationsmöglichkeiten für Attribute und Attributstrukturen
- semantische Unterscheidung zwischen Datenobjekten und Datenbeziehungen
- Identifizierung und Zugriff auf Datenobjekte und Datenbeziehungen über Attributinhalte, unabhängig von Schlüsseln
- umfassende semantische Integritätsprüfungen aufgrund semantisch definierter Wertedomänen für die Attribute

Beispiele	Kennzeichen	DV-Realisierung
Entity-Relationship Modell (Chen 1977)	- Flache Datenstrukturen (Satzorientierung) - Erweitertes Netzwerkmodell mit Beziehungen zwischen Objekttypen	SAG ADABAS-Erweiterung
Objekt Role Model (Bachmann 1977)	- Flache Datenstrukturen (Satzorientierung), aber rekursive Beziehungen zwischen Datenobjekten gleichen Typs - Erweitertes Netzwerkmodell mit 1:1-Beziehungen zwischen Objekten - Dynamische Rollen von Datenobjekttypen (Abbildung in "Role Records")	Prototyp-Implementierung durch Honeywell-Bull
Molecular Data Model (Bachmann 1983)	- Satzorientierung bei der Abspeicherung der Datenobjekte - "Partnership Sets" stellen logische Verbindungen definierbaren Typs (und Regeln) her - Erweitertes Netzwerkmodell mit 1:1- und n:m-Beziehung	

Abb. 4.3.2/20: Ansätze zur Entwicklung von Objekt-Beziehungs-Modelle

Um Datenbankrealisierungen von Entity-Relationship-Modellen ist es zur Zeit still geworden. Offensichtlich sind die Vorteile gegenüber relationalen Modellen nur bei speziellen Anwendungen (z. B. das Data Dictionary PREDICT CASE der SOFTWARE AG) so groß, daß sich der zusätzliche Aufwand in der Implementierung lohnt.

4.3.2.2.2. Objektorientierte Modelle (Klassenmodelle)

Mit objektorientierten Datenmodellen sollen Datenkonzepte entwickelt werden, die die Struktur und das Verhalten von Umweltobjekten beliebiger Art und Komplexität uneingeschränkt durch Datenobjekte erfaßbar machen und den "semantic gap" vermeiden, d. h. ein Sachverhalt der Realität soll jeweils durch genau ein Datenmodellkonstrukt (1:1-Abbildung) erfaßt werden und nicht (wie in den klassischen Modellen) über eine 1:n-Zerlegung in mehreren Einzelkonstrukten. Dies soll sowohl hinsichtlich struktureller (datenmäßige Repräsentation) als auch verhaltensmäßiger Aspekte (Operationen auf Sachverhalte) gelten. Durch Möglichkeiten anwendungsspezifischer Datenobjekte und Operatoren soll die Lücke zwischen Datenhaltung und Anwendungsprogrammierung verkleinert werden (vgl. Dittrich (1987,1989, 1990), Dabrowski/Fong/Yang (1990)).

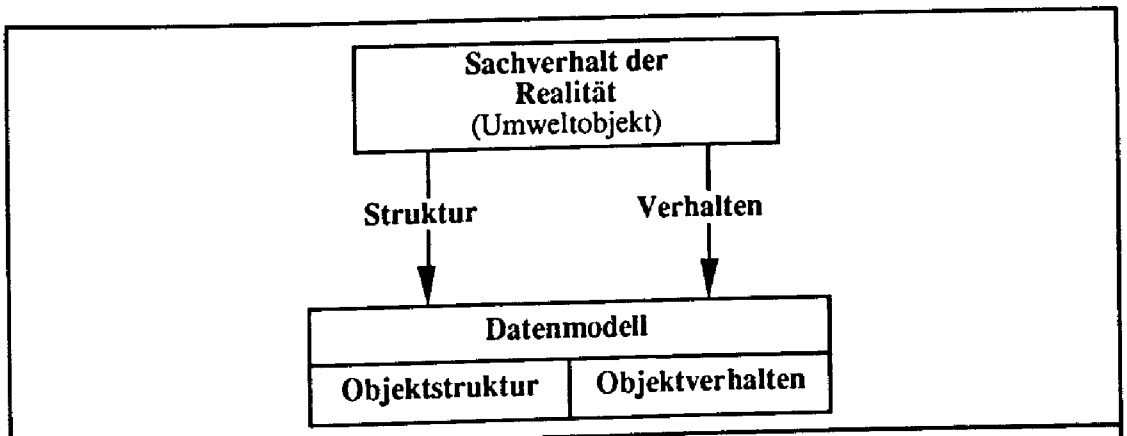


Abb. 4.3.2/21: Berücksichtigung struktureller und verhaltensmäßiger Aspekte beim Datenmodell

In struktureller Hinsicht ist es das Anliegen, nicht nur werteorientierte (alphanumerische) Datenstrukturen abbilden zu können, sondern auch komplexe Datenobjekte wie Grafiken, Dokumente etc. (z.B. im CAD-Bereich), die ihrerseits wiederum aus Unterobjekten bestehen. Dabei können die Objekte hierarchisch aufgebaut sein; es kann aber auch vorkommen, daß ein Unterobjekt Bestandteil mehrerer Objekte ist (überlappende Objekte).

Der Anwender soll in der Lage sein, benutzerspezifisch neue Objekttypen und die für diesen Typ zulässigen Operatoren zu definieren.Dabei soll er in der Lage sein, den inhaltlichen Zusammenhang der Daten zu wahren und nicht gezwungen werden, diesen durch Normalisierungsschritte o.ä. zu zerstören.

Die bisherigen Ansätze bauen auf Konzepte "objektorientierter Programmierung" auf, d. h. sie reichern auf der einen Seite herkömmliche Datenbankkonzepte um Konzepte der objektorientierten Systementwicklung an und sie erweitern objektorientierte Programmiersprachen (z. B. SMALLTALK) um Datenbankeigenschaften (z. B. dauerhafte Speicherung von Objekten, Mehrbenutzerzugriff). Wichtige Merkmale sind dabei die Verwaltung von Objektklassen und die Vererbung von Eigenschaften zwischen Objekten. Klassendefinitionen enthalten Vererbungsregeln innerhalb einer Klassenhierarchie (Oberklasse zu Unterklasse) sowie Beziehungen zwischen verschiedenen Klassen (analog ERM). Die Klassendefinitionen können eindimensional (Baumstruktur) oder mehrdimensional (Netzstruktur) sein.

Forderung	Ausprägung	Kennzeichen
1. Modellierung komplex aufgebauter Objekte	1.1 Komplexe Objekte	Objekte können aus mehreren Unterobjekten zusammengesetzt sein.
	1.2 Objekttypen	Datenobjekte sind jeweils zu spezifizieren als Instanzen bestimmter Objekttypen.
	1.3 Objektoperatoren	Datenoperatoren zum Umgang mit komplexen Objekten und zum Auf-/Abbau von Objektstrukturen.
2. Definitionskonzept für neue Objekttypen	2.1 Benutzerspezifizierung von Typen/Operatoren	Der Benutzer kann strukturell neue Objekttypen definieren und die für diesen Typ charakteristischen Operatoren selbst festlegen.
	2.2 Objektidentifikator	Alle Objekte müssen unabhängig von ihren aktuellen Werten eigenständig identifizierbar sein; die Vergabe von Schlüsseln soll damit überflüssig sein.
	2.3 Typhierarchien und Vererbung	Ein Objekttyp kann ein Untertyp (Spezifizierung) eines Obertyps sein; erbt als solcher alle strukturellen und verhältnismäßigen Eigenschaften des Obertyps.

Abb. 4.3.2/22: Anforderungen an objektorientierte Datenbanken (vgl. Dittrich (1990))

Objektorientierte Datenbanken sollen Anwendungssachverhalte genauer in der Datenbank repräsentieren können. Für betriebswirtschaftliche Anwendungen sind von besonderer Bedeutung:

- die explizite Unterstützung von Objektversionen, die flexible Strukturobjekte erlaubt,
- die Ausrichtung auf komplexe Objekte, die z.B. rekursive Strukturen ermöglicht,
- die Möglichkeit der Vererbung, die die Modularisierung und Übertragung von Konzepten erlaubt und damit den Datenbankentwurf verbilligt (Wiederverwendbarkeit),
- die Datenkapselung, die die Schnittstellenstruktur und die Konsistenzprüfung vereinfacht,
- die unmittelbare Verbindung der Objektstruktur zu den vorgelagerten Stufen Datenkonstruktion und Datenmodellierung,
- die erleichterte Konsistenzprüfung durch benutzerspezifische Definition der Operatoren.

Der grundsätzliche Vorteil objektorientierter Ansätze liegt in dem „Mehr an Semantik" in der Datenbasis, das es einerseits erlaubt, das natürliche Denken in Objekten (als Verbindung von Informationen und Funktionen) ohne Bruch in ein DV-Modell umzusetzen, andererseits aber auch die künstliche Zerlegung in Programm und Datenteile erspart und daher die Entwicklung und den Betrieb eines DV-Anwendungssystems erleichtert.

Merkmal	Kennzeichen	Bewertung
1.1 Komplexe Objekte	bestehen aus Subobjekten in hierarchischen oder vernetzten Strukturen	+ unterstützen semantische Konstruktionsoperatoren (is part of) + unterstützen Operationen, die auf Relationen beruhen
1.2 Objekttypen	Objekte gleichen Typs (Attribute, Verhalten) werden zu Objektklassen zusammengefaßt. Klassen werden teilweise auch als abstrakte Datentypen aufgefaßt. Klassen definieren sich - über gleichartige Datenstrukturen (instance variables) - über gleichartige Schnittstellen (common protocols) - über gleichartige Methoden	+ Klassen dienen als Schema (templates) für die Bildung neuer Klassen + ermöglichen Vererbung (inheritance) von Eigenschaften + ermöglichen die Wiederverwendung von Programmcode Im internen Schema können Klassen als Satztypen gedacht werden, die alle Meta-Daten enthalten, um einen speziellen Satz zu bilden und zu benutzen.
1.3 Objekt-operatoren	- Datenkapselung (Encapsulation) Interne Repräsentation eines Datenobjekts wird verdeckt ("eingekapselt"), sichtbar bleibt nur die Menge der benutzerdefinierten Operatoren des Typs (ADT - abstrakte Datentypen)	+ vermeidet unsinnige Werte von Objekten und erleichtert damit Konsistenzprüfungen + ermöglicht die Definition neuer, auf Byteketten aufbauender primitiver Datentypen + Trennung des Objektes in Schnittstelle (interface) und interner Zustand + Verbindung von Daten und Prozedur in einem Objekt und in einer Datenbank; kein separates Programm in separatem File
2.1 Benutzer-spezifizierung von Objekten/ Operatoren	Polymorphes Verhalten (polymorphic behavior) - das Verhalten eines Objektes auf eine Nachricht hängt von der Klasse des Objektes ab - d. h. dynamische Bindung der Reaktion auf eine Nachricht (run-time binding), d. h. die Prozedur, die auf eine Nachricht antwortet, wird zur Laufzeit erst festgelegt. Nachrichtenorientierung, d. h. die Objekte tauschen Nachrichten aus, die beim Empfänger Prozeduren auslösen, die die Antwort auf die Nachricht generieren.	+ Anknüpfung an vorgelagerte Datenmodellierung wird erleichtert + Objektversionen unterstützen Abbildung von zeitlichen und sachlichen Generationen + Wahrung des inhaltlichen Zusammenhanges von Daten ermöglicht Performance-Vorteile
2.2 Objekt-identifikator	Datenobjekt wird eigenständig über Identifikator (Surrogat) und nicht (wie in klassischen Modellen) über seinen "Wert" identifiziert	+ zwei Objekte können sich das gleiche Objekt teilen (Object Sharing) + Das Update des Sohn-Objekts aktualisiert sowohl das Vater- als auch das Mutter-Objekt
2.3 Typhierarchien und Vererbung	das Abstraktionsprinzip der Generalisierung wird über Vererbung (wie in den objektorientierten Programmiersprachen) realisiert, dazu können Obertyp-Untertyp-Beziehungen definiert werden	+ unterstützt saubere, stufenweise verfeinernde Entwurfsmethode + reduziert Redundanz, + spezifische Semantik muß sauber definiert werden. + Vererbungskonzept unterstützt die semantischen Konstruktionsoperatoren GENERALISIERUNG und SPEZIFIZIERUNG

Abb. 4.3.2/23: Merkmale objektorientierter Datenbanken

Bei der Entwicklung objektorientierter Datenbanksysteme existieren zwei Entwicklungsrichtungen

(1) Erweiterung objektorientierter Programmiersprachen
(2) Erweiterung der Funktionalität üblicher Datenbanksysteme

	Ansatzpunkt	
Merkmal	**Objektorientierte Programmiersprachen**	**Objektorientierte Datenbanken**
Grundansatz	Ausrichtung auf prozedurale Eigenschaften innerhalb von Objektklassen	Ausrichtung auf strukturelle Eigenschaften komplexer Objekte
Objektidentität	Identifizierung über Stellvertreter (Schlüssel)	Identifizierung über Objektinhalte und Objektidentifikatoren
Objekt-/Datenkapselung (object encapsulation)	Verbindung von Operationen und einfachen Objekten	Datenzugang nur über objektspezifizierte Operatoren
Klassenstruktur	Explizite Definition	Implizite Definition über gemeinsame Merkmale/Merkmalsausprägungen möglich
Nachrichtenorientierung (message passing)	Frei programmierbare Nachrichten	Bereitstellung von Nachrichtentypen für Objekttypen
Vererbungskonzept (class inheritance)	i. d. R. hierarchische Vererbung von Methoden (one dimensional inheritance)	in fortgeschrittenen Konzepten auch multidimensionale Vererbung von Methoden/Attributen (multidimensional inheritance)
Polymorphes Verhalten	Dynamische Bindung der Reaktion auf eine Nachricht	Verankerung des Verhaltens auf eine Nachricht im Objekt
Komplexe Objekte (composite objects)	i. d. R. nur einfache Objekte	Ausrichtung auf komplexe Objekte

Abb. 4.3.2/24: Vergleich objektorientierter Programmiersprachen und Datenbanken

Neben einer Reihe von Prototypensystemen existiert eine zunehmende Anzahl von Produkten auf dem Markt.

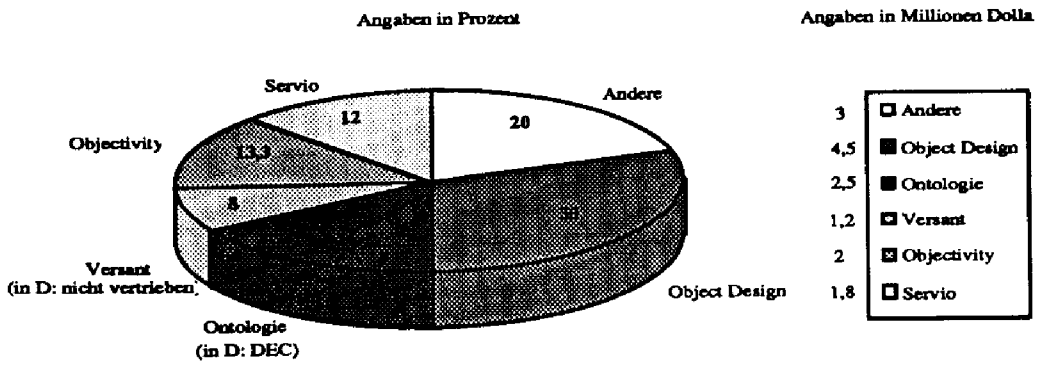


Abb. 4.3.2/25: Der Markt für objektorientierte Datenbanksysteme (Quelle: Computerwoche 1992 1/2,S.11)

Strukturell objektorientierte Datenmodelle

Von strukturell-objektorientierten Datenmodellen (structural object oriented) spricht man, wenn ein Konzept zur Modellierung komplexer Objekte vorhanden ist (Forderung 1), andere Punkte z. B. ein Objektidentifikator und ein Vererbungskonzept können hinzukommen.

Strukturell objektorientierte Modelle

-> erlauben die Konstruktion komplex aufgebauter Objekte durch beliebig verschachtelte Tupel-/Mengenbildung,
Die komplexen Datenobjekte oder auch „Komposit-Objekte" sind ihrerseits wieder aus Datenobjekten zusammengesetzt.
-> besitzen Operationen zur Arbeit mit komplexen Objekten, die fest vorgeben sind *(Beispiel: Auffinden und Lesen eines Objekts mit allen Bestandteilen, Navigation durch die komplexe Objektstruktur).*
-> besitzen die Möglichkeit zur Definition und Abspeicherung von Operationen und Prozeduren.

Die in der Literatur aufgeführten Systeme IBM-XSQL; IBM-AIM-P; PRIMA, DAMOKLES sind sämtlich Forschungsdatenbanken; in marktreifen Produkten taucht dieser Ansatz bisher kaum auf (vgl. Dittrich (1990), S.234). Eine Ursache dafür ist die komplizierte Speicherungstechnik. Diskutiert wird beispielsweise (vgl. Adiba (1987), S. 101)

-> die direkte Objektspeicherung (Direct Storage Model), bei der die komplexen Objekte physisch zusammenhängend in einen Adreß-/Speicherraum abgelegt werden
-> die normalisierte Objektspeicherung (Normalized Storage Model), bei der die komplexen Objekte in atomare Objekte zerlegt werden und in flachen Tabellen abgespeichert werden
-> die geschachtelte Speicherung, indem als Attribute einer Relation selbst wiederum (Unter-) Relationen zugelassen sind (NF2-Datenmodell).

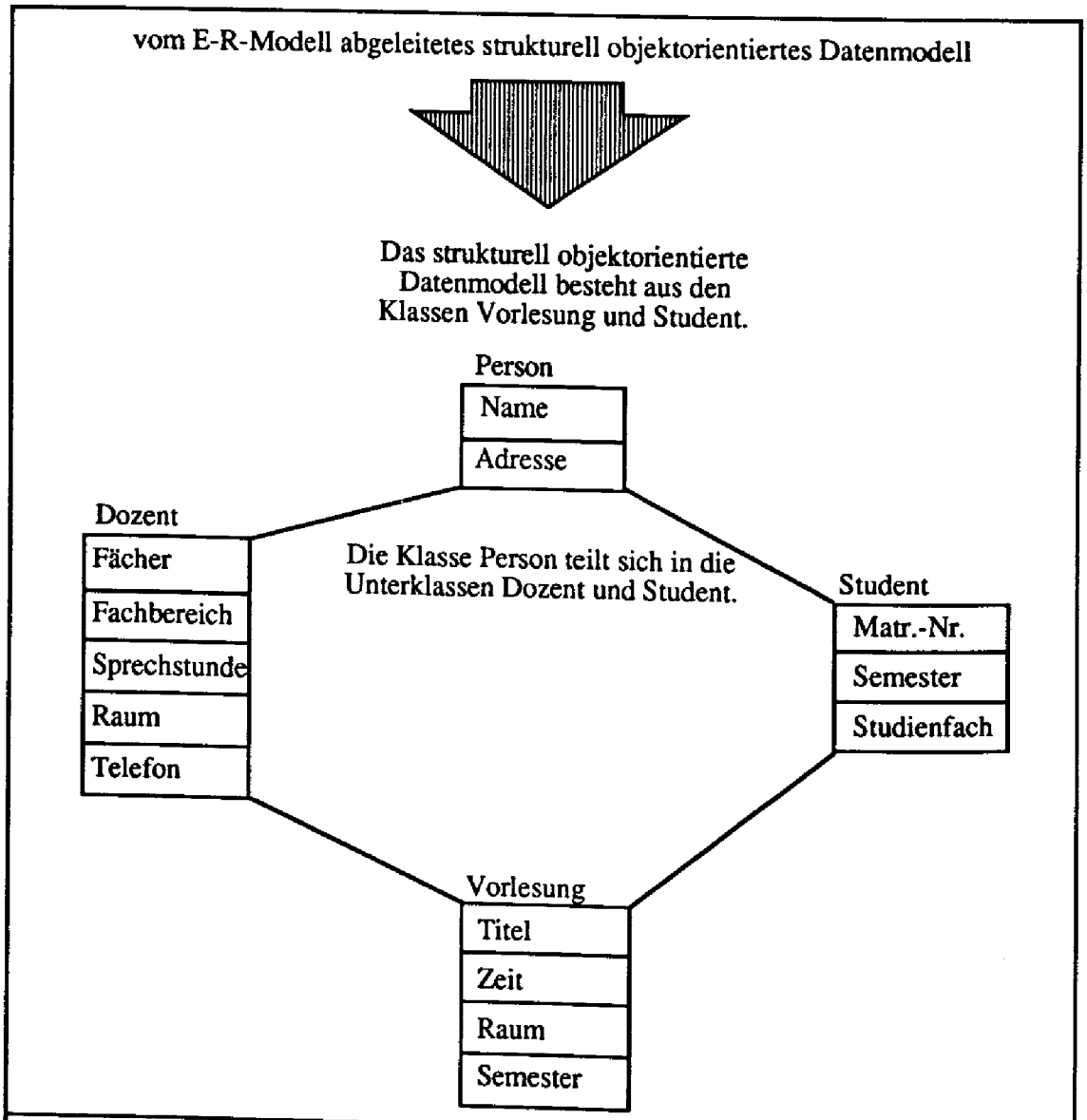


Abb. 4.3.2/26: Beispiel für ein strukturell objektorientiertes Modell

Strukturell objektorientierte Modelle sind im engeren Sinne noch keine semantischen Datenmodelle. Sie bieten nur die Möglichkeit, komplexe Datenobjekte abzuspeichern.

NF2-Objektmodelle

NF2-Objektmodelle (Non First Normal Form) gehören zur Gruppe der strukturell-objektorientierten Datenbanken (vgl. Dittrich (1990), Vossen/Witt (1990)). Kennzeichen von NF2-Modellen ist, daß einzelne Attributwerte nicht skalar sein müssen, sondern ihrerseits strukturiert sein können. Es wird auf die erste Normalform verzichtet, als Attributwerte sind Wiederholungen oder Aggregationen von Werten zulässig Erlaubt sind teilweise auch komplexe Datenobjekte, z. B. Prozeduren, Bedingungen, Tabellen und Rekursionen.

NF2-Objektmodelle sind der Versuch, mit dem relationalen Datenmodell nicht-traditionelle Anwendungsfelder (Bürokommunikation, CAD/CAE-Anwendungen etc.) zu bedienen, die die Speicherung von Texten, Graphiken oder Operationen oder von Sprache, Geräuschen bedingen.

Bei NF-2-Objektmodellen mit eingebetteten Relationen (NF2-Relations) ist es möglich, daß bestimmte Attribute von Datensätzen ihrerseits wiederum Relationen (= Tabellen) enthalten. Dieses können flache Tabellen oder "geschachtelte" Relationen sein, deren Attribute nochmals Tabellen enthalten . Damit lassen sich hierarchische Objektstrukturen abbilden und zusammenhängende Teile komplexer Objekte müssen nicht verstreut abgespeichert werden. Dies erleichtert den Zugriff und vermeidet Join-Prozeduren. In anderen NF2-Modellen wird versucht, im Relationenmodell beliebige strukturierte Datenobjekte abzubilden. (vgl. Vossen/Witt (1990), S.102).

In der einfachsten Form der NF2-Datenbank wird der Konstruktor „Tabelle (= Relation)" nicht nur einmal, sondern „hierarchisch geschachtelt" mehrfach angewendet. Man erhält geschachtelte Tabellen: Tabellen in Tabellen in Tabellen. Dabei wird jedoch am fixen Relationenschema festgehalten, d. h. die Schachtelungstiefe ist ausprägungsunabhängig im Schema der Relation festgelegt. Da der grundsätzliche Konstruktor bei dieser Form der NF2-Datenbank nicht durchbrochen wird, kann die Logik der Operationen und damit auch der Datenbanksprachen beibehalten werden (vgl. Scholl/Schek (1990), S. 106)).

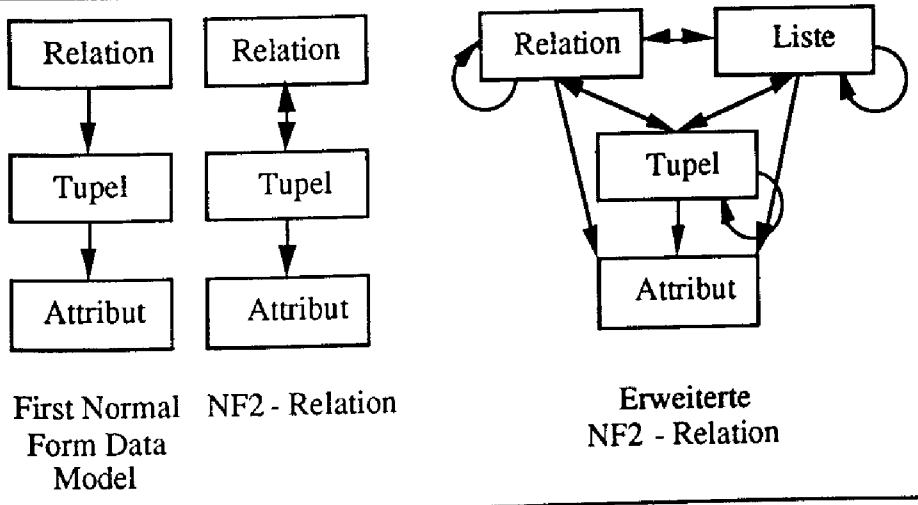


Abb. 4.3.2/27: Unterschiede der NF2-Datenmodelle (vgl. Dadam/Linnemann (1989), S. 666)

Weitergehende Ansätze lassen rekursiv geschachtelte Relationen zu, um Netzstrukturen zwischen den Objekten abbilden zu können. Damit ersparen sie dem Nutzer, sich auf eine Stelle in der Hierarchie festzulegen, das Schema ist symmetrisch. Das gestattet die Formulierung von Abfragen in der jeweils am besten geeigneten hierarchischen Sicht; die Transformation auf das jeweilige interne Schema leistet das System (vgl. Scholl/Schek (1990), S. 112)).

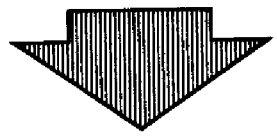
Die Datendefinitionssprache bei erweiterten relationalen Modellen besteht im Grundsatz aus 4 Konstruktoren: e = für einfache Typen, l = für Listen, m für Mengen und t für Tupel

Klasse	Kennzeichen	Datenmodelle (Beispiele)	Datendefinitions-sprache enthält	Rekursion erlaubt
Erweiterte Relationenmodelle	- Mengen-Listen-konstruktoren werden bei der Typenbildung benutzt	Erweitertes NF2-Modell Lau Rel AIM - P	(mtl) + beliebig häufig hinterein-ander	ja
Relationenmodelle mit mehrstufigen Hierarchien	- atomare Attributwerte - hierarchisch ineinander geschachtelte Relationen	Verso SQL /NF	(mt) + (beliebig häufig hinterein-ander)	nein
Relationenmodelle mit einstufig nicht atomaren Attributwerten	- Attributwerte können Tupel, Mengen oder Listen sein	1 N F + - Modell	mt mtt mtm mtl	nein
Flaches Relationenmodell	- atomare Attributwerte - flache Relationen	1NF - Modell	mt	nein

Abb. 4.3.2/28: Gegenüberstellung der Relationenmodelle

NF2-Datenstrukturmodelle ermöglichen die Modellierung von Objekttypen für Non-Standard-Datenbankanwendungen (geometrische Objekte, Koordinatenangaben, Dokumente). Vorteile bieten sie auch bei komplexen, hierarchisch aufgebauten Objekttypen (z. B. bei Stücklisten und Organisationsstrukturen).

vom E-R-Modell abgeleitete



Relationenmodell mit einstufig nicht atomaren Attributwerten:
Zusammenfassung der Attribute Zeit und Raum des Entities Vorlesung zur Relation Belegung

Vorlesung	Durchführung	
	Zeit	Raum
internes Rechnungswesen	Di. 9 - 11	C1
Datenbanken	Do. 14 - 16	H5.324
Marketing	Mi. 9 - 11	H5.324

Mehrstufige relationenwertige Hierarchien:
Zusammenfassung der Attribute Zeit und Raum zur Relation Belegung

Vorlesung	Durchführung		
	Durchführung	Zeit	Raum
internes Rechnungswesen		Mo. 11 - 16	C1
		Di. 9 - 11	P7.206
Datenbanken	Durchführung	Zeit	Raum
		Mo. 14 - 16	B1
		Di. 11 - 13	H2
		Do. 14 - 16	B1
Einführung in die DV	Durchführung	Zeit	Raum
		Mi. 9 - 11	H4
		Fr. 11 - 13	H1

Erweitertes Relationenmodell:
Zusammenfassung der Attribute Zeit und Raum zur Relation Belegung, Liste von Bezeichnungen für eine Vorlesung, Rekursion durch neu hinzugekommenes Attritut Qualifikation

Vorlesung	Durchführung			Qualifikation
	Durchführung	Zei	Raum	
internes Rechnungswesen		Mo. 11 - 16	C1	Kostenrechnung
		Di. 9 - 11	P7.206	
Datenbanken, Informations-systeme	Durchführung	Zei	Raum	Einführung in die EDV
		Mo. 14 - 16	B1	
		Di. 11 - 13	H2	
		Do. 14 - 16	B1	
Einführung in die DV, BWL V: EDV, DV-Grundstudium	Durchführung	Zei	Raum	-
		Mi. 9 - 11	H4	
		Fr. 11 - 13	H1	

Abb. 4.3.2/29: Beispiele für die Typen von NF2-Datenmodellen

Verhaltensmäßig objektorientierte Datenmodelle

Verhaltensmäßig-objektorientierte Datenmodelle (operationaly object-oriented) ermöglichen die Definition neuer, einfach strukturierter Objekttypen und spezifischer Operatoren durch den Benutzer. Beispielsweise erlaubt ein solcher Operator das Auffinden, Modifizieren und Löschen eines bestimmten Objekttyps samt seiner Untertypen. Dazu sind drei Schritte erforderlich: (1) für die Objekte ist deren individuelle Struktur (Attribute, zugehörige Wertedomänen) und deren Verknüpfung zu Klassen zu definieren, (2) für jeden Objekttyp und dessen Untertypen ist das zugehörige Verhalten abzubilden und schließlich ist (3) für jeden gewünschten Operator dessen Anwendungsbereich (= zugehörige Objekte), die erforderlichen Ein- und Ausgabeparameter sowie dessen Wirkung zu beschreiben (vgl. Dittrich (1990), S. 232).

Beispiel:
Der Objekttyp"Student" wird durch Attribute beschrieben (vgl. Alagic (1988))

```
TYPE Student   = RECORD   Matrikelnummer    STRING 7
                          Name              STRING 30
                          Semester          STRING 2
                          .....Fachbereich  STRING 3
                          .....Adress       STRING 35
                          .....Telefon      STRING 12
                 END
TYPE Studenten = ENTITY SET OF Student
```

Der Objekttyp "Student" besitzt ein bestimmtes Verhalten, das sich durch die Ober-Operatoren "Einschreiben", "Prüfungen absolvieren" und "Exmatrikulieren" und spezifizierte Sub-Operatoren beschreiben läßt.

```
TYPE Studenten = ACTIONS   Einschreiben
                           Prüfung
                           .....Exmatrikulieren
                 END
```

Bestimmte Subklassen wie Studenten der Fachrichtung Wirtschaftsinformatik, Informatik oder Betriebswirtschaftslehre durchlaufen im Rahmen des Operators "Prüfungen absolvieren" spezifische (durch die Prüfungsordnung festgelegte) Regeln, die in entsprechenden Sub-Operatoren jeweils mit ihren Eingabeparameter, Prozeduren und Ausgabeparametern beschrieben werden.

Durch anwendungsspezifische Objekte soll die Struktur, durch die Operatoren das Verhalten von Umweltobjekten in der Datenbank nachgebildet werden können. Durch die Operatoren kann eine Objektklasse vollständig und umfassend beschrieben werden. Die Implementierung des Modells muß dann sicherstellen, daß ein Zugriff auf definierte Objekte nur über die spezifizierten Operatoren möglich ist (Prinzip der Datenkapselung). Lediglich in der Phase der Objektimplementierung sind direkte Zugriffe möglich. Die bisher vorgeschlagenen Systeme erlauben es, die objektspezifischen Operatoren aus Befehlen einer Datenmanipulationssprache zu bilden und dabei nach Bedarf auch Befehle einer objektorientierten Programmiersprache (z. B. C++) einzubinden. Verhaltensmäßig orientierte Modelle bauen somit häufig auf objektorientierten Programmiersprachen auf.

Objektorientierte Operatoren	Klassenorientierte Operatoren	Prozedurorientierte Operatoren
GET DELETE DROP SEARCH	TYPE SUBTYPE CASE FOREACH	IF..THEN..ELSE DO..UNTIL BEGIN..END REPORT PROCEDURE

Abb. 4.3.2/30: Beispiele zu Operatoren einer objektorientierten Datenbenk (vgl. Alagic (1988))

Verhaltensmäßig objektorientierte Datenmodelle sind semantisch orientiert. Sie ermöglichen die Identifikation von Datenobjekten über Inhalte und spezielle Objektidentifikatoren und ersparen daher dem Nutzer die Definition von Schlüsselattributen, was weniger bei betriebswirtschaftlichen, wohl aber bei vielen technischen Anwendungen eine Erleichterung darstellt. Weiterhin bieten die Konzepte umfassende semantische Integritätsbedingungen (vgl. Dittrich (1990)).

Beispiele für verfügbare Systeme auf dem Markt sind: *Ontologic VBASE, Serviologic GEMSTONE, ONTOS*

Voll objektorientierte Datenmodelle

Voll objektorientierte Datenmodelle (behavioral object-orientation) verbinden die Eigenschaften strukturell und verhaltensmäßig objektorientierter Konzepte und unterstützen damit sowohl komplexe Objekte als auch benutzerspezifische Operatoren. Sie erlauben zusätzlich die Definition benutzerspezifischer, komplexer Datenobjekte und darauf zielender Operatoren.

Entwicklungsrichtung	Erweiterung objektorientierter Programmiersprachen um Datenbankeigenschaften	Erweiterung klassischer Datenmodelle um objektorientierte Eigenschaften
Erweiterungen	- Permanente Datenspeicherung und Archivierungsprozeduren - Transaktionsmanagement - Unterstützung von Abfragesprachen und Zugriffssprachen	- Komplexe Datentypen (Graphiken, Dokumente, abstrakte Datentypen) - Datenhierarchien und Objekthierarchien
Beispiele	STATICE (Symbolics) - LISP G-BASE (Grapheels) - LISP GEMSTONE (Serviologie) - SMALLTALK ONTOS (Ontologie) - C++	POSTGRES (Berkeley) - INGRES IRIS (HP) - ALLBASE STARDURST (IBM) EXODIES (Univ. Wisconsin) GENESIS (Univ. Texas-Austin)

Abb. 4.3.2/31: Entwicklungsrichtungen objektorientierter Datenmodelle, (vgl. Dabrowski,/ Fong/Yang (1990))

Das Gebiet der objektorientierten Datenbanken ist zur Zeit noch nicht abschließend diskutiert: Es existieren zur Zeit noch keine einheitlichen Datenstrukturen zur Abspeicherung komplexer Datenobjekte. Dabei ist eine Frage, ob diese für alternative Anwendungsfelder

(CAD/CAM; betriebswirtschaftliche Datenmodellierung; Artificial Intelligence) übergreifend abgeleitet werden können. Die Abgrenzung zur objektorientierten Programmierung ist zur Zeit noch unklar. Welche Konzepte lassen sich gemeinsam nutzen, an welchen Stellen gibt es abweichende Anforderungen? Es existieren auch noch keine Datenbanksprachkonzepte, geschweige denn eine Standardisierung analog SQL. Eine solche Datenbanksprache müßte Klassen- und Vererbungsstrukturen für den Nutzer deutlich und manipulierbar machen.

Die Vorteile objektorientierter Datenbanken liegen generell darin, daß wesentlich mehr an Semantik innerhalb der Datenbasis gehalten werden kann. Die Objektstrukturen werden von vornherein problemspezifisch ausgelegt und die darauf zulässigen Operatoren definiert. Die einzelnen Anwendungsprogramme können damit z. B. von Aufgaben der Konsistenzprüfung, der Abbildung zulässiger Operatoren entlastet werden. Doppelentwicklungen in Anwendungsprogrammen, die auf den gleichen Datenbestand zugreifen, werden vermieden. Durch die Verwendung von Typhierarchien, eindeutigen Schnittstellen und die Vererbung wird ein höheres Maß an Wiederverwendbarkeit erwartet. Die Vorteile bei komplexen Objekten und durch benutzerspezifizierbare Operatoren werden zur Zeit noch durch Performance-Nachteile und fehlende Standardisierung erkauft.

4.3.2.3. Zeitorientierte Datenmodelle

Zeitorientierte Datenmodelle (auch zeitbezogene Datenmodelle) berücksichtigen explizit zeitliche Aspekte, so zum Beispiel

- daß sich die Ausprägungen von Attributen im Laufe der Zeit ändern können,
- daß die Existenz von Objekten, Relationen oder Attributen bzw. das Interesse, sie im Datenmodell zu betrachten, auf bestimmte Zeiträume beschränkt sein kann,
- daß unterschiedliche (historische) Versionen eines Datenbestandes gehalten werden müssen.

Zeitorientierte Datenmodelle unterscheiden sich wesentlich von den herkömmlichen (statischen) Datenbanken (snapshot databases), die nur über Umwege ermöglichen, zeitliche Versionen von Datenbanken zu speichern und deren zeitliche Entwicklung nachzuvollziehen. Zwar läßt sich durch zusätzliche zeitbezogene Attribute auch in herkömmlichen Datenbanken angeben, ab wann bzw. bis wann eine Relation gültig ist, doch werden dabei nur Versionen abgespeichert, ohne daß deren zeitliche Zusammenhänge deutlich werden.

Zeitliche Aspekte sind schon bei der Datenmodellierung zu berücksichtigen. Daraus wären dann zeitorientierte Datenbanksprachen abzuleiten, die zeitliche Abfragen (z.B. "gültig bis", "gültig ab") und die Einrichtung von Zeitobjekten erlauben. Schließlich sind Fragen des internen Schemas zu klären, um die erforderlichen Datenmengen effizient abzuspeichern (vgl. Ling/Bell (1990)).

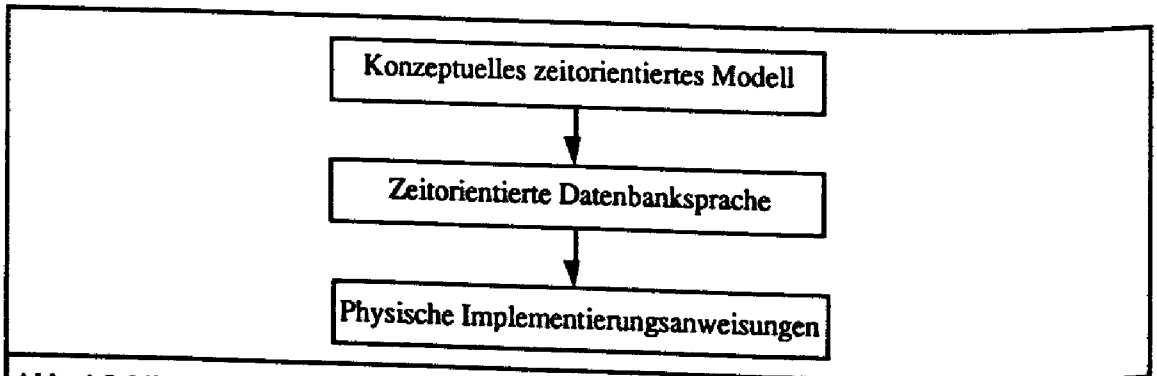


Abb. 4.3.2/32: Aufbau eines zeitorientierten Datenmodells

Datenbestände werden im Zeitablauf geändert, weil (unternehmens-)interne oder externe Veränderungen in der Realität oder in den verwendeten DV-Systemen aufgetreten sind. Häufig resultieren Änderungen z. B. aus Anforderungen des Gesetzgebers, aus einer veränderten Organisation oder DV-Technologie. Die klassischen Datenmodelle sind dann nur eingeschränkt verwendbar, da sie einen unscharf definierten "aktuellen" Stand halten, der von vielen Mitarbeitern mit unterschiedlichen zeitlichen Erkenntnisständen eingepflegt wurde.

	extern verursachte Änderungen	**intern verursachte Änderungen**
Organisatorische Ebene	> Akquisition durch Konzern	> veränderte Organisationsstrukturen
Ökonomische Ebene	> veränderte Steuergesetzgebung	> veränderte Marktanforderungen
Funktionale Ebene	> veränderte Kundenanforderungen	> veränderte Produktionsabläufe
Technologische Ebene	> veränderte Kommunikationstechnologie der Kunden	> verwendete Datenbank-Software > verwendete Speichertechnologie

Abb. 4.3.2/33: Beispiele für Änderungen im Zeitablauf

Solche Änderungen sind in der wirtschaftlichen Realität außerordentlich häufig. Das Problem liegt nicht nur darin, ältere Versionen einer Datenbank rekonstruieren zu können, sondern aus Gründen der zeitlichen Vergleichbarkeit ältere Datenbank-Versionen auf heutige Strukturen und umgekehrt heutige Versionen auf ältere Strukturen zu transformieren.

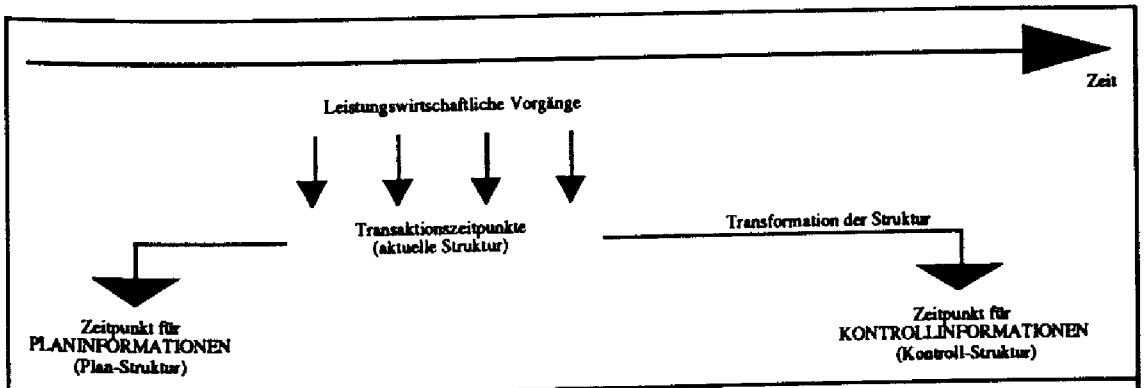


Abb. 4.3.2/34: Notwendigkeit für Strukturtransformationen

Infolgedessen dürfen ältere Datenbankversionen weder in ihrem Inhalt noch in ihrer Struktur verändert werden und es muß möglich sein, ältere in neue Versionen (und umgekehrt) zu überführen.

Für betriebliche Anwendungen sind entsprechende Datenmodelle von großem Interesse, da sich die Sachverhalte und Strukturen in Unternehmen in einem kontinuierlichen Wandel befinden und es aus Gründen der Überprüfbarkeit (z.B. Steuer- und Wirtschaftsprüfungen) und der Vergleichbarkeit (z.B. Zeitvergleiche in der Unternehmenssteuerung) notwendig ist, die Geschichte von Datenobjekten, Beziehungen und Ereignissen nachzuvollziehen. Traditionell wird der Zeitaspekt in betriebswirtschaftlichen Rechenwerken durch Periodisierung berücksichtigt; dies gilt beispielsweise für die Größen des Rechnungswesens. Gelänge es, Datenbanksysteme zu entwickeln, die periodenübergreifende betriebliche Zusammenhänge transparent machen könnten, so ergäben sich ganz neue Möglichkeiten der betrieblichen Steuerung und Disposition (vgl. Riebel (1989)).

	Mögliche Fragestellungen	Notwendige Datenelemente
Leistungs-/ Kosten-rechnung	- In welcher Zeit lassen sich welche Kosten abbauen?	- Zeitliche Vertragsmerkmale (Laufzeiten, Kündigungs-modalitäten etc.)
Finanzrechnung	- Wann werden welche Ein-/Auszahlungen erwartet?	- Zahlungsbestimmende Vertragsmerkmale (Zahlungsfristen, Zahlungs-verpflichtungen)
Bilanzierung	- Wann sind welche Erträge/Aufwendungen bzw. Einnahmen/Ausgaben zu realisieren, um einen bestimmten Gewinn zu erzielen?	- Zeitliche Dispositionszeiträume für bilanzierungsrelevante Handlungen

Abb. 4.3.2/35: Beispiele für die betriebliche Relevanz zeitorientierter Datenmodelle (vgl. Knolmayer/Bötzel/Disterer (1991))

Zumindestens theoretisch sollten in einer zeitorientierten Datenbank keine Lösch- und Update-Funktionen durchgeführt werden bzw. diese dürfen nicht zu nicht mehr rekonstruierbaren Änderungen führen. Die Geschichte eines Datenbestandes soll jederzeit nachvollziehbar sein. Die Abfragesprache muß dazu die unterschiedlichen Zeitdimensionen ansprechen können.

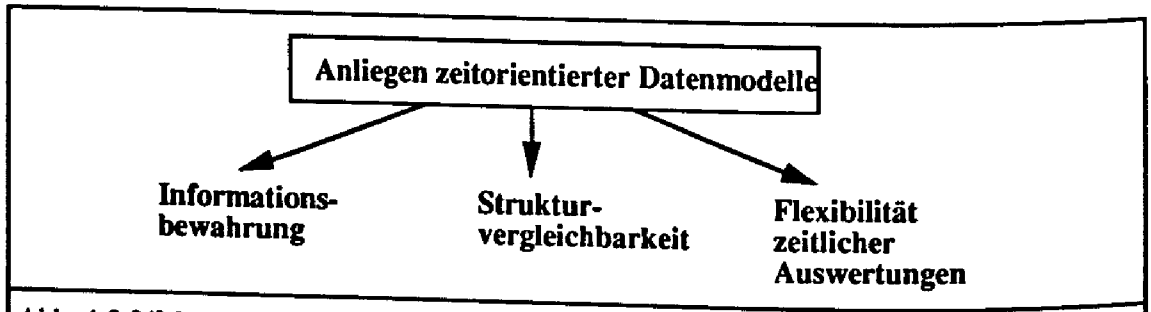


Abb. 4.3.2/36: Anliegen zeitorientierter Datenbanken

Für zeitorientierte Datenmodelle wurden bereits eigene Datenbankschemata, -systeme und -sprachen entwickelt. Praktisch relevant sind diese bei betrieblichen Anwendungssystemen, in denen es fast immer notwendig ist, die zeitliche Dimension (z. B. Berichtszeiträume, Trennung von Plan- und Ist-Zeiten) detailliert zu erfassen. Allerdings fehlt noch die entsprechende (kommerzielle) Datenbanksoftware. Zur technischen Realisierung muß noch der benötigte Speicherplatz gesenkt und die Performance gesteigert werden. Möglichkeiten bieten Speicherhierarchien, da historische Datenbestände ja nur bei Fehlern aktualisiert werden müssen und optische Massenspeicher.

Zeitorientierte Datenbanken berücksichtigen zwar die Zeit, umfassen jedoch nur eine komparativ statische, nicht jedoch eine dynamische Sicht der Datenbank. Es werden zwar unterschiedliche Zustände im Zeitablauf gespeichert, nicht jedoch die Ereignisse, die auf diese Zustände eingewirkt haben. Dynamische Datenmodelle sollten demgegenüber interne oder externe Ereignisse, die zu einem bestimmten Zeitpunkt oder in einem Zeitintervall auf die Daten einwirken, abbilden. Sie halten entsprechende Hilfsmittel (z. B. Petri-Netze) bereit, um zeitliche Integritätsbedingungen abzubilden.

Modellierung in zeitorientierten Datenmodellen

1. Zeittypen

Zeiten lassen sich grundsätzlich abbilden als Zeitpunkte oder Zeiträume. Dabei läßt sich jeder Zeitpunkt als Kombination aus einem (früheren oder späteren) Zeitpunkt und einem Zeitraum und jeder Zeitraum als Differenz zweier Zeitpunkte erfassen (Beginn, Ende). Im Rahmen zeitorientierter Datenmodelle werden folgende Zeittypen unterschieden (vgl. Knolmayer/Bötzel/Disterer (1991)):

- die Existenzzeit (Beginn, Ende) erfaßt den Zeitpunkt, zu dem eine Zustandsänderung in der Realität wirksam wird,
- die Transaktions- oder Registrierungszeit (transaction time, physical time) erfaßt den Zeitpunkt des Abspeicherns in der Datenbank,
- die Bestimmungszeit (effective time) erfaßt den Zeitpunkt, zu dem eine gewisse Beobachtung gemacht wurde,
- die Gültigkeitszeit (valid time, logical time) erfaßt den Zeitpunkt einer Zustandsänderung in der Realität unterschieden nach
  - Zeitpunkt und
  - Zeitraum (Beginn, Ende).

Beispiel:
Die Änderung eines Dauerauftrages zum 1. Oktober durch Schreiben des Kunden vom 15. August, das am 16. August in der Bank eingeht. Sie wird am 17. August in die Datenbank eingepflegt. Der Kunde erhält 10 Tage vor Wirksamwerden der Änderung ein Bestätigungsschreiben.

Zeittypen	Erläuterung	Beispiel
Existenzzeit	erfaßt den Zeitpunkt, zu dem eine Zustandsänderung in der Realität wirksam wird.	1. Oktober
Gültigkeitszeit (valid time)	erfaßt den Zeitpunkt einer Zustandsänderung in der Realität.	15. August
Bestimmungszeit (effective time)	erfaßt den Zeitpunkt, zu dem die Beobachtung gemacht wurde. (Meßzeitpunkt)	16. August
Transaktions-/ Registrierungszeit (transaction time)	erfaßt den Zeitpunkt einer Zustandsänderung in der Datenbank.	17. August
Benutzerspezifizierte Zeiten (user defined times)	erfassen beliebige Zeitpunkte, die die Benutzer der Datenbank aus Gründen der Zweckmäßigkeit einrichten.	20. September

Abb. 4.3.2/37: Zeittypen in zeitorientierten Datenbanken

Transaktionszeiten lassen sich üblicherweise aus der Systemuhr ableiten; alle anderen Zeittypen sind durch das externe Schema zu pflegen. Die Transaktionszeit wird in der Literatur auch als physische Zeit (physical time) bezeichnet, während die anderen Zeiten

logische Zeiten (logical times) genannt werden, die in der Welt des Datenbanknutzers existieren.

Benutzerspezifizierte Zeiten existieren für betriebswirtschaftliche Anwendungen in vielfacher Form. Sie sind interessant z. B. bei

- Laufzeiten von Verträgen
- Dispositionszeitpunkten wie Kündigungszeitpunkte
- Nutzungsdauern von Maschinen etc.
- Berichtszeiträume wie Umsatzperioden

(vgl. Knolmayer/Bötzel/Disterer (1991))

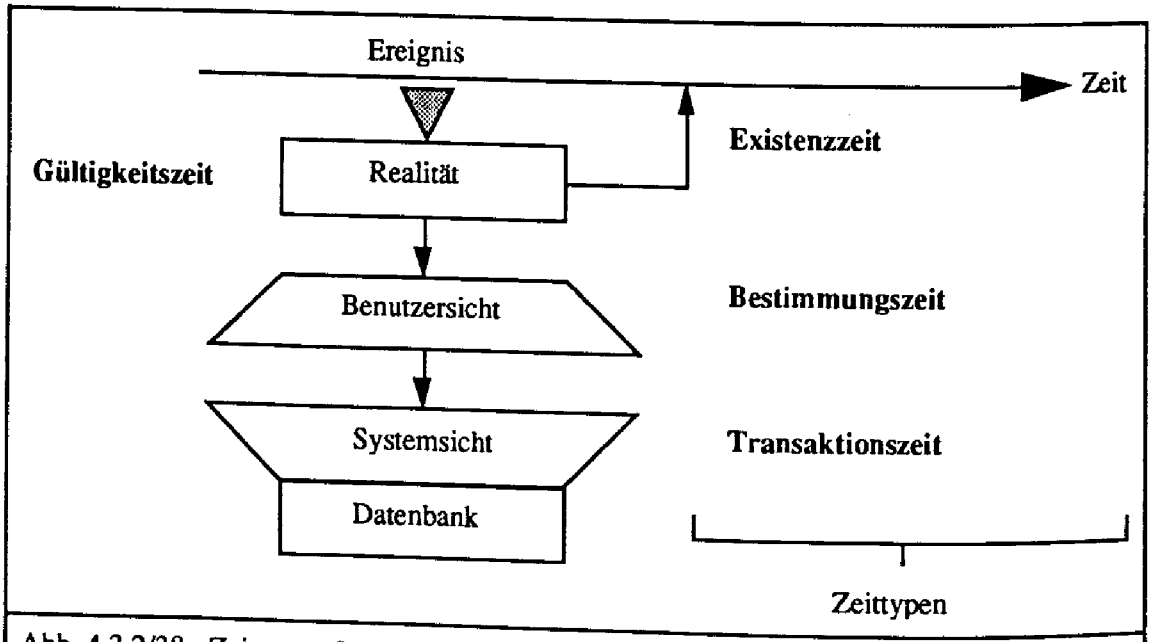


Abb. 4.3.2/38: Zeittypen für zeitorientierte Datenbanken

2. Zeitarten und Zeitmaßstab

Zeiten können sich entweder auf die Vergangenheit, die Gegenwart oder die Zukunft beziehen. Zeitorientierte Datenmodelle unterscheiden sich in ihrem konzeptuellen Schema unter anderem darin, ob sie nur Vergangenheitszeiten, Vergangenheitszeiten und Gegenwartszeiten oder auch Zukunftszeiten betrachten.

Zeitarten	Bezeichnung	Beispiel
Vergangenheitszeiten	Ist-Zeiten	gestrige Maschinenbelegung
Gegenwartszeiten	Aktuelle Zeit	aktuelle Maschinenbelegung
Zukunftszeiten	Planungszeiten Prognosezeiten	Maschinenbelegungsplanung

Abb. 4.3.2/39: Zeitarten

Zukunftsorientierte Zeiten werden in Planungs-, Prognose- und Reservierungssystemen verwendet, Gegenwarts- und Vergangenheitszeiten in Abrechnungs-/Administrations- sowie in Berichtssystemen. Zukunftszeiten lassen sich unterteilen in Prognose- und Planungszeiten. Planungszeiten sind die Zeiten, für die Aktionen gedanklich antizipiert werden, während Prognosezeiten die Zeiten umfassen, für die Auswirkungen von Aktionen erfaßt werden. Nur Gegenwarts- und Vergangenheitszeiten sind im strengen Sinne empirische Zeiten. Damit

sind Gültigkeitszeiten nur für vergangene oder aktuelle Ereignisse bestimmbar, während Prognose- oder Plandaten immer zu benutzerspezifischen Zeiten gehören.

Generell gibt es zwei Arten der zeitlichen Bezugnahme: Entweder wird ein festes Referenzereignis (z. B. bestimmtes Datum) verwendet oder man bezieht sich auf eine implizite Referenz („heute", „jetzt"). Feste Referenzen ermöglichen es, mehrere Zustände zeitlich parallel abzubilden und zu unterscheiden.

Zeiten lassen sich als Zeitpunkte oder Zeitintervalle abbilden.

	Beginn	**Ende**
Zeitpunktbetrachtung	Startzeitpunkt	Endezeitpunkt
Intervallbetrachtung	Startzeitpunkt	Startzeitpunkt + Gültigkeitsintervall

Abb. 4.3.2/40: Zeitpunkte und Zeitintervalle

Die notwendige Feinheit des Zeitmaßstabs ist problemabhängig: So ist beispielsweise bei den Auszahlungen eines Geldautomaten die Erfassung mit einer Minuten-Genauigkeit anzustreben; bei der Arbeitszeiterfassung der Mitarbeiter mögen Zehnminutenintervalle ausreichen, für die Verzinsung von Kontoständen Tage, während für die Kostenerfassung im Rechnungswesen Monate genügen können.

3. Zeitabhängige Elemente

In Bezug auf das zeitliche Verhalten sind drei Typen von Datenelementen (Objekte, Beziehungen, Attribute) zu unterscheiden: Zeitkonstante Elemente ändern sich in ihrer Existenz und Struktur nicht. Im strengen Sinne trifft dieses nur für Vergangenheitsaussagen zu, aber auch bestimmte geographische oder strukturelle Aussagen sind zeitlich als konstant anzusehen (Beispiel: Vorname eines Mitarbeiters). Zeitabhängige Elemente können sich in ihrer Existenz und Struktur ändern.

Beispiel:
Ein Mitarbeiter wird eingestellt, zur Führungskraft befördert und in eine höhere Gehaltsklasse umgruppiert.

Zeitobjekte schließlich kennzeichnen Zeitpunkte oder Zeiträume.

Die Zeitdimension kann die Existenz oder die Attributausprägungen von Objekten, Beziehungen oder Ereignissen auf jeder der drei Abstraktionsebenen betreffen.

	Existenz	**Attributstruktur**	**Attributwerte**
Objekt	Neues Element bzw. neuer Typ	Neues Attribut bzw. neuer Attributtyp	Veränderter Wert bzw. veränderte Wertedomäne
Beziehung			
Ereignis			

Abb. 4.3.2./41: Zeitabhängige Elemente

Auf der Realebene wird die Existenz oder die Attributausprägung von Objekten, Beziehungen und Ereignissen sich häufig ändern. Hingegen ist auf der Typebene ein Datenmodell

zeitlich recht stabil, d. h. Attributstrukturen und Existenz bestimmter Objekt-, Beziehungs- und Ereignistypen werden sich selten ändern.

Beispiel:
Es ist ein Merkmal eines Unternehmens, daß es über Mitarbeiter verfügt. Die Attributstrukturen des Datenobjektes „Mitarbeiter" werden sich von Zeit zu Zeit z. B. durch neue Steuergesetze ändern. Recht häufig werden sich demgegenüber die Gehälter der Mitarbeiter und ähnliche Attributwerte ändern.

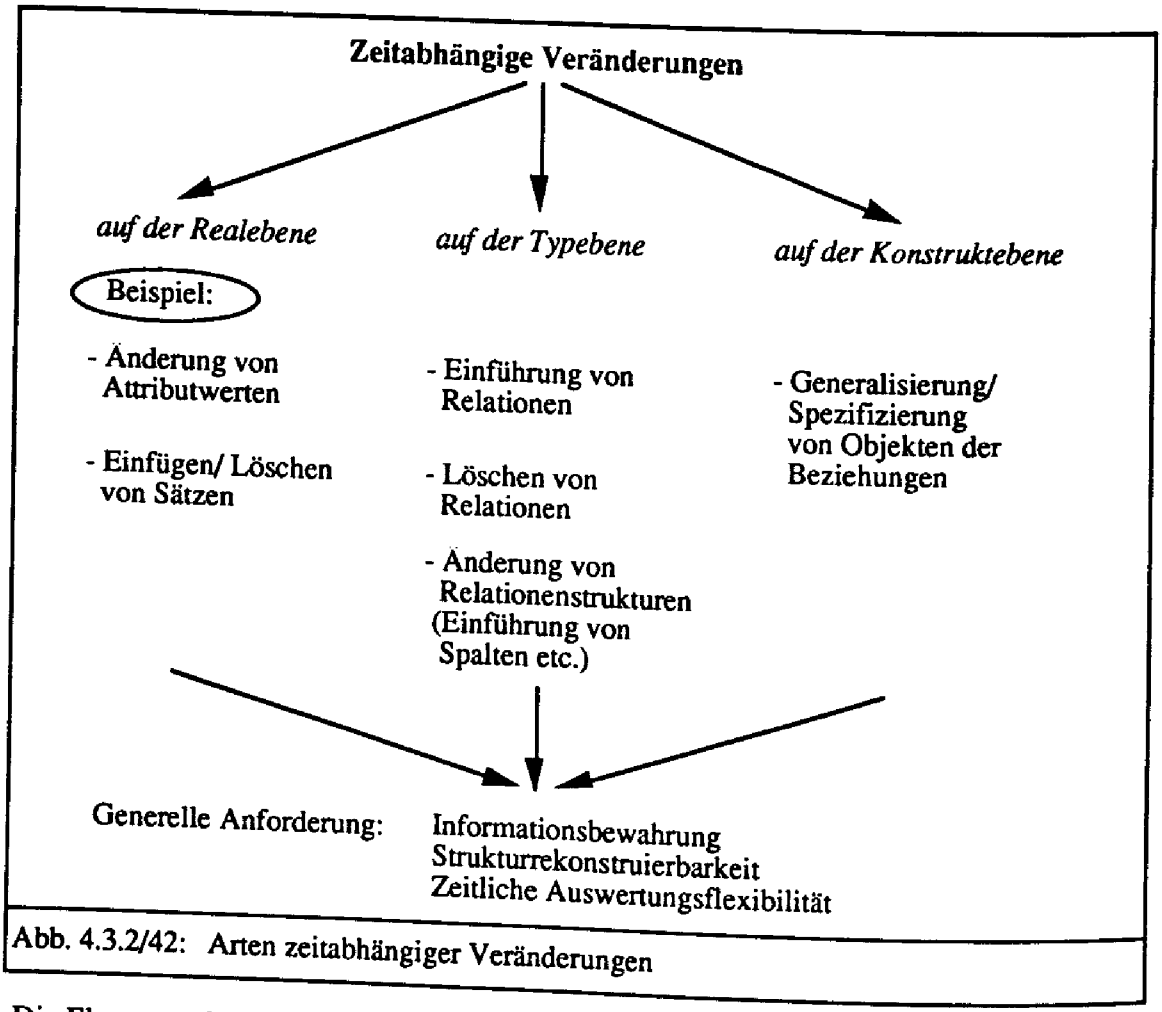


Abb. 4.3.2/42: Arten zeitabhängiger Veränderungen

Die Elemente einer Datenbank werden durch Ereignisse verändert, d. h. durch Transaktionen, Operationen oder Aktionen von Menschen oder Programmen, die die Datenbank benutzen. Existenzändernde Ereignisse verändern die Struktur des Datenmodells, d. h. die resultierenden Zustände unterscheiden sich von den vorgelagerten durch neue Objekte, Beziehungen oder Attribute bzw. durch deren Wegfall. Zustandsändernde Ereignisse haben keine strukturellen Auswirkungen, d. h. die neuen Zustände besitzen nur andere Attributausprägungen bei unveränderter Struktur des Datenmodells.

Ereignisse können stetig oder diskret sein. Zeitlich stetige Ereignisse generieren zeitlich stetige Zustände *(Beispiel: Wetterereignisse und Temperatur)*; diskrete Ereignisse sind folglich zeitlich diskrete Zustände *(Beispiel: Kauf und Lagerbestand)*.

Ereignisse können in der Datenbank für die Vergangenheit und Gegenwart erfaßt und für die Zukunft prognostiziert werden. Beispielsweise läßt sich bei Verträgen mit definierter Laufzeit das Ereignis „Auslaufen des Vertrages" und das Ende der Gültigkeitszeit vorhersagen.

4. Zeitfolgen (Geschichte) eines Datenbestandes

Ein Datenbestand durchläuft eine zeitliche Geschichte, die durch die Entwicklung der einzelnen Objekte, Beziehungen und Attribute beschrieben werden kann. Man unterscheidet folgende Typen von Zeitfolgen (vgl. Härder (1984): Bei zustandserhaltenden Zeitfolgen behält eine Aussage so lange ihre Gültigkeit, bis ein neues Ereignis registriert wird (*Beispiel: Zuordnung eines Mitarbeiters zu einer Abteilung*). Demgegenüber wirken bei zustandsveränderlichen Zeitfolgen die Ereignisse kontinuierlich auf die Objekte ein, d. h. die Zustände ändern sich ständig (*Beispiel: Temperatur in einem Produktionsprozeß*). Ableitbare Zeitfolgen sind dadurch gekennzeichnet, daß der Zustand eines Objektes sich zwischen den Ereignissen endogen entsprechend einer definierten Vorschrift weiterentwickelt (*Beispiel: Verzinsung eines Kontostandes*). Bei ereignisorientierten Zeitfolgen wirken die Ereignisse diskret auf die Objekte ein, die Objekte der Datenbank haben zwischen den Ereignissen keinen definierten Zustand (*Beispiel: Belegung einer Maschine*).

Geschichte (Zeitfolgen)	Ereignis	Zustand
Zustandserhaltende Zeitfolgen	diskret	diskret
Beispiel	*Kauf*	*Lagerbestand eines Produktes*
Ableitbare Zeitfolgen	diskret	kontinuierlich
Beispiel	*Einzahlung/Auszahlung auf einem Bankkonto*	*Kontostand (durch kontinuierliche Verzinsung)*
Ereignisorientierte Zeitfolgen	diskret	keiner
Beispiel	*Maschinenbelegung*	
Zustandsveränderliche Zeitfolgen	kontinuierlich	kontinuierlich
Beispiel	*Prozeßbedingungen*	*Temperatur in einem Reaktor*

Abb. 4.3.2/43: Typen der Geschichte einer Datenbank

Diese Geschichtsarten kommen bei der zeitlichen Entwicklung von Datensätzen und/oder Relationen selten in Reinform vor. So ist z. B. die Entwicklung eines Kontos eine Mischung aus ereignisorientierter (Ein- und Auszahlungen) und ableitbarer (Zinsberechnung) Geschichte (vgl. Härder (1984)).

Die Geschichte eines Datenbestandes soll nach bestimmten Grundsätzen in einem zeitorientierten Datenmodell erfaßt werden.

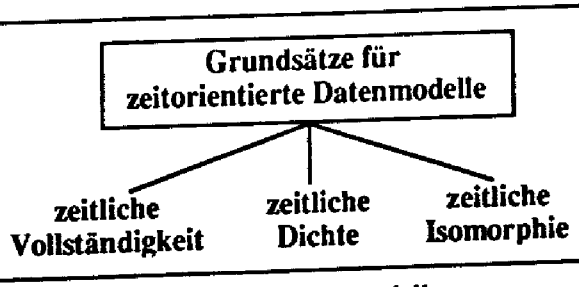


Abb. 4.3.2/44: Grundsätze für zeitorientierte Datenmodelle

Zeitliche Vollständigkeit besagt, daß die Datenbank durch eine Folge von Zuständen vollständig die zeitliche Entwicklung der abgebildeten Realität beschreibt. Unter zeitlicher Dichte wird verstanden, daß der Zustand der Datenbank zu jedem Zeitpunkt aus einer vollständi-

gen zeitlichen Geschichte (Folge von Ereignissen) nachvollzogen werden kann. Zeitliche Isomorphie heißt, daß die Abbildung der zeitlichen Zustände in der Datenbank in Geschwindigkeit und Reihenfolge den zeitlichen Zuständen in der Realität folgt.

Arten zeitorientierter Datenmodelle

Da sich zeitorientierte Datenmodelle zu einem Hauptgebiet der Datenbankforschung entwickelt haben, existieren eine Vielzahl kaum mehr zu überschauender Ansätze (vgl. Ling/Bell (1990)). Diese lassen sich danach typisieren

-> wieviele und welche Zeitdimensionen betrachtet werden?
-> welche Methode der Zeitstempelung vorgenommen wird?
-> welche Methode zur Abspeicherung der zeitabhängigen Werte verwendet wird?
-> welche Ebene des Datenbankentwurfs betrachtet wird?

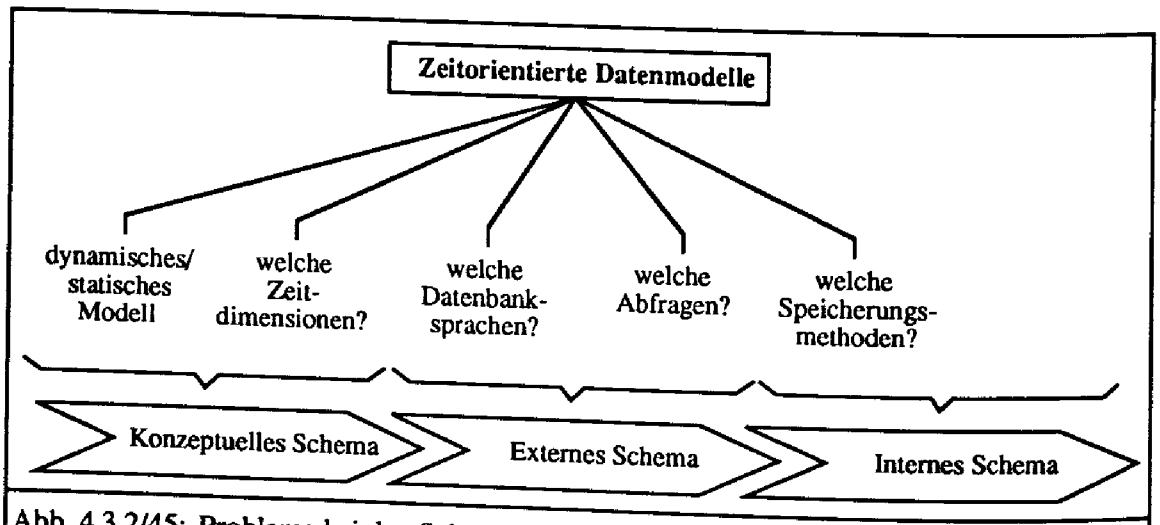


Abb. 4.3.2/45: Probleme bei den Schemata zeitorientierter Datenmodelle

Im folgenden wird in Anlehnung an Snodgrass nach der Art und der Zahl der berücksichtigten Zeiten typisiert (vgl. Snodgrass (1987), Snodgrass/Ahn (1986)).

Zeitart	Zeitstempel	Datenbanktyp	
Transaktionszeit	Anfang Ende	Rückschau-Datenbank	Temporale Datenbank
Gültigkeitszeit	Anfang Ende	Historische Datenbank	
Benutzerspezifizierte Zeit	problemabhängig		

Abb. 4.3.2/46: Zeitorientierte Datenbanktypen nach Zeitarten

In Rückschau-Datenbanken wird nur die (physische) Registrierungszeit eines Datums berücksichtigt, während Historische Datenbanken die (logische) Gültigkeitszeit erfassen. Temporale Datenbanken verbinden beide Ansätze.

Rückschau-Datenbanken

In Rückschau-Datenbanken (rollback databases) wird einem Datenbankeintrag automatisch die Registrierungszeit beigefügt. Gespeichert wird jeweils eine neue historische Version. Somit kann eine Datenbank jederzeit so ausgewertet werden, wie sie sich zu einem

beliebigen Zeitpunkt dargestellt hat - die Datenbank erlaubt eine zeitliche Rückblende. Beispielsweise wird im relationalen Modell jede Tabelle mit einer Registrierungszeit versehen. Es entsteht eine Folge zeitlich gestempelter, statischer Tabellen.

Abfragen sind auf jede historische Version zulässig; Änderungen beziehen sich hingegen immer auf den letzten Zustand der Datenbank.

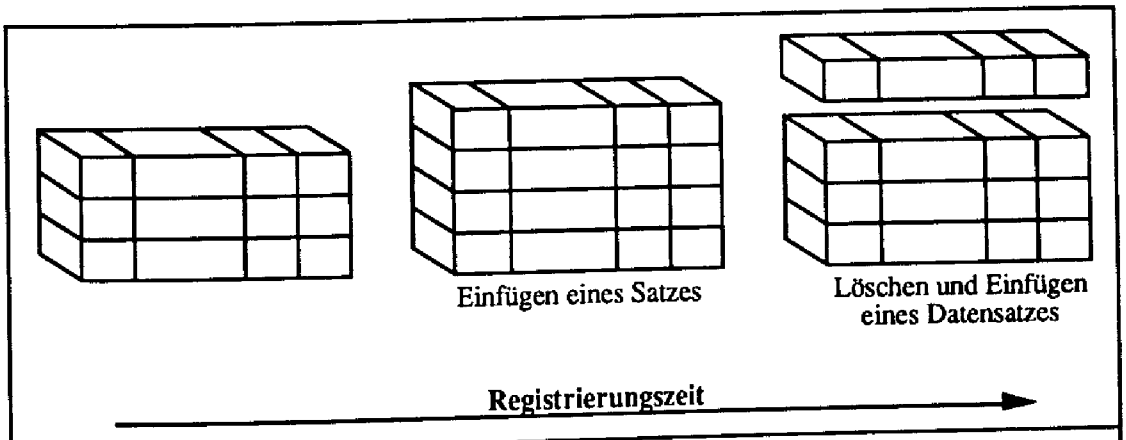


Abb. 4.3.2/47: Entwicklung einer Relation in einer "rollback database" (Snodgrass (1987), S. 251)

Da Daten in "rollback databases" niemals überschrieben werden, sind jederzeit Prüfungen von technischen oder inhaltlichen Fehlern (z.B. Management Audits) möglich (vgl. Knolmayer (1989)).

Historische Datenbanken

In sogenannten Historischen Datenbanken (historical databases) wird statt der technischen Registrierungszeit die Gültigkeitszeit einer Aussage in der Realität verwendet. Es wird nicht die Geschichte der Datenbank, sondern jene der Realität unter Nutzung des besten Kenntnisstandes ("history as it is best known") gespeichert. Abfragen auf historische Versionen der Datenbank sind somit nicht möglich; Fehler werden durch Überschreibungen in der aktuellen Datenbank korrigiert (vgl. Snodgrass/Ahn (1986b)).

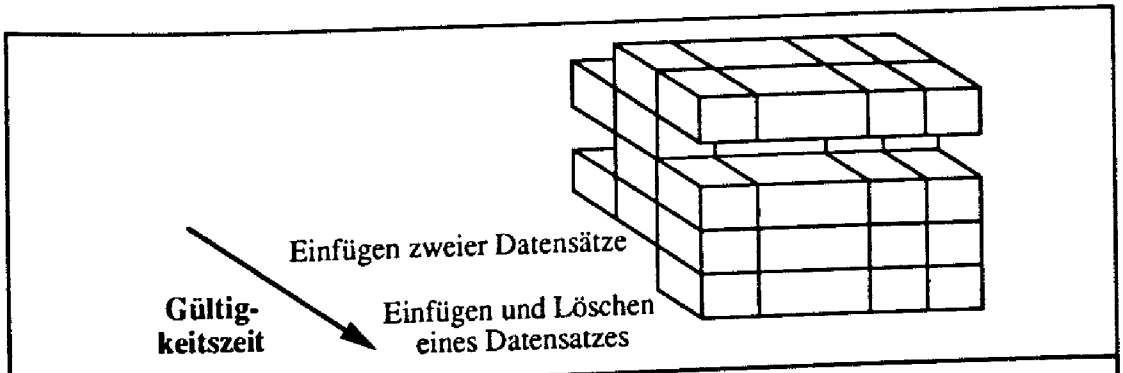


Abb. 4.3.2/48: Relation einer Historischen Datenbank (Snodgrass (1987), S. 251)

Während "rollback databases" eine Folge statischer Datenbankversionen halten, speichern historische Datenbanken die Geschichte der Realität in einer, kontinuierlich fortgeschriebenen und ggf. auch korrigierten Version. Es ist damit möglich, die Geschichte nachzuvollziehen, nicht aber etwaige Fehler und Korrekturen in der Datenbank (vgl. Snodgrass (1987), Snodgrass/Ahn (1986b)).

Während sich Registrierungszeiten aus DV-Systemzeiten ableiten lassen, bedarf es für Gültigkeitszeiten einer spezifischen Modellierung der Zeit und spezieller Datenbankoperationen. Die Vorschläge für historische Datenbanken lassen sich danach unterscheiden, ob die Gültigkeitszeit (VALID TIME FROM & VALID TIME TO) einzelner Attribute oder ganzer Sätze (Tupel) gespeichert werden (vgl. Ling/Bell (1990)).

Art der Zeitstempelung	**mit Tupelstempelung**		**mit Attributstempelung**	
Zeitarten	Vergangenheitszeiten	Vergangenheits-/Gegenwartszeiten	Vergangenheitszeiten	Vergangenheits-/Gegenwartszeiten
Autoren	Jones/Mason (1980) Clifford/Warren (1983)	Bradley (1986) Lum/ Dadam/ Erba etc. (1984)	Klopprogge/ Lockemann (1983)	Clifford (1985)

Abb.4.3.2/49: Ausgewählte Vorschläge historischer Datenbanken (vgl. Ling/Bell (1990))

Tupelstempelung benötigt weniger Implementierungsaufwand, ist jedoch bei zeitlichen Datenbankabfragen und -manipulationen aufwendiger.

Clifford/Warren (1983) schlagen beispielsweise vor, die zeitlich aufeinanderfolgenden Zustände durch jeweils neue Relationen mit Zustandsattributen zu beschreiben. Ihre Idee ist es, jeden Satz der Datenbank mit einem "dualen Gültigkeitsattribut" zu versehen, das angibt, ob ein bestimmter Satz in einem bestimmten Zustand gültig ist oder nicht. Es ist deutlich, daß eine solche "Totaldatenbank" sehr redundant ist und hohe Speicherkapazitäten erfordert.

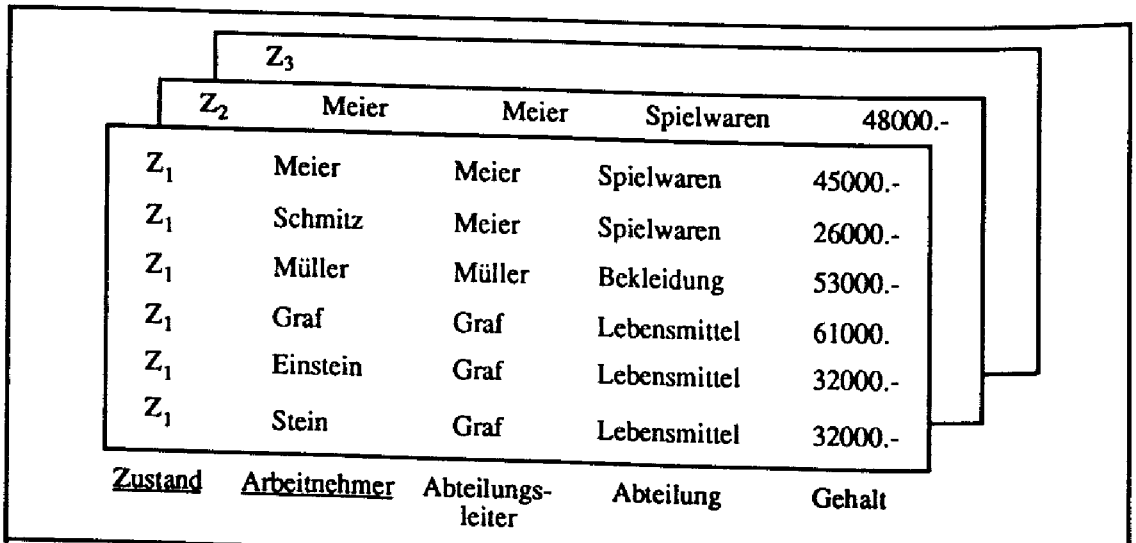


Abb.4.3.2/50: Zustandsattribute in der Relation Arbeitnehmer_Relation (vgl. Clifford/ Warren (1983), S. 222)

Temporale Datenbanken

Temporale Datenbanken (temporal databases) berücksichtigen sowohl die physische Registrierungszeit als auch die logische Gültigkeitszeit. Sie benötigen vier Dimensionen, um eine statische relationale Datenbank um die beiden Zeitdimensionen zu erweitern. Eine

temporale "Tabelle" läßt sich auffassen als eine Folge historischer Zustände, wobei jeder Zustand eine komplette Historie umfaßt.

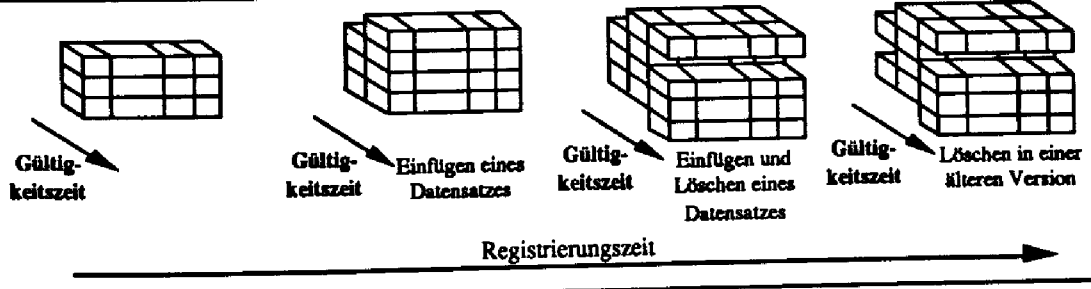


Abb. 4.3.2/51: Relationen einer temporalen Datenbank (Snodgrass (1987), S. 252)

Mit temporalen Datenbanken läßt sich sowohl die Entwicklung des Datenbestandes (über die Transaktionszeiten) als auch die Historie der Realität (über die Gültigkeitszeiten) nachvollziehen. Es läßt sich nach dem Zeitbezug und der Zeitstempelung weiter differenzieren.

Zeitbezug	**Vergangenheit**		**Vergangenheit & Gegenwart & Zukunft**	
Zeit-stempelung	Tupel	Attribute	Tupel	Attribute
Autoren	Abbod/Brown/ Noble (1987)	Tansel (1986)	Snodgrass/Ahn (1986) Ben-Zvi (1982)	

Abb. 4.3.2/52: Ausgewählte Vorschläge temporaler Datenbanken (vgl. Ling/Bell (1990))

Zeitorientierte Datenbanken mit benutzerspezifizierten Zeiten

Neben der technischen Transaktionszeit und der realitätsbezogenen Gültigkeitszeit gibt es weitere Zeiten, die aus dem Problemverständnis des Benutzers resultieren. Nicht alle zeitorientierten Datenmodelle unterstützen neben der logischen Gültigkeitszeit und der physischen Transaktionszeit weitere, vom Benutzer spezifizierte Zeiten. Generell lassen sich solche Zeiten als normale Attribute integrieren, doch sind spezielle Abfragemöglichkeiten notwendig. Benutzerspezifische Zeiten können sich z. B. ergeben aus rechtlichen Attributen des zugrundeliegenden Ereignisses oder aus wirtschaftlichen Eigenschaften (z. B. Nutzungsdauer).

Ursachen benutzerspezifischer Zeiten	**Beispiele**
Rechtliche Attribute	Kundenauftrag: - rechtliche Bindungsdauer - späteste Erfüllungszeit
Wirtschaftliche Attribute	- Bereitstellungszeit - Plan-/Vorschau-/Ist-Zeiten
Persönliche Attribute	- Wiedervorlagezeiten

Abb. 4.3.2/53: Ursachen benutzerspezifischer Zeiten

Der Ansatz von Ariav verwendet spezielle Attribute (Time Stamp Attributes) für alle Merkmale einer Relation, für die eine zeitlich orientierte Selektion notwendig ist (vgl. Ariav

(1983, 1986)): Hier werden jeweils die Tupel einer Relation mit der Transaktionszeit gestempelt.

Methoden der Zeitstempelung

Bei der Zeitstempelung werden die Elemente eines Datenbestandes mit den Zeitmerkmalen versehen. Es werden folgende Methoden unterschieden:

Bei der relationenbezogenen Zeitstempelung wird das "normale" relationale Modell um eine zeitliche Dimension ergänzt; gesamte Relationen werden mit einer Zeitdimension gestempelt (vgl. Ben Zvi 1982). Die tupelbezogene Zeitstempelung arbeitet mit der Zeitstempelung von Sätzen. Lum etc. schlagen vor, die jeweils aktuellen Sätze in einer Relation zu halten und mit den historischen Sätzen zu verketten. Die Zeitstempelung von Tupeln ist relativ einfach zu realisieren, doch ist der Aufbau einer eindeutigen Datenmanipulationssprache schwierig (vgl. Lum /Dadam/Erba (1984)).

Die attributbezogene Zeitstempelung arbeitet mit zeitabhängigen Attributen und unterscheidet zeitkonstante von zeitabhängigen Attribute sowie Zeitattributen. Die Zeitstempelung von Attributen führt zu Tupelstrukturen, die die erste Normalform nicht erfüllen (NF2). Die Implementierung ist aufwendiger, jedoch ist die Logik für Manipulationssprachen einfacher (vgl. Clifford (1985)).

Mitarbeiter-Nr.	Name	Abteilung	Gehalt	Gültigkeitszeit
M101	Meier	Forschung	50000	1.10.1990
M101	Meier	Forschung	55000	15.3.1991
S573	Schmidt	Auftragsbearbeitung	47800	10.11.1990

(a) Tupelbezogene Zeitstempelung mit einer logischen Zeit (Gültigkeitszeit)

Mitarbeiter-Nr.	Name	Abteilung	Gehalt	Gültigkeitszeit	Transaktions-zeit
M101	Meier	Forschung	50000	1.10.1990	22.9.1990
M101	Meier	Forschung	55000	15.3.1991	15.3.1991
S573	Schmidt	Auftragsbearbeitung	47800	10.11.1990	5.11.1990

(b) Tupelbezogene Zeitstempelung mit einer logischen (Gültigkeitszeit) und einer physischen Zeit (Transaktionszeit)

Mitarbeiter-Nr.	Name	Abteilung		Gehalt	
M101	Meier	Forschung	(1.10.1990)	50000	(1.10.1990)
M101	Meier	Forschung	(1.10.1990)	55000	(15.3.1991)
S573	Schmidt	Auftragsbearbeitung	(10.11.1990)	47800	(10.11.1990)

(c) Attributbezogene Zeitstempelung mit einer logischen Zeit (Gültigkeitszeit)

Mitarbeiter-Nr.	Name	Abteilung	Gehalt
M101	Meier	(Forschung, 1.10.1990)	(50000, 1.10.1990), (55000, 15.3.1991)
S573	Schmidt	(Auftragsbearbeitung, 10.11.1990)	(47800, 10.11.1990)

(d) Wertbezogene Zeitstempelung

Abb. 4.3.2/54: Beispiele zu den Methoden der Zeitstempelung (vgl. Ling/Bell (1990))

Methoden zur zeitlichen Versionenspeicherung

Um eine zeitorientierte Datenbank zu implementieren, ist nach effizienten Speicherungsmethoden zu suchen. Als Methoden werden neben der Komplett- die Veränderungsspeicherung diskutiert (vgl. Dadam/Lum/Werner (1984)).

	Kennzeichen	Vor-/Nachteile
Komplett-speicherung	Jede Version der Datenbank wird vollständig gespeichert.	+ Rechenzeit/Zugriffsverhalten - Speicherplatzbedarf
Veränderungs-speicherung	Es wird nur die Veränderung der aktuellen Version gegenüber früheren Versionen gespeichert.	- Rechenzeit/Zugriffsverhalten, da jeder Zugriff erst die Berechnung der aktuellen Komplettversion der Datenbank bedingt + Speicherplatzbedarf

Abb. 4.3.2/55: Grundformen der Versionenspeicherung

Die Veränderungsspeicherung bedingt die Komplettspeicherung einer Version: Entweder wird dann vorwärtsschreitend die Veränderung zur älteren Komplettversion ausgewiesen oder es werden rückwärtsschreitend ältere Versionen aus einer neueren Komplettspeicherung rückgerechnet.

	Komplett-speicherung	Veränderungsspeicherung (Referenzdatenbank)
Vorwärtsschreitende Veränderungsspeicherung	älteste Version	zu ältester Version
		zu vorhergehender Version
Rückwärtsschreitende Veränderungsspeicherung	neueste Version	zu neuester Version
		zu unmittelbar nachgelagerter Version

Abb. 4.3.2/56: Formen der Veränderungsspeicherung (vgl. Dadam/Lum/Werner (1984))

Interessant sind die Formen, die auf einer unveränderten Basisversion (entweder der ältesten Version oder der unmittelbar nachgelagerten Version) aufbauen, da dort jede Veränderung nur einmal berechnet werden muß. Natürlich sind auch Mischformen denkbar, bei denen als Referenz eine „mittelalte" Datenbank gewählt wird und eine Kombination aus vorwärts- und rückwärtsschreitender Veränderungsspeicherung erfolgt. Insgesamt günstigste Form ist eine rückwärtsschreitende Veränderungsspeicherung, die auf der unmittelbar nachgelagerten Version aufbaut (vgl. Dadam/Lum/Wagner (1984)).

Der große Speicherbedarf zeitorientierter Datenbanken läßt sich nur mit neuester Speichertechnolgie (optische Speicher etc.) und intelligenten Speicherorganisationsformen bewältigen. Das interne Schema muß die Verknüpfung der einzelnen historischen Sätze sowie die Speicherstrukturierung unter Berücksichtigung zeitorientierter Abfragestrukturen leisten. Die existierenden Implementierungsvorschläge lassen sich in Partialansätze (die statische Datenbanksysteme zeitorientiert erweitern) und Totalansätze (die neue Datenbanksysteme entwickeln) unterscheiden (vgl. Ling/Bell (1990)).

Basismodell	Autoren	Sprache	Basissprache
hierarchisch	Schiel (1983)	x	
relational	Ariav (1986)	x	SQL
	Ben Zvi	x	
	Snodgrass/Ahn (1986, 87)	TQUEL	QUEL(Ingres)
Objekt-Beziehung	Klopprogge/Lockemann (1983)	TERM	PASCAL
Objektorientiert	Clifford/Croker (1988)		

Abb.4.3.2/57: Ausgewählte zeitorientierte Datenmodelle (vgl. Ling/Bell(1990) und Knolmayer/ Bötzel/ Disterer (1991))

4.3.3. Beurteilungskriterien für Datenmodelle

Datenmodelle lassen sich nicht unabhängig von der zu lösenden Aufgabe und den zur Verfügung stehenden Hilfsmitteln beurteilen.

Kriterien	Subkriterium	Gesichtspunkt
Mächtigkeit	Semantisches Datenmodell	Verfügbarkeit semantischer Strukturen - Unterstützung von semantischen Konstruktionsoperatoren
	Objektmächtigkeit	Abbildbarkeit heterogener, auch komplexer Objekte (Dokumente, Graphiken)
	Versionenmächtigkeit (Zeitliche Dimension)	Verfügbarkeit physischer und logischer Zeiten Definierbarkeit benutzerspezifischer Zeittypen
	Integritätsbedingungen	Abbildbarkeit von logischen Integritätsbedingungen/Prüfprozeduren - komplexe statische Bedingungen hinsichtlich der Attribute und deren Domänen [data abstraction] - komplexe Transaktionen incl. logischen Abhängigkeiten [procedural abstraction]
Datenmodellierungssprache	Umfang	Datenstrukturkonzept Datendefinitionssprache
	Komfort	Graphische Unterstützung - Einfachheit und Erlernbarkeit (Präzision und Eindeutigkeit) - Ausrichtung auf Sprach- und Denkgewohnheiten der Benutzertypen
	Recherchemächtigkeit	Semantische Navigation = d. h. Nutzer kann Datenmodell in unbegrenzter Schachtelungstiefe erkunden und frei navigieren
	Benutzerschnittstelle	Datendefinitionssprache? Datenmanipulationssprache vorhanden?
	Entwicklungswerkzeuge	Transaktionsorientierte Programmiersprache? CASE-Instrumente
Effizienz	DV-Effizienz	Läßt sich das Datenmodell effizient hinsichtlich Speicherplatz, Laufzeit, Zugriff abbilden?

Abb. 4.3.3/1: Beurteilungskriterien für Datenmodelle

5. Datenschemabildung

5.1. Kennzeichnung

Aufgabe der Datenschemabildung ist es, die vorliegenden Datenstrukturen formal logisch eindeutig und redundanzfrei zu beschreiben, um diese dann in einem Datenbanksystem zu implementieren. Ziel ist es, vorhandene Inkonsistenzen zu beseitigen, um Ziele wie

- Speicherredundanz
- Laufzeitverhalten
- Änderungsaufwand im späteren DV-System zu optimieren.

Bei der Datenschemabildung sind zwei Vorgehensweisen zu unterscheiden. Das formal-orientierte Vorgehen setzt an den logischen Strukturen des Datenmodells an. Mit mathematischen Operationen wird das logische Datenmodell formal vereinfacht, um die Ziele der Schemabildung (Redundanzminimierung, logische Eindeutigkeit) zu erreichen.

	Ziele	**Vorgehen**
formal-orientiert	-> Redundanzminimierung -> Vermeidung von Zugriffs-anomalien	-> Logisch-mathematische Operationen
materiell-orientiert	-> schnelle Zugriffszeiten -> geringer Änderungsaufwand -> Speicherplatz	-> Analyse der Kommunikationsfrequenzen bestimmter externer „Views" -> Nutzung der physikalisch-technischen Eigenschaften der Speicherhierarchie

Abb. 5.1/1: Vorgehensweisen der Datenschemabildung

Die materiell-orientierte Vorgehensweise nutzt demgegenüber Eigenschaften des logischen Modells und des externen Schemas. Dazu gehören organisatorische oder nutzerspezifische Eigenschaften aber auch Spezifika der verwendeten Informationssystemarchitektur, d. h. der zu verwendenden Hardware/Software. Beispielsweise wird geprüft:

-> ob sich bestimmte Attributausprägungen jederzeit aus anderen Attributen berechnen lassen und daher im (statischen) Datenschema entbehrlich sind, jedoch Anforderungen an das (dynamische) Schema stellen.
-> ob bestimmte Zeitstempelungen durch die Datenbanksoftware besonders gut unterstützt werden.

Aus der Sicht der Wirtschaftsinformatik werden DV-technische Ziele von ökonomischen Überlegungen dominiert. Dabei werden neben den Betriebskosten für Rechenzeit und Speichernutzung auch die Investitionsauszahlungen für die Systementwicklung betrachtet.

Bei der Schemabildung sind zu unterscheiden

-> die Strukturzerlegung
-> die Struktursynthese

Bei der Strukturzerlegung wird ein gegebenes Datenmodell nach formalen und/oder materiellen Kriterien in seine Bestandteile zerteilt. Bekannteste Methode der Strukturzerlegung ist die Normalformenlehre nach Codd (vgl. Codd (1970)). Obwohl sie ursprünglich eng mit

dem relationalen Datenmodell verbunden ist, kann sie grundsätzlich als eigenständige, auch für andere Modelle anwendbare Methode der Schemabildung gelten.

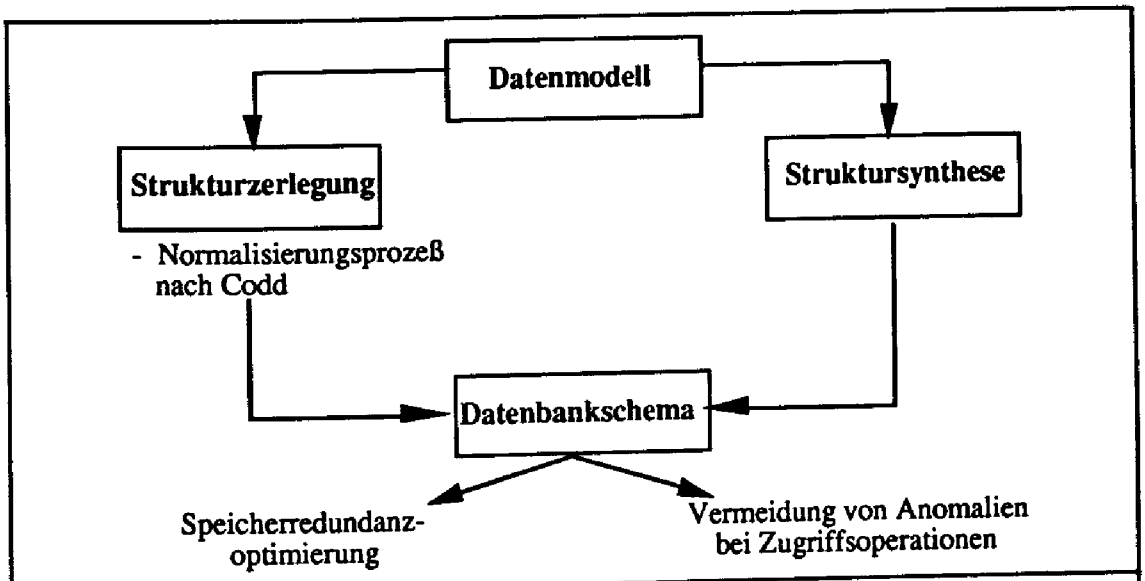


Abb. 5.1/2: Strukturzerlegung und Struktursynthese

Bei der Strukturanalyse liegen die betriebswirtschaftlichen Objekte und Zusammenhänge in einem detaillierten Datenmodell in elementarer Form vor. Diese werden dann in einem BOTTOM UP-Vorgehen zu einem funktionsspezifischen Datenschema verdichtet.

5.2. Aufgaben

5.2.1. Statische Schema-Bildung

5.2.1.1. Strukturzerlegung nach der Normalformenlehre

Bei der Strukturzerlegung wird das Datenmodell mit der an realen Informationsobjekten und Beziehungen orientierten Struktur logisch-mathematisch in einem „Top down"-Vorgehen in seine Attribute zerlegt. Auf der Basis der unnormalisierten Ausgangsrelationen werden sogenannte Normalformen gebildet, die redundanzminimal, weniger komplex und transparenter sind.

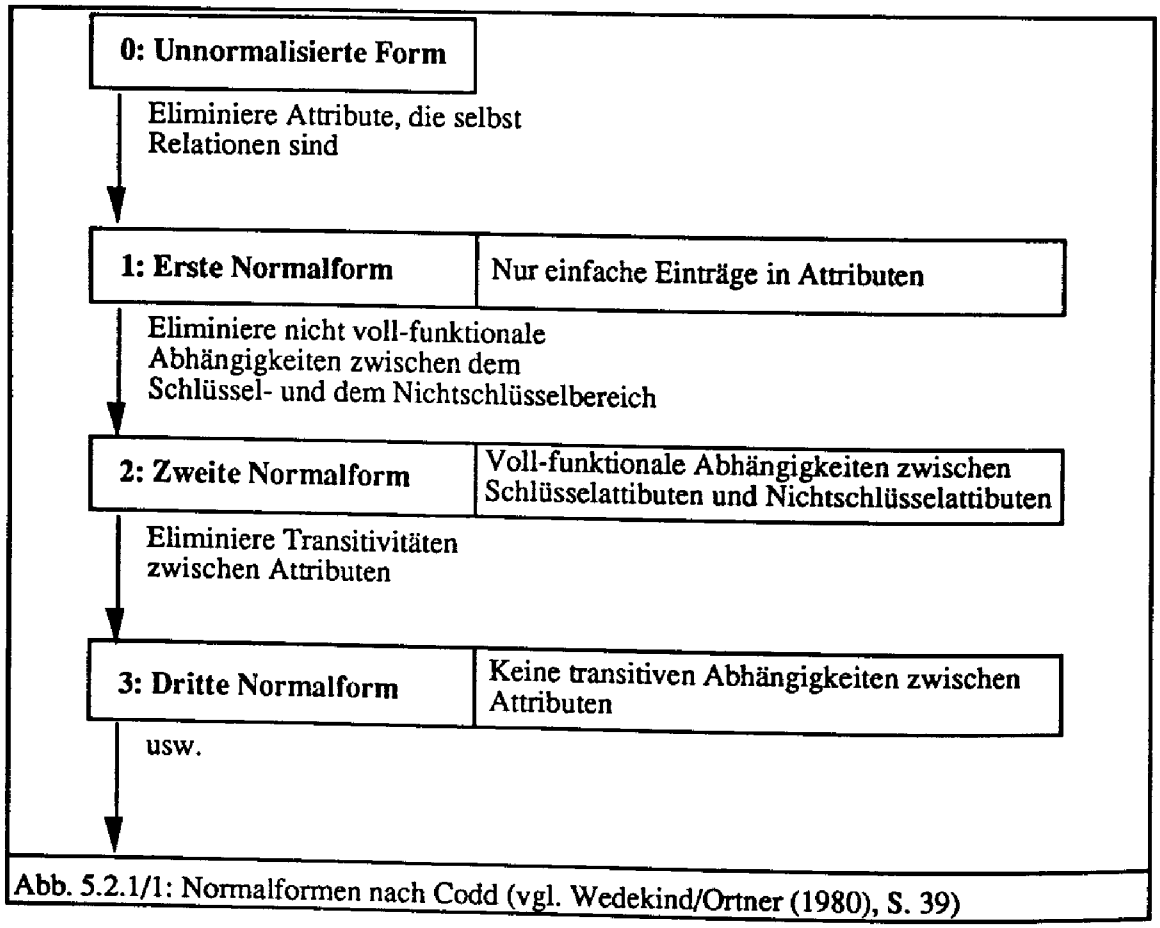


Abb. 5.2.1/1: Normalformen nach Codd (vgl. Wedekind/Ortner (1980), S. 39)

Folgende Normalformen werden unterschieden (vgl. Wedekind (1981), Vetter (1987)):

1. Normalform:

Forderung: Eine Tabelle oder Relation darf nur aus einfachen Einträgen (Attributen) bestehen, das heißt sie ist nicht weiter zerlegbar.

Maßnahme: Datensätze (Tupel), die Attribute mit mehreren Einträgen besitzen, werden in mehrere Datensätze zerlegt.

2. Normalform:

Forderung: Eine Tabelle (=Relation) darf nur aus Einträgen (Attributen) bestehen, die voll funktional abhängig vom Gesamtschlüssel sind, nicht von Schlüsselteilen. Mit dem Begriff "voll funktional abhängig" wird die vollkommene Zugehörigkeit eines Attributes zu einem Primärschlüssel umschrieben.

Maßnahme: Tabellen (Relationen) werden so in mehrere Tabellen zerlegt, daß jedes Attribut eindeutig vom jeweiligen Schlüssel abhängt.

3. Normalform:

Forderung: Eine Tabelle oder Relation darf keine Einträge (Attribute) enthalten, die untereinander funktional abhängig sind, so daß das eine Attribut ein Quasi-Schlüssel für das andere darstellt (und umgekehrt).

Maßnahme: Tabellen (Relationen), in denen zwei oder mehrere Attribute funktional (transitiv) voneinander abhängen, werden so zerlegt, daß eine zusätzliche Tabelle gebildet wird, in der das eine Attribut Schlüssel für das andere Attribut wird.

4. Normalform:

Forderung: Es sind natürliche Verbundoperationen (natural join) zwischen Attributen einer Relation zu beseitigen.

Maßnahme: Tabellen (Relationen), in denen zwei oder mehrere Schlüsselattribute existieren, zwischen denen ein natürlicher Verbund besteht, werden in mehrere Tabellen zerlegt.

5. Normalform:

Forderung: Beseitige alle logisch nicht möglichen natürlichen Verbundoperationen (natural join).

Maßnahme: Durch eine zusätzliche Zuordnungstabelle werden nur erlaubte Verbindungen ermöglicht.

Ein konsequenter Normalisierungsprozeß setzt das Datenmodell in ein Datenschema mit einem hohen syntaktischen und semantischen Aussagegehalt um. Dabei werden auch Strukturbrüche logischer oder inhaltlicher Art in der Datenstruktur deutlich, die u. U. einen Rücksprung auf die Phase 1: "Konstruktion" oder Phase 2: "Modellierung" erforderlich machen. Weiterhin werden Speicheranomalien erkannt, die beim späteren Betrieb des Datenbanksystems bei Einschub-, Modifikations- und Löschoperationen auftreten können und komplexe Prüfprogramme erfordern. Nachteil ist, daß die Strukturzerlegung sachliche Zusammenhänge zersplittert. Daher wird bei der Software-Entwicklung häufig auf eine mathematisch konsequente Normalisierung verzichtet. So wird beispielsweise vom SAP-R3-System berichtet, daß bewußt auf eine durchgängige Realisierung der 2. und 3. Normalform verzichtet wurde.

Beispiel:

Es seien folgende Relation gegeben:

Veranstaltung (Vorlesungs-Nr., Vorlesung, Hörsaal, Zeit, Dozentenname, Dozentenadresse, Fächer (des Dozenten), Fachbereich, Raum, Sprechstunde, Telefon, Student, Matrikelnr., Anzahl Semester, Studienfach, Studentenadresse, Veranstaltungsbesuch)

Eine Ausprägung könnte folgendermaßen aussehen:

Vorlesungs-Nr.	Vorlesung	Hör-saal	Zeit	Dozenten-name	Dozenten-adresse	Fächer	Fach-bereich	Raum	...
053734	Datenbanken	B1	Mo. 14-16	Müller	Paderborn	Wirtschafts-informatik	5	B3. 247	...

...	Sprechstunde	Telefon	Student	Matr.-Nr.	Anzahl Semester	Studienfach	Studenten-adresse	Veranstal-tungsbesuch
...	Fr. 8-9	2801	Schmidt, Meier, Kohl	3349300, 3443922 3349444	5, 7, 5	BWL Wirtschafts-informatik Informatik	Paderborn Paderborn Detmold	WS 91 WS 91 WS 90

(Aus technischen Gründen wurden zwei Zeilen benutzt; dargestellt ist eine zusammenhängende Relation.)

1. Normalform:

Die Relation enthält keine Attributwerte, die sich aus mehreren Elementen zusammensetzen.

Vorlesungs-Nr	Vorlesung	Hörsaal	Zeit	Dozenten-name	Dozenten-adresse	Fächer	Fach-bereich	Raum	...
053734	Datenbanken	B1	Mo. 14-16	Müller	Paderborn	Wirtschafts-informatik	5	B3. 247	...
053734	Datenbanken	B1	Mo. 14-16	Müller	Paderborn	Wirtschafts-informatik	5	B3. 247	...
053734	Datenbanken	B1	Mo. 14-16	Müller	Paderborn	Wirtschafts-informatik	5	B3. 247	...

...	Sprechstunde	Telefon	Student	Matr.-Nr.	Anzahl Semester	Studienfach	Studenten-adresse	Veranstal-tungsbesuch
...	Fr. 8-9	2801	Schmidt	3349300	5	Betriebswirt-schaftslehre	Paderborn	WS 91
...	Fr. 8-9	2801	Meier	3443922	7	Wirtschafts-informatik	Paderborn	WS 91
...	Fr. 8-9	2801	Kohl	3349444	5	Informatik	Detmold	WS 90

Das künstliche Schlüsselattribut ist unterstrichen. Der Dozent ist kein Schlüsselattribut, weil von der Annahme ausgegangen wird, daß jede Vorlesung von genau einem Dozenten gehalten wird.

Nach der 1. Normalform gilt noch: Soll die Vorlesung mit der Nr. 053734 in "Datenbanken und betriebliche Datenmodelle" umbenannt werden, so muß die Änderung nicht einmal, sondern in jedem Satz durchgeführt werden.

2. Normalform

Die Attribute Vorlesung, Hörsaal, Zeit, Dozentenname, Dozentenadresse, Fächer, Fachbereich, Raum, Sprechstunde und Telefon sind abhängig vom künstlichen Schlüsselattribut Vorlesungs-Nr. Führen wir die Matrikelnummer als weiteres künstliches Schlüsselattribut ein, so sind die Attribute Student, Semester, Studienfach und Studienadresse davon abhängig. Die Attribute sind also nicht funktional abhängig von der Schlüsselkombination (Vorlesungs-Nr., Matr.-Nr.). Von der Attributkombination (Vorlesungs-Nr., Matr.-Nr.) hängt das Attribut Veranstaltungsbesuch ab.

Daraus ergeben sich folgende Relationen:

Vorlesungs-Nr.	Vor-lesung	Hör-saal	Zeit	Dozenten-name	Dozenten-adresse	Fächer	Fach-bereich	Raum	Sprech-stunde	Telefon
053734	Daten-banken	B1	Mo. 14-16	Müller	Paderborn	Wirtschafts-informatik	5	B3. 247	Fr. 8-9	2801

Matr.-Nr.	Student	Anzahl Semester	Studienfach	Studentenadresse
Schmidt	3349300	5	Betriebswirtschaftslehre	Paderborn
Meier	3443922	7	Wirtschaftsinformatik	Paderborn
Kohl	3349444	5	Informatik	Detmold

Vorlesungs-Nr.	Matr.-Nr.	Veranstaltungsbesuch
053734	3349300	WS 91
053734	3443922	WS 91
053734	3349444	WS 90

Nach der 2. Normalform gilt weiterhin: Hält der Dozent Müller mehrere Vorlesungen, so muß z. B. eine eventuelle Änderung seiner Telefonnr. bei jeder Vorlesung neu durchgeführt werden.

3. Normalform

Eine Vorlesung wird immer eindeutig von einem Dozenten gehalten, dieser hat einen eigenen Büroraum . Da ein Name mehrfach vorkommen kann, wird die eindeutige Raumnummer als Schlüssel gewählt. Über diesen kann auf den Dozentennamen, dessen Adresse, die Fächer, den Fachbereich, die Sprechstunde und die Telefonnr. geschlossen werden, d. h. diese Attribute hängen transitiv von der Vorlesungs-Nr. über die Raumnummer ab.

Vorlesungs-Nr.	Vorlesung	Hörsaal	Zeit	Raum
053734	Datenbanken	B1	Mo. 14-16	B3.247

Dozentenname	Dozentenadresse	Fächer	Fachbereich	Raum	Sprechstunde	Telefon
Müller	Paderborn	Wirtschaftsinformatik	5	B3.247	Fr. 8-9	2801

Matr.-Nr.	Student	Anzahl Semester	Studienfach	Studentenadresse
3349300	Schmidt	5	Betriebswirtschaftslehre	Paderborn
3443922	Meier	7	Wirtschaftsinformatik	Paderborn
3349444	Kohl	5	Informatik	Detmold

Vorlesungs-Nr.	Matr.-Nr.	Veranstaltungsbesuch
053734	3349300	WS 91
053734	3443922	WS 91
053734	3349444	WS 90

4. Normalform

Die folgende Relation befindet sich in der 3. Normalform.

Vorlesungs-Nr.	Matr.-Nr.	Semester
053734	3349300	WS 91
053734	3349300	SS 91
053734	3349444	WS 91
053734	3349444	SS 91
057343	3349300	WS 91
057343	3394333	WS 91
057343	3349300	SS 90
057343	3394333	SS 90

Es wird folgendes angenommen: Jede Vorlesung wird von mehreren Studenten gehört bzw. jeder Student hört mehrere Vorlesungen (m:n-Beziehung). Zwischen Vorlesungs-Nr. und Semester besteht ebenso eine m:n-Beziehung: Jede Vorlesung wird in mehreren Semestern und in jedem Semester werden mehrere Vorlesungen gelesen. Zwischen den Relationen

Vorlesungs-Nr.	Matr.-Nr.
053734	3349300
053734	3349444
057343	3349300
057343	3394333

und

Vorlesungs-Nr.	Semester
053734	WS 91
053734	SS 91
057343	WS 91
057343	SS 90

kann durch einen natürlichen Verbund die gleiche Relation gebildet werden. Daher stellen die Relationen die 4. Normalform dar. Durch die Hinzunahme von weiteren Tupeln (057343, 3643399, SS 90) oder durch das Löschen eines Tupels (057343, 3394333, WS 91) würde dieser Sachverhalt aufgehoben werden. Damit wäre die Aufspaltung der Relationen für die 4. Normalform nicht mehr möglich.

5. Normalform

Das Tupel (057343, 3394333, SS 90) der Relation (Vorlesungs-Nr., Matr.-Nr., Semester) soll nicht mehr Bestandteil der betrachteten Relation sein.

Vorlesungs-Nr.	Matr.-Nr.	Semester
053734	3349300	WS 91
053734	3349300	SS 91
053734	3349444	WS 91
053734	3349444	SS 91
057343	3349300	WS 91
057343	3394333	WS 91
057343	3349300	SS 90

Die Relation wird zu Relationen mit jeweils 2 Attributen projiziert, die die 5. Normalform darstellen:

Vorlesungs-Nr.	Matr.-Nr.
053734	3349300
053734	3349444
057343	3349300
057343	3394333

Vorlesungs-Nr.	Semester
053734	WS 91
053734	SS 91
057343	WS 91
057343	SS 90

Matr.-Nr.	Semester
3349300	WS 91
3349300	SS 91
3349444	WS 91
3349444	SS 91
3394333	WS 91
3349300	SS 90

Aus diesen läßt sich durch natürliche Verbunde der Relationen die Ursprungs-Relation rekonstruieren:

Verbund der Relationen (Vorlesungs-Nr., Matr.-Nr.) und (Vorlesungs-Nr., Semester):

Vorlesungs-Nr.	Matr.-Nr.	Semester	
053734	3349300	WS 91	
053734	3349300	SS 91	
057334	3349444	WS 91	
057334	3399444	SS 91	
057343	3349300	WS 91	
057343	3349300	SS 90	
057343	3394333	WS 91	
057343	3394333	SS 90	*<- zu löschende Zeile*

Aus dieser Relation muß die letzte Zeile entfernt werden. Dies wird durch einen weiteren Verbund zwischen der Relation (Vorlesungs-Nr., Matr.-Nr., Semester) und der Relation (Matr.-Nr., Semester) erreicht:

Vorlesungs-Nr.	Matr.-Nr.	Semester
053734	3349300	WS 91
053734	3349300	SS 91
057334	3349444	WS 91
057334	3399444	SS 91
057343	3349300	WS 91
057343	3349300	SS 90
057343	3394333	WS 91

5.2.1.2. Struktursynthese

Bei der Struktursynthese wird im Unterschied zur Strukturzerlegung „bottom up" vorgegangen. Alle Attribute der Objekte und Beziehungen werden vorgegeben und die erlaubten funktionalen Abhängigkeiten zwischen beschreibenden und identifizierenden Attributen betrachtet. Die resultierenden "elementaren Funktionalrelationen" werden dann auf Redundanz überprüft und hieraus Relationen in der Dritten Normalform gebildet. Hierfür sind konkrete Algorithmen entwickelt worden (vgl. Wedekind (1981), S.214ff., Vetter (1987), S. 193ff)).

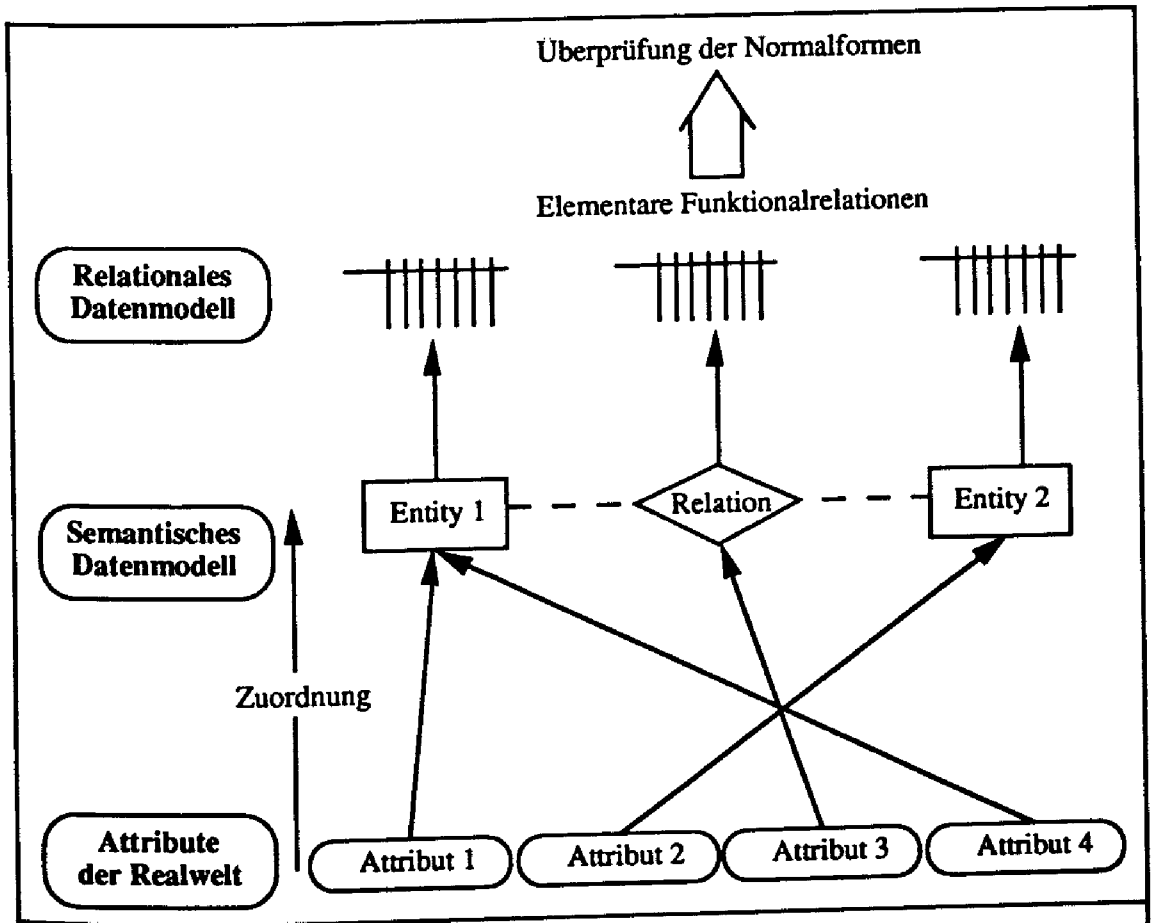


Abb. 5.2.1/1: Bottom-Up-Vorgehen der Struktursynthese

Die Struktursynthese besitzt geringe praktische Relevanz, da der abstrakte Begriff der "elementaren Funktionalrelation" den Ausgangspunkt bildet. Sie ist nur für eine globale Modellierung, nicht aber für die Ableitung der Detailschemta geeignet.

5.2.2. Dynamische Schema-Bildung

Bei der dynamischen Schema-Bildung ist es das Ziel, die durchzuführenden und in der Datenbank zulässigen Transaktionen möglichst redundanzfrei und logisch eindeutig abzubilden.

1. Vorgehensweise: Transaktionszerlegung (Transaktions-Analyse)

Notwendige Transaktionen werden in Elementaroperationen zerlegt, die sich auf bestimmte logische Grundoperationen zurückführen lassen (Sakai (1983), S. 118). Zudem werden nach der logischen und zeitlichen Grundstruktur bestimmte Klassen von Transaktionen gebildet.

2. Vorgehensweise: Transaktionssynthese

Bei der Transaktionssynthese werden aus den Abhängigkeiten der Attribute zueinander und der zulässigen Wertedomänen unmittelbar die erforderlichen Prozeduren und transaktionssichernden Programme abgeleitet. Erleichtert wird der Ansatz bei Verwendung objektorientierter Programmkonzepte.

5.2.3. Erzeugung von logischen Benutzersichten

Benutzer benötigen für ihre Anwendungen meist nur Teile der logischen Sicht des Systems; diese werden in Sichten zusammengefaßt. Deren Datenstrukturelemente werden aus Basisrelationen abgeleitet; die Beziehungen zwischen Elementen der Basisrelationen und Elementen von Sichten werden jeweils definiert. Aus Basisrelationen kann eine beliebige Anzahl von Sichten abgeleitet werden; ihre Strukturelemente können sich überlappen. Die Elemente der Sichten müssen aus der Datenbank ableitbar sein; sie müssen aber nicht unbedingt in Basisrelationen vorhanden sein (vgl. Dittmann (1977), S. 167f.).

Relationen einer Sicht können sich von den normalisierten Basisrelationen in Anzahl und Reihenfolge der Attribute und zusätzlich in der Darstellungsform und Genauigkeitsangabe der Attributausprägungen unterscheiden. Sie sind durch intrarelationale oder interrelationale Abbildungen aus den Basisrelationen zu erzeugen. Dabei kann es aus Performancegründen günstig sein, für bestimmte Sichten spezielle Relationen zu erzeugen, und diese auch zu speichern.

Statische Unterschiede zwischen Basisrelationen und Sichten bestehen, wenn die Attribute unmittelbar aus den Basisrelationen entnommen werden können und nur noch zugeordnet werden müssen, jedoch Struktur-, Darstellungs- und Benennungsänderungen existieren. Dynamische Unterschiede bestehen, wenn die Attribute algorithmisch aus einem oder mehreren Attributen von Basisrelationen erzeugt werden, ohne dort explizit tabelliert zu sein (Beispiel: Alter aus dem Attribut Geburtsdatum).

Jede Sicht sollte im Datenverwaltungssystem katalogisiert werden. Es ordnet dabei die lokalen Namen der Elemente von Sichten den globalen in den Basisrelationen zu.

6. Implementierung in ein Datenbanksystem

Aufgabe der Implementierung ist die technisch handhabbare und effiziente Überführung des Datenbankschemas in

- die zentrale Beschreibung der Datenbestände eines Datenbanksystems (Datenmodelladaption)

- die Abfragesichten, die vom Benutzer verlangt und vom Datenbanksystem unterstützt werden (Sichtenextraktion).

Technisch handhabbar bedeutet dabei, daß das Datenmodell nur Anforderungen stellen soll, die vom verfügbaren Datenbanksystem auch erfüllbar sind. Unter Effizienz ist zu verstehen, daß das System für sämtliche Benutzer ein gutes Laufzeitverhalten hat und die Speicherplatzanforderungen gering sind. Bei verteilten Datenbanken kommt hinzu, daß die Transportvolumina in den Kommunikationsnetzen möglichst gering sein sollten. Bei der lokalen Implementierung kommt es darauf an, die Nutzungsstruktur der verteilten Datenbank zu antizipieren, um zu einer sinnvollen lokalen Daten-, Funktions- und Lastverteilung zu kommen.

Die Implementierung ist kein einmaliger Prozeß. Da sich die DV-Systemarchitektur und der internen Datenbankstruktur (speziell bei den externen Schemata) laufend ändern, ist sie als kontinuierlicher Re-Designprozeß anzulegen. Dieser Prozeß ist auf der Fachabteilungsseite und auf der DV-Seite organisatorisch zu verankern.

Ziel ist dabei, die Verantwortlichkeiten für die Pflege der Dateninhalte und Datenstrukturen festzulegen und die Rechte der Nutzer auf bestimmte Datensichten zu beschränken. Der Prozeß der organisatorischen Implementierung läßt sich in Phasen einteilen (vgl. Zehnder (1988), S. 210 ff). In der Initialisierungsphase sind die erforderlichen Stammdaten zu beschaffen und in die Datenbank einzupflegen. Der spätere Nutzen einer Datenbank hängt entscheidend von der sorgfältigen Datenrecherche und Datenimplementierung ab.

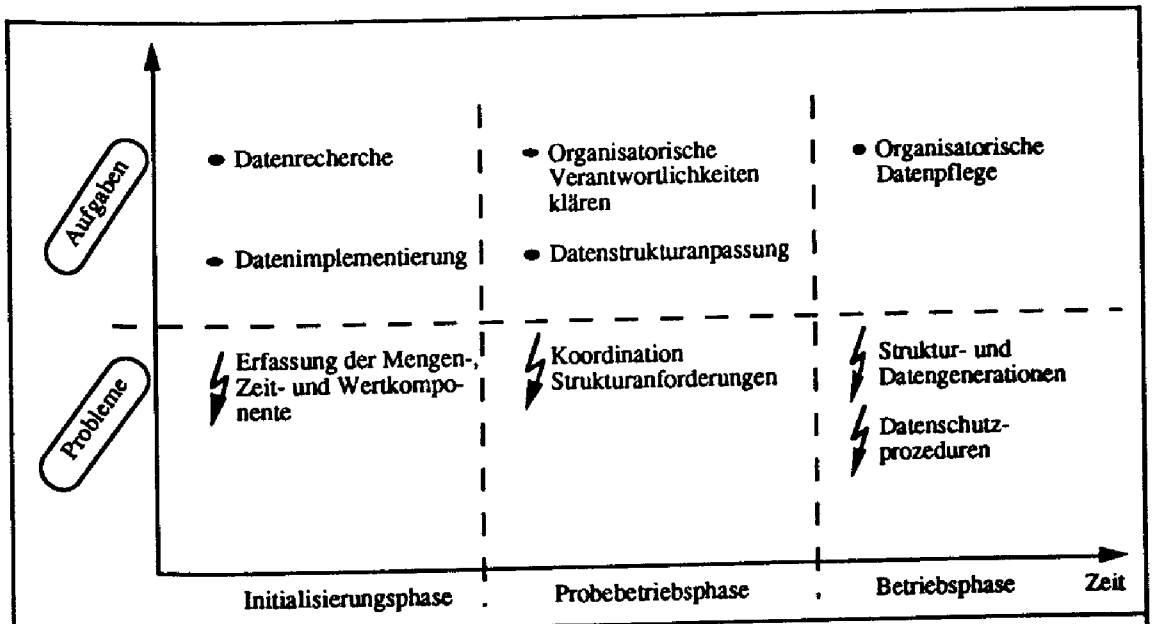


Abb. 6/1: Aufgaben und Probleme in den Phasen der Implementierung

Die Initialisierungsphase ist schwierig, weil es zeitaufwendiger manueller Erfassungs-, Strukturierungs- und Zuordnungsprozesse bedarf. Viele PPS-Datenbanken sind beispielsweise nie zum betrieblichen Nutzen eingesetzt worden, weil es nicht gelang, die erforderlichen Initialdaten durch umfassende Zeit-, Mengen- und Wertstudien zu erheben und einzupflegen. Üblicherweise sind in dieser Phase Sonderaktionen erforderlich, die sehr aufwendig sind und daher zeitweise eingestelltes Sonderpersonal benötigen. Daten sind manuell zu erfassen, aus dem vorhandenen DV-System ab- und überzuleiten und mit den geplanten Strukturierungsmerkmalen zu versehen.

Wichtig ist, daß ein Zeitpunkt definiert wird, zu dem die Daten tagesaktuell zu erheben und einzupflegen sind. Das Ziel der Probebetriebsphase ist, die gewonnenen Daten parallel in die traditionellen Systeme der Datenhaltung als auch in die Datenbank einfließen zu lassen. Die für die jeweiligen (partiellen) Datenbestände verantwortlichen Organisationseinheiten lernen dabei die übergreifende Bedeutung der von ihnen zu liefernden Datenelemente kennen und passen diese an die Erfordernisse anderer Einheiten und DV-Systeme an.

In der abschließenden Betriebsphase stehen Aspekte des Datenschutzes (Zugriffs- und Aktualisierungsrechte) sowie der Datenpflege im Vordergrund. Dabei ist die laufende Aktualisierung der Dateninhalte und -strukturen von Belang, wobei sicherzustellen ist, daß deren zeitliche Entwicklung nachvollzogen werden kann.

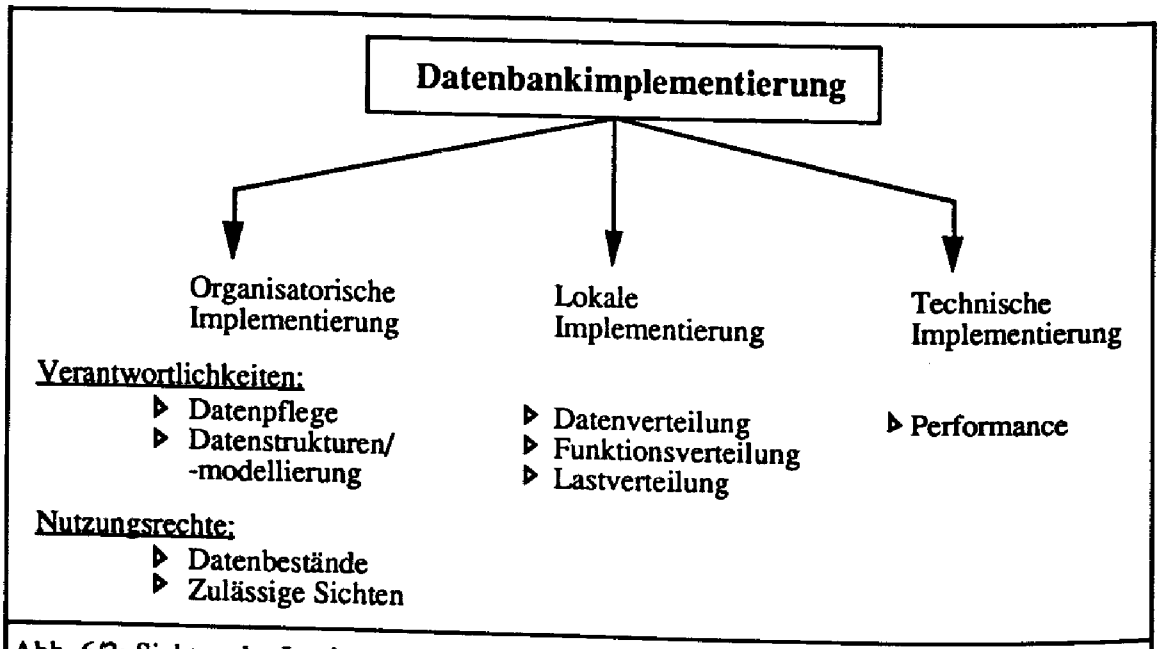


Abb. 6/2: Sichten der Implementierung

Implementierung der Datenelemente

Zur Implementierung der Datenelemente sind folgende Schritte durchzuführen:

1. Schritt: Festlegung der physischen Abbildung

Für jedes Datenelement ist anzugeben, wieviele Speichereinheiten (z. B. in kB) für ein spezifisches Datenelement vorzusehen ist. DB-Systeme setzen oft physische Grenzen für die Länge zulässiger Datensätze. Ggf. müssen Strukturen neu modelliert werden, um zu zulässigen Satzlängen zu kommen (Segmentieren logischer Sätze).

Zu entscheiden ist in diesem Schritt über

-> den Satzaufbau, die Feldtypen und die Formate,
-> die Einführung redundanter Daten und virtueller Felder,
-> die Datenkompression.

2. Schritt: Bilden von Satzbündeln

Ziel ist die Unterbringung von Sätzen in physischer Nachbarschaft, um häufig benötigte externe Sichten mit höherer Effizienz gewinnen zu können. Zwischen der Satzbündelung

und der Schemabildung bestehen vielfältige Interdependenzen. Freiheitsgrade sind neben der physisch benachbarten Speicherung die Zuweisung von Blockgrößen und Speicherplätzen.

	Bündelung	**Blockgröße**	**Speicherplatz-zuweisung**
Häufige Anforderung einer Teilmenge von Datensätzen	gebündelt	große Blockgröße	nahe Anker
Häufiger Direktzugriff auf einzelne Sätze	keine	kleine Blockgröße	

Abb. 6/3: Bewertungsgrößen für die Bildung von Satzbündeln

3. Schritt: Definition der physischen Suchpfade

Bei der Festlegung der Suchpfade ist die Zugriffshäufigkeit, die Zugriffsstruktur (d. h. in welchen Sichten) und die lokale Verteilung der Nutzer zu beachten.

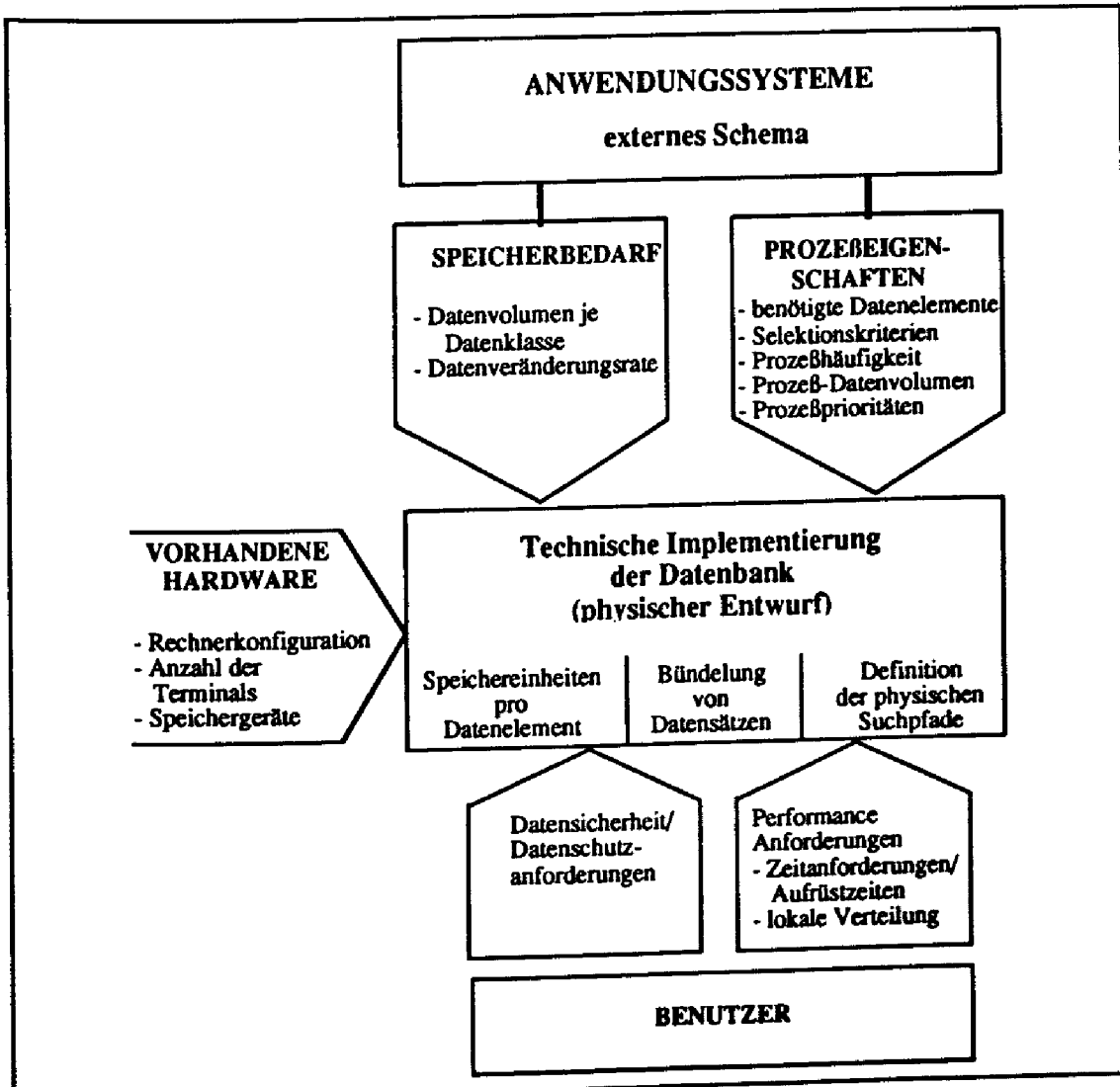


Abb. 6/4: Einflußgrößen der Implementierung (vgl. Mayr/ Dittrich/Lockemann (1987), S. 543)

Kapitel 3: Datenbanksysteme

1. Kennzeichen

Datenbanken sind Softwaresysteme zur Einrichtung, Verwaltung und Nutzung von Datenbeständen entsprechend dem logischen Datenmodell.

Das logische Datenmodell legt dabei im Grunde fest, welche

-> Datenobjekt-Typen
-> Datenstruktur-Typen
-> Datenoperator-Typen

vom Datenbanksystem gefordert werden.

Der Anwender wird häufig Kompromisse bei der Umsetzung des Datenmodells in ein konkretes Datenbanksystem machen müssen.

Anforderungen des Datenmodells	werden vom Datenbanksystem nicht erfüllt
hinsichtlich Datenobjekten	• Ablage komplexer Objekte in separaten Dateien mit Verweis in der Datenbank • Aufspaltung in mehrere einfache Attribute und Berechnung des komplexen Objektes in externer Sicht
hinsichtlich Datenstrukturen	• nur hinsichtlich Flexibilität und Programmabhängigkeit nutzerrelevant; durch Programme umgehen
hinsichtlich Datenoperatoren	• Verankerung in den Programmen der externen Sicht

Abb. 1/1: Wege zur Realisierung von Datenmodellanforderungen außerhalb des Datenbanksystems

Neben der Suche nach einem alternativen Datenbanksystem und dessen technischer und wirtschaftlicher Bewertung existieren eine Reihe von pragmatischen Wegen.

Allerdings sind Datenbanksysteme „lebende Systeme", d. h. sie werden durch die jeweiligen Hersteller entsprechend den Markterfordernissen weiterentwickelt. Es entstehen bei dieser Entwicklung oft Hybrid-Systeme, d. h. der jeweilige Anbieter fügt neue Möglichkeiten hinzu, ohne ein grundlegend neues Systemdesign durchzuführen.

2. Bestandteile

Ein Datenbanksystem läßt sich in Form eines Schalenmodells darstellen (vgl. Martin (1988), S. 154ff., Zehnder (1989), S. 110ff.).

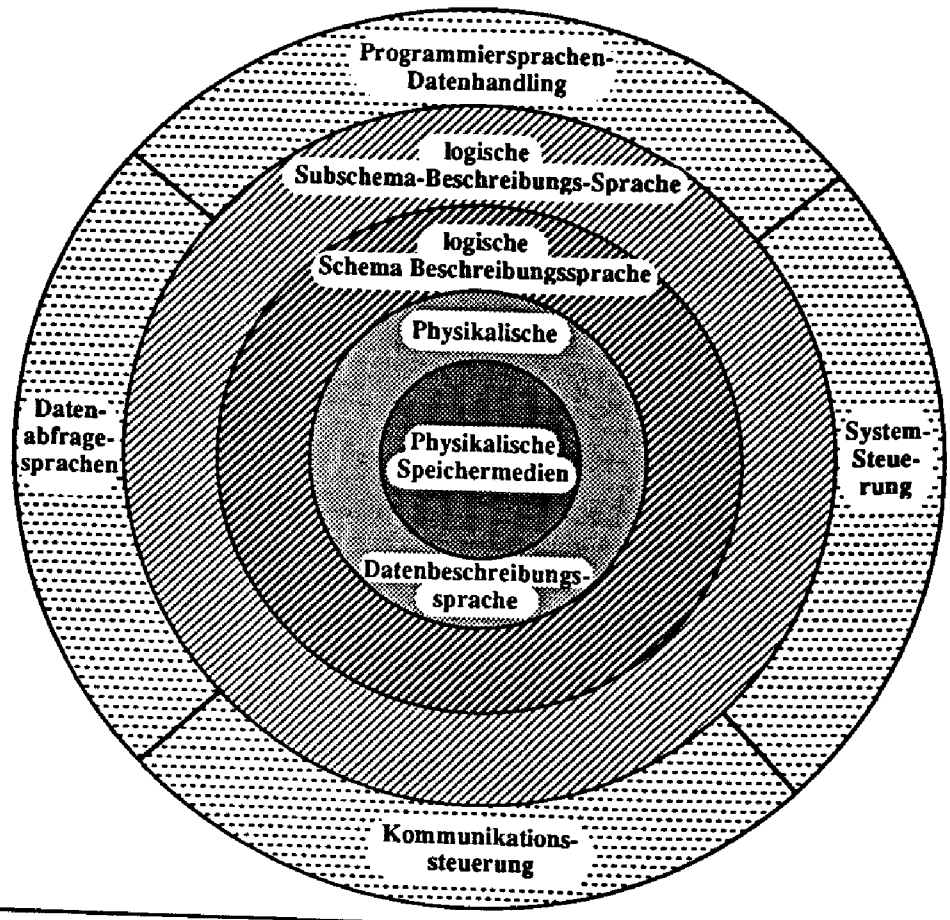


Abb. 2/1: Schichtenmodell eines Datenbanksystems (in Anlehnung an Martin (1988), S. 155 ff)

Dabei lassen sich drei Schichten differenzieren:

1. Schicht: Physikalische Schicht (internes Schema)

Zur physikalischen Schicht gehört zum einen die physikalische Datenbank in Form entsprechend adressierbarer Speichermedien. Zum anderen verfügen die Datenbanksysteme über eine physikalische Datenbeschreibungssprache (Physical Data Description Language), mit deren Hilfe die Speichermedien und die physischen Datenstrukturen spezifiziert werden. Dabei sind unter anderem die Speicherungs-, Adressierungs- und Suchtechniken zu bestimmen. Einzelfragen sind dabei die Festlegung der physischen Blocklänge, die Verwendung von Komprimierungstechniken (vgl. Schlageter/Stucky (1983), Niedereichholz (1983)).

2. Schicht: Logische Schicht (logisches Schema)

Zur logischen Schicht gehören die Datenbanksprachen, die die logischen Sichten der Anwender auf die Daten beschreiben. Danach leitet das Datenbanksystem die Anwendersichten aus den physikalischen Sätzen ab.

Mittels der Schema-Beschreibungssprache (Data Definition Language) wird das logische Modell und dessen Zusammenhänge mit der physikalischen Speicherung spezifiziert. Der einzelne Anwender definiert dann sein jeweiliges externes Schema mit der sogenannten

Subschema-Beschreibungssprache (Data Manipulation Language). Häufig sind diese Sprachen eingebettet in eine prozedurorientierte Programmiersprache der 3.Generation (z. B. COBOL)

3. Schicht: Externe Schicht (externes Schema)

Zur Handhabungsschicht gehören erstens Programmier-Datenhandling-Komponenten, die es ermöglichen, von Programmiersprachen über Komponenten der Data-Manipulation-Language auf die Datenbank zuzugreifen. Weiterhin existieren in den Datenbanksprachen Datenabfrage-Komponenten, mit denen ohne vorgeschaltete Programme auf die Datenbank zugegriffen werden kann.

Spezielle Kommunikationskomponenten (Data Communication Language) regeln den Datenaustausch zwischen den dezentralen Arbeitsstationen (mit deren Abfragen, Programmen, dezentralen Datenbankausschnitten) und dem zentralen Datenbestand. Dabei ist die logische Konsistenz des Datenbestandes unabhängig von dessen Verteilung und dem Datenschutz zu sichern. Die Arbeitsstationen sind oft selbständige Rechner, die auf der Grundlage der Datenbank eigene Programme betreiben oder spezielle Benutzeroberflächen verwenden. Dabei soll die Data Communication Language die Eigenheiten der Kommunikationsdienste/-netze und Protokolle verdecken und einen unabhängigen Betrieb von (frontend-) Auswertungsprozessen und (backend-) Datenverwaltungsprozessen sicherstellen.

Schicht	Aufgaben	Instrumente
Physikalische Schicht	Leistungssteuerung - Anlegen von Zugriffspfaden - Auswahl von Datensicherungsstrategien - Verteilung der Daten auf Datenträger - Auswahl von Speicherverwaltungsstrategien - Reorganisation der Datenbasis Datensicherheit Datenschutz	**Physical Data Description Language**
Logische Schicht	Operatorendefinition - Definition zulässiger Operatoren	**Data Manipulation Language (DML)**
	Strukturdefinition - Definition des Datenmodells	**Data Definition Language (DDL)**
	Unverletzbarkeit der Datenbasis - Sicherung von Integritätsbedingungen	**Data Transaction Language**
Externe Schicht	Handling Mehrnutzerbetrieb	**Data Communication Language (DCL)**
	Handling Netzkommunikation	
	Realisierung Benutzerschnittstellen	

Abb. 2/2: Bestandteile eines Datenbanksystems und seine Aufgaben

Die Komponenten eines Datenbanksystems sind unterschiedlich wichtig für die Systemeinrichtung und den Systembetrieb. Daher kann es sich aus Performance-Gründen anbieten, Komponenten nach bestimmten Phasen auszulagern und nur bei Bedarf wieder zu installieren.

Phasen	Teilaufgaben	Komponenten
Initialphase	- Einrichtung der Speichermedien	Physical Data Description Language
	- Einrichtung der DB-Struktur	Data Definition Language
Produktions-phase	- Sicherung der DB-Konsistenz - Sicherung der DB-Kommunikation	Data Manipulation Language
	- Sicherung der DB-Kommunikation	Data Communication Language

Abb. 2/3: Phasen des DB-Einsatzes und Teilaufgaben der Komponenten

2.1. Datenbanksprachen

Datenbanksprachen bestehen jeweils aus einer Befehlsstruktur zum Aufbau (Data Definition Language) und zur Benutzung der Datenbank (Data Manipulation Language) sowie zugehöriger Übersetzungskomponenten (Compiler bzw. Interpreter).

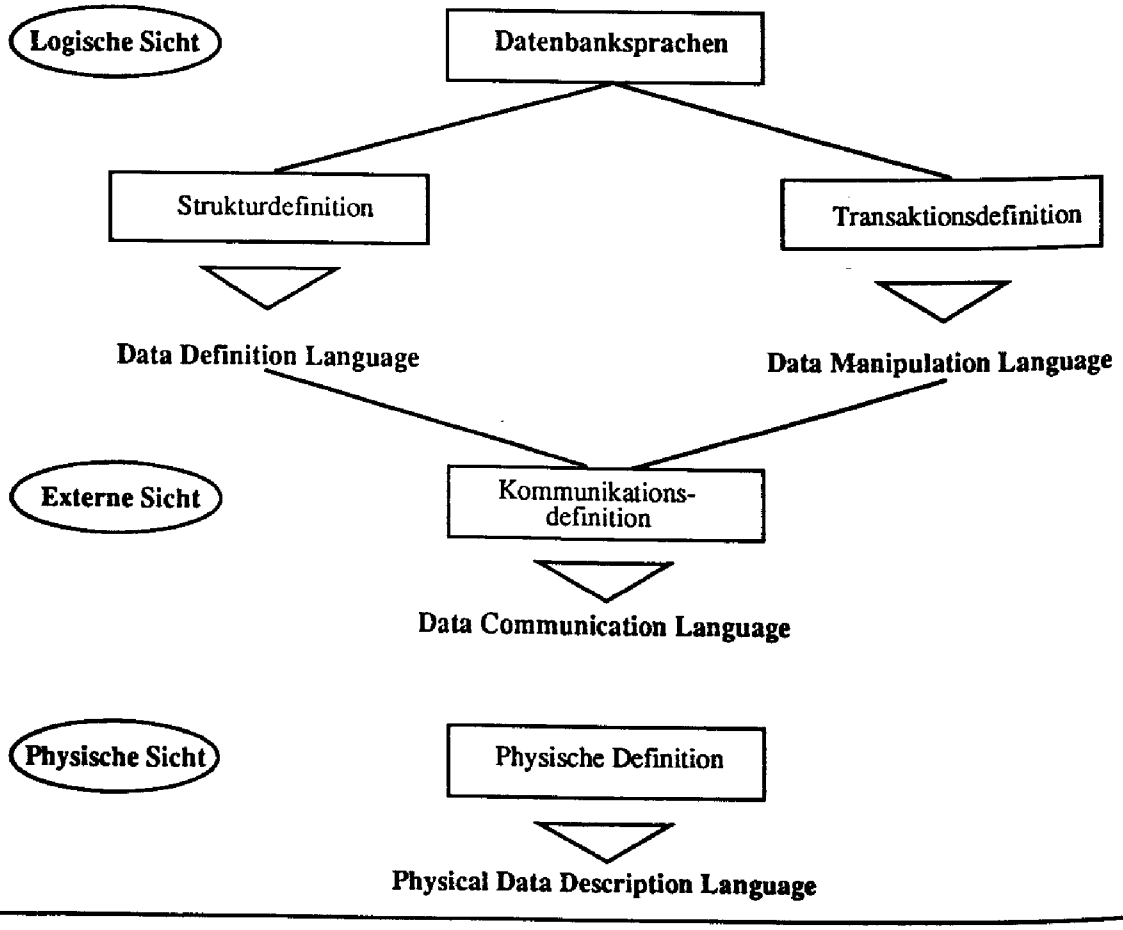


Abb. 2.1/1: Funktionen und Arten von Datenbanksprachen

Der Aufbau der Datenbank samt Programmschnittstellen und physikalischer Speicherstruktur ist normalerweise eine Aufgabe für den DV-Fachmann. Demgegenüber sind bei der Be-

nutzung einer Datenbank unterschiedliche Anwendertypen zu differenzieren und daraus Anforderungen an eine Datenmanipulationssprache abzuleiten (vgl. Zehnder (1989), S.117ff).

	Benutzertyp	**Kennzeichen**
1)	Gleichartige, interaktive Abfragen (menschliche Nutzer)	- hohe Frequenz - hohe Vorstrukturierung
2)	Abfragen und Dateninput durch Programme	- hohe Frequenz - mittlere Vorstrukturierung
3)	Verschiedenartige, interaktive Abfragen	- mittlere Frequenz - keine Vorstrukturierung
4)	Umorganisation der Datenbank	- geringe Frequenz - geringe Vorstrukturierung

Abb. 2.1/2: Benutzertypen für eine Datenmanipulationssprache

Neben dem Benutzertyp sind bestimmte Fälle der Datenselektion und Datenbankausgabe zu unterscheiden (vgl. Zehnder (1989), S.110ff).

Benutzereingabe	**Selektionsprozeß**	**Datenbankausgabe**
1. Adresse des Datums	Adressenorientierte Suche	a)Ausgewählte Daten
2. relative Adresse		b)Anzahl der Elemente
3. Merkmal des Datums	Mengenorientierte Suche	c) Kenngrößen
4. Wertebereiche		

Abb. 2.1/3: Selektionstypen für eine Datenmanipulationssprache

Bei der adressenorientierten Suche hat der Benutzer die logische Adresse zu spezifizieren, nach denen er sucht (*Beispiel: Student mit der Matrikelnummer 4711*). Demgegenüber werden bei der mengenorientierten Suche nur die Merkmalsausprägungen von gesuchten Sätzen angegeben (*Beispiel: Student mit Veranstaltungsbelegung Datenbank*).

Datenmanipulationssprachen (DML) lassen sich in Typen einteilen: Transaktionsorientierte DML unterstützen die erforderlichen Transaktionen zum Suchen, Modifizieren und Löschen von Dateninhalten. Demgegenüber sind applikationsorientierte Sprachen an Programmieranforderungen orientiert.

Selbständige DML können unabhängig von einer Programmiersprache eingesetzt werden, enthalten also alle Befehle, die für eine Mutation (Datenauswahl, Datenausgabe, Einfügen, Löschen) benötigt werden. Eingebettete DML ergänzen eine Wirts-Programmiersprache um spezielle Anweisungen oder Makros für den Datenbankverkehr. Selbständige Sprachen sind i. d. R. für ungeübte Benutzer leichter verständlich; für Anwendungsprogrammierer sind eingebettete Sprachen effizienter.

Kriterium	Ausprägungen	
Ausrichtung	Transaktionsorientiert *SQL*	Applikationsorientiert *ADABAS-Natural* *PROGRESS*
Anlehnung an Programmiersprache	Selbständig *SQL*	Eingebettet *DL/1 (COBOL)* *CODASYL-DBTG (COBOL)*
Vorgehen bei der Selektion von Datenbeständen	Deskriptiv *SQL*	Prozedural *DL/1* *CODASYL-DBTG*
Umfang der selektierten Datenbestände	Datenmengen *SQL*	Datensätze *DL/1* *CODASYL-DBTG*
Benutzeroberfläche	Verbal *SQL*	Graphisch *QBE*

Abb. 2.1/4: Typisierung von Datenmanipulationssprachen mit Beispielen

Bei deskriptiven Sprachen werden die auszuwählenden Daten als Datenmenge gesamthaft gekennzeichnet (WAS?); die erforderlichen Datenoperationen müssen nicht im einzelnen beschrieben werden. Prozedurale Sprachen stellen hingegen Operationen (wie FIND, OBTAIN, INSERT, STORE, MODIFY, DELETE) zur Verfügung, mit denen die Teilschritte zu programmieren sind, um die Daten zu manipulieren (WIE?). Prozedurale Sprachen existieren für hierarchische und Netzwerk- Datenbanksysteme, relationale Systeme verfügen über deskriptive DML. Deskriptive Sprachen entsprechen eher der menschlichen Denkweise und sind für ungeübte Benutzer geeignet. Prozedurale Sprachen setzen demgegenüber voraus, daß der Benutzer das externe Datenbankschema kennt.

Man spricht von Datenabhängigkeit, wenn bei der Formulierung von Operationen auf Datenverwaltungssysteme die vordeklarierten Datenstrukturen, Dateiorganisationsformen usw. berücksichtigt werden müssen. Ziel der Entwicklung deskriptiver Sprachen ist ein möglichst hoher Grad an Datenunabhängigkeit, d. h. die weitgehende Entkoppelung von Anwendung und Datenrepräsentation (vgl. Reuter/Schiele/Zeller (1988)).

Verbale Sprachen verwenden "normale" DV-Ausdrücke; der Benutzer formuliert seine Anfrage in einer linearen Notation. Bei graphik-orientierten Sprachen werden auf einem graphischen Bildschirm zweidimensional die Daten gekennzeichnet: Der Benutzer benennt zuerst die Relation, mit der er arbeiten will. Die Antwort wird nun dadurch spezifiziert, daß ein Beispielsatz in diese Tabelle eingesetzt wird (vgl. Schlageter/Stucky (1983), S.155ff, Zehnder (1989), S.120ff).

Sprachniveau	niedrig	mittel	hoch	sehr hoch	Beispiele
Kennzeichen	Adressen-auswahl	Attribut werte	Mengenbe-schreibung	Natürliche Sprache	
Netzwerk	X				DBTG
Hierarchisch	X	X			DL1
Relational	X	X	X		SQL, QBE

Abb. 2.1/5: Abhängigkeit des Sprachniveaus vom Datenmodelltyp (vgl. Everest/Weber (1977), S. 345)

Bei der Beurteilung von Datenbanksprachen müssen neben den benutzerspezifischen Aspekten System-Gesichtspunkte herangezogen werden. Diese beschreiben die Anforderungen, die eine Datenbank-Sprache an das DV-System stellt, um eine bestimmte Performance zu erreichen. Zur Zeit gilt wohl die Regel, daß herstellerspezifische Datenbanksprachen die Eigenheiten der jeweiligen Hardware/Software besser ausnutzen und dadurch ein besseres Speicherplatz- und Laufzeitverhalten aufweisen.

Systemgesichtspunkte	**Vorteile**	**Nachteile**
Unabhängigkeit von physischer Speicherstruktur	- Flexibler Speichermedieneinsatz - Investitionssicherheit, da neue Medien nicht gleichzeitig neue DB-Sprache bedingen	- Performance geringer
Herstellerübergreifende Normung	- Portabilität zwischen Rechner-/ Datenbanksystemen - Personaltransfer möglich - Langfristige Stabilität für Unternehmensdatenmodell - Einsetzbarkeit von Standardsoftware mit Standard-DBS-Zugriffen	- geringere Performance - Behinderung der Innovation

Abb. 2.1/6: Vor- und Nachteile verschiedener Systemgesichtspunkte

Das Leistungsverhalten einer Datenbanksprache ist für deren Akzeptanz ein entscheidender Faktor. Dadurch wird nicht nur der Benutzerkomfort bestimmt, sondern mittelfristig auch die Aktualität und Vollständigkeit der bestehenden Datenbestände. Daher läßt sich häufig der Leistungsgrad in Abhängigkeit vom Benutzertyp (Datenlevel, Informationslevel, Anweisungslevel, Security-Level, Funktions-Level) abstufen und die Betriebsart (Batch oder Realtime) wählen .

2.1.1. Datenbank-Sprache für das hierarchische Modell

Als Beispiel wird die Sprache DL/1 der IBM betrachtet, die auf dem Datenbanksystem IMS (= Information Management System) aufbaut. Es handelt sich um eine prozedurale, in COBOL eingebettete Datenbanksprache (vgl. Martin (1988), S.194ff, Schlageter/ Stucky (1983), S. 121ff)

Die Begriffsstrukturen des IMS sind mit denen des Entity-Relationship-Modells nur begrenzt vergleichbar. Daher werden sie kurz erläutert:

Bei einer IMS-Datenbank muß man die physische und die logische Ebene unterscheiden. Physisch besteht die Datenbank aus Datenbankteilen (physical databases - PDB), die sich aus Typen physikalischer Datenbankrecords (physical database records types - PDBR-T) zusammensetzen. Zu jedem PDB gibt es hierarchisch aufgebaute Ausprägungen (physical database records - PDBR). Ein PDBR besteht aus einem Wurzel-Objekt mit hierarchisch untergeordneten Objekten.

Jeder Datenbanksatz (PDBR) besteht aus Segmenten (*segments*). Wurzelsegmente (*Root segments)* sind die Einstiegspunkte in die Baum-Struktur des Satzes, d. h. ein Einstieg in die Datenstruktur ist jeweils nur an einem Wurzelsegment möglich. Jedes Segment besteht aus Feldern, die vergleichbar sind mit Attributen bzw. Attributausprägungen im Entity-Relationship-Modell.

Jeder physische Datenbankteil (PDB) wird durch eine *database description (DBD)* beschrieben.

Um mehrere PDBs durch logische Zeiger miteinander in Beziehung zu setzen, werden logische Verbindungen (*logical databases* - LDB) konstruiert. LDB können jedoch auch dazu verwendet werden, um benutzerspezifische Sichten auf die Datenbankstruktur darzustellen. Die Beschreibung dieser Sichten erfolgt im *program communication block* (PCB). Die für einen Benutzer relevanten logischen Datenbanken (LDB) sind sein externes Modell.

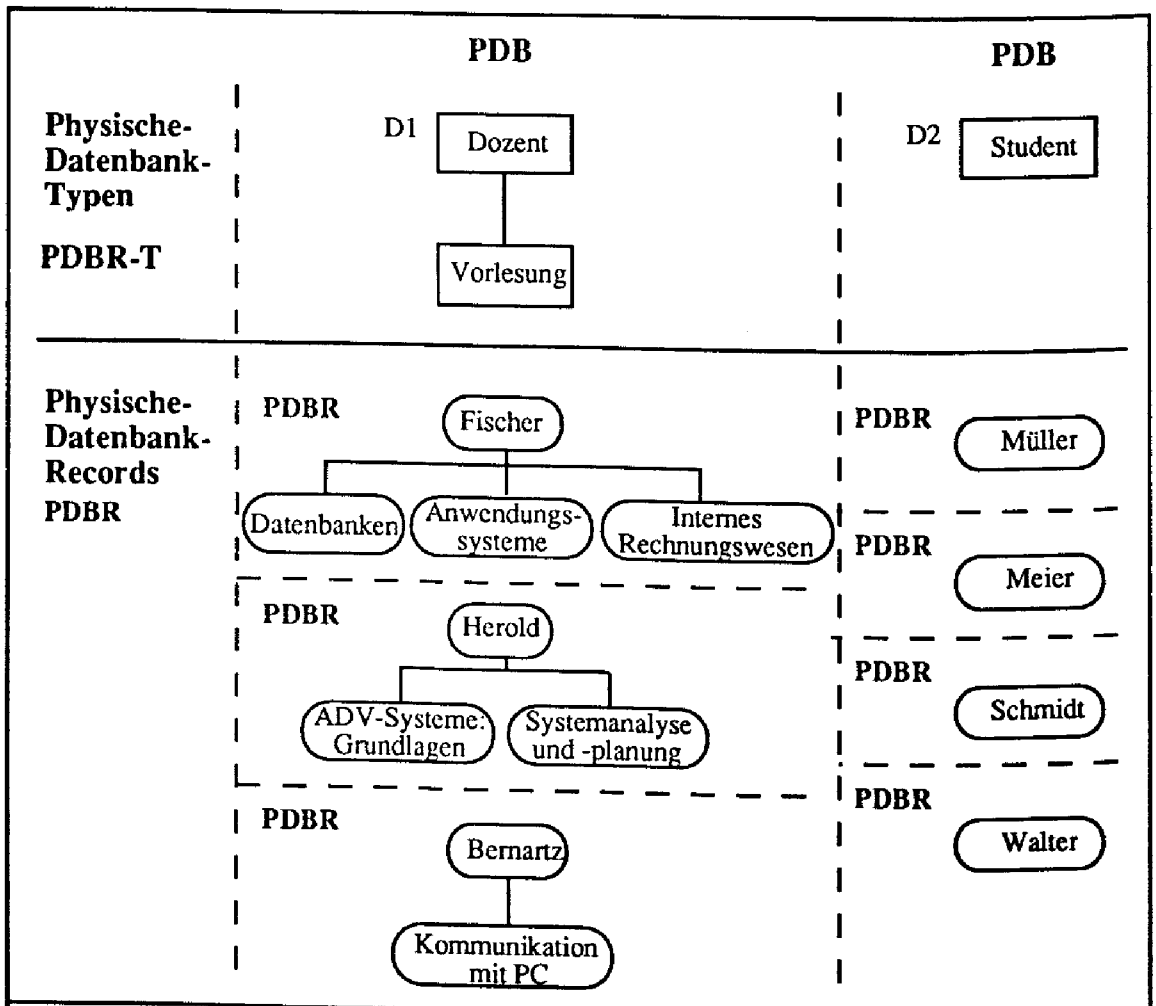


Abb. 2.1.1/1: Erläuterung der Begriffe des IMS-Systems

Der Aufbau einer Datenbankbeschreibung (DBD) soll anhand der Dozenten und der Vorlesungsdaten dargestellt werden :

```
D1:  DBD     NAME = D1, ACCESS = HIDAM
     SEGM    NAME = Dozent, BYTES = 113
     FIELD   NAME = (Name, SEQ, U), BYTES = 12, START = 1
     FIELD   NAME = Fachbereich, BYTES = 2, START = 13
     FIELD   NAME = Schwerpunkt, BYTES = 22, START = 15
     FIELD   NAME = Raum, BYTES = 7, START = 37
     FIELD   NAME = Telefn, BYTES = 4, START = 44
     FIELD   NAME = Sprechstunde, BYTES = 25, START = 48
     FIELD   NAME = Adresse, BYTES = 40, Start = 73
     SEGM    NAME = (Vorlesung, SEQ, M), PARENT = Dozent, BYTES = 42
     FIELD   NAME = Titel, BYTES = 15, START = 1
     FIELD   NAME = Zeit, BYTES = 12, START = 16
     FIELD   NAME = Raum, BYTES = 7, START = 28
     FIELD   NAME = Semester, BYTES = 7, START = 35
```

Der Baum, der die Daten für Dozent und Vorlesung enthält, heißt D1. Die Zugriffstechnik (physische Datenorganisationstechnik) ist HIDAM. IMS kennt vier solcher Techniken.

Zugriffsmethode	Speicher-organisation	Speicher-medium
HSAM: Hierarchical Sequential Access Method Hierarchisch Sequentielle Zugriffsmethode	sequentiell	sequentiell
HISAM: Hierarchical Index Sequential Access Method Hierarchisch indexsequentielle Zugriffsmethode	index-sequentiell	sequentiell
HDAM: Hierarchical Direct Access Method Hierarchische Direktzugriffsmethode	direkt	blockadressiert
HIDAM: Hierarchical Index Direct Access Method Hierarchisch Indizierte Direktzugriffsmethode	index-sequentiell	blockadressiert

Abb. 2.1.1/2: Physische Datenorganisationstechniken bei IMS

Für jeden Knoten des Baumes wird ein Segment definiert. Jedes Segment besitzt Felder, die mit ihrem Namen, ihrer Feldgröße in Bytes und ihrer Startadresse innerhalb des Segments angegeben werden. Die Startadresse beschreibt jeweils die Lage des ersten Bytes innerhalb des Segments. Bei den identifizierenden Feldern muß die Sortierung innerhalb des Baumes (SEQ = sequentiell) und die Häufigkeit angegeben werden (U = eindeutig (unique), M = mehrfach (multiple)). Bei hierarchisch untergeordneten Entities wie Dozent wird die unmittelbare Abhängigkeit durch Angabe des "Vater"-Entities angegeben .

Die Datenbankbeschreibung für den Studenten erfolgt analog:

```
D2:  DBD     NAME = D2, ACCESS = HIDAM
     SEGM    NAME = Student, BYTES = 61
     FIELD   NAME = (Matr.-Nr., SEQ, U), BYTES = 7, START = 1
     FIELD   NAME = Name, BYTES = 12, START = 8
```

FIELD NAME = Semester, BYTES = 2, START = 20
FIELD NAME = Studienfach, BYTES = 15, START = 22
FIELD NAME = Adresse, BYTES = 25, START = 37

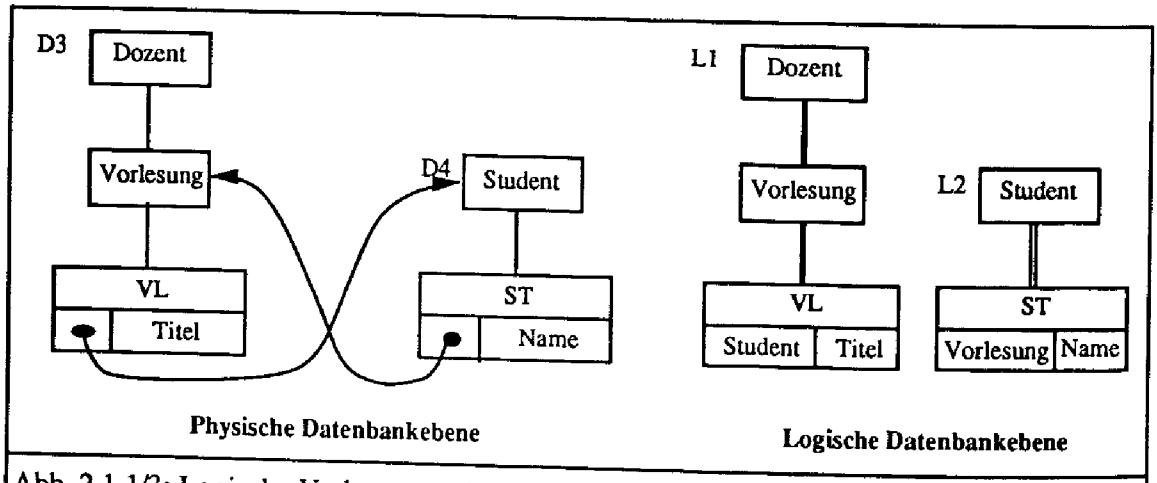


Abb. 2.1.1/3: Logische Verkettung mit Zeigern

Zur Darstellung von m:n-Beziehungen werden logische Strukturen benutzt, die durch Zeiger realisiert werden. Dazu werden zusätzliche Entities vergleichbar mit den Kett-Entities im Netzwerkmodell eingeführt:

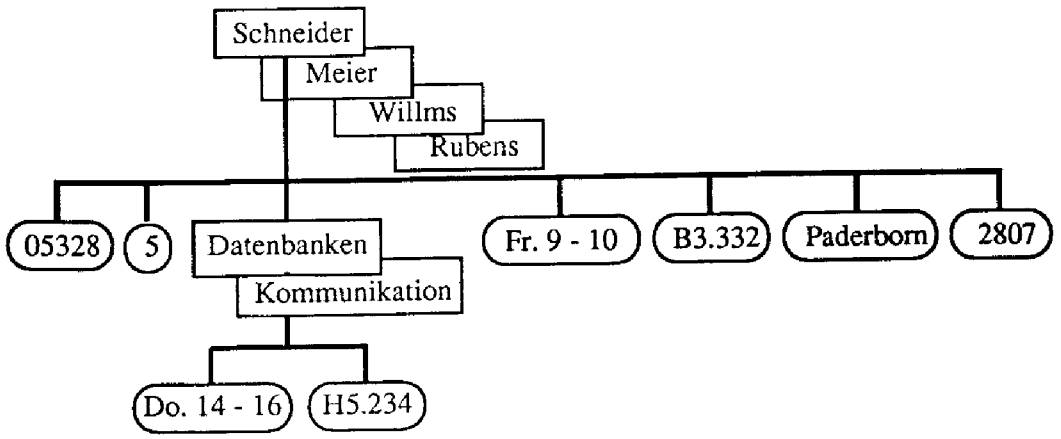
D3:
DBD NAME = D3, ACCESS = HIDAM
SEGM NAME = Dozent, BYTES = 113
FIELD NAME = (Name, SEQ, U), BYTES = 12, START = 1
FIELD NAME = Fachbereich, BYTES = 2, START = 13
FIELD NAME = Schwerpunkt, BYTES = 22, START = 15
FIELD NAME = Raum, BYTES = 7, START = 37
FIELD NAME = Telefnr, BYTES = 4, START = 44
FIELD NAME = Sprechstunde, BYTES = 25, START = 48
FIELD NAME = Adresse, BYTES = 40, Start = 73
SEGM NAME = (Vorlesung, SEQ, M), PARENT = Dozent, BYTES = 42
LCHILD NAME = (ST, D4), PAIR = VL
FIELD NAME = Titel, BYTES = 15, START = 1
FIELD NAME = Zeit, BYTES = 12, START = 16
FIELD NAME = Raum, BYTES = 7, START = 28
FIELD NAME = Semester, BYTES = 7, START = 35
SEGM NAME = VL, POINTER = (LPARNT, PAIRED,...), PARENT = (Vorlesung, (Student, D4)), Bytes = 52
FIELD NAME = (Nr., SEQ, M),...
FIELD NAME = (Matr.-Nr., SEQ, U), BYTES = 7, START = 1
FIELD NAME = Name, BYTES = 12, START = 8
FIELD NAME = Semester, BYTES = 2, START = 20
FIELD NAME = Studienfach, BYTES = 15, START = 22
FIELD NAME = Adresse, BYTES = 25, START = 37

D4:
DBD NAME = D4, ACCESS = HIDAM
SEGM NAME = Student, BYTES = 61
LCHILD NAME = (VL, D3), PAIR = ST
SEGM NAME = ST, POINTER = (LPARNT, PAIRED,...),

PARENT = (Student, (Vorlesung, D3)), Bytes = 52
FIELD NAME = (Matr.-Nr., SEQ, U), BYTES = 7, START = 1
FIELD NAME = Name, BYTES = 12, START = 8
FIELD NAME = Semester, BYTES = 2, START = 20
FIELD NAME = Studienfach, BYTES = 15, START = 22
FIELD NAME = Adresse, BYTES = 25, START = 37

Die physische Verkettung erfolgt durch die Angabe PARENT = (Vorlesung, (Student, D4)), d. h., die *Vorlesung* ist physischer Vater, Segment *Student* aus der PDB D4 ist logischer Vater von *ST*. In VL wird durch die LCHILD-Anweisung *(VL, D3)* ausgedrückt, daß das Segment *Student* logischer Vater von Segment *VL* der PDB D3 ist.

Diese Methode ist analog für die zweite Sichtweise der m:n-Beziehung vorhanden. Die Anweisung PAIR = VL bzw. PAIR = ST besagt, daß die Beziehung von D3 die Umkehrung der Beziehung von D4 ist.



Abfrage: Benutzt Dozent Schneider für seine Vorlesungen Raum H5.234

Lösung:		Kommentar:
	GU Dozent (Name = Schneider)	Dozent ist Einstiegspunkt in die zu durchsuchende Baumstruktur. Der zu suchende Name ist "Schneider".
NS:	GN Vorlesung (Raum = H5.234)	Finde in der nächsten Ebene alle Vorlesungen nach "Raum = H5.234".
	Goto NS	Gehe zur nächsten Vorlesung.

Abb. 2.1.1/4: Abfragen im hierarchischen Datenbankschema

2.1.2. Datenbank-Sprache für das Netzwerk-Modell

Als Beispiel wird die Sprache der CODASYL-Gruppe DBTG verwendet. Diese basierte ursprünglich auf der Programmiersprache COBOL, wurde inzwischen aber in weitere Programmiersprachen eingebettet. Zugehörige Datenbankkerne sind Siemens UDS, DEC DBMS10, CDC IMF, Honeywell IDS, Unisys Sperry DMS 1000; es handelt sich somit um ein verbreitetes System. (vgl. Martin (1988)und Zehnder (1989), S. 102ff).

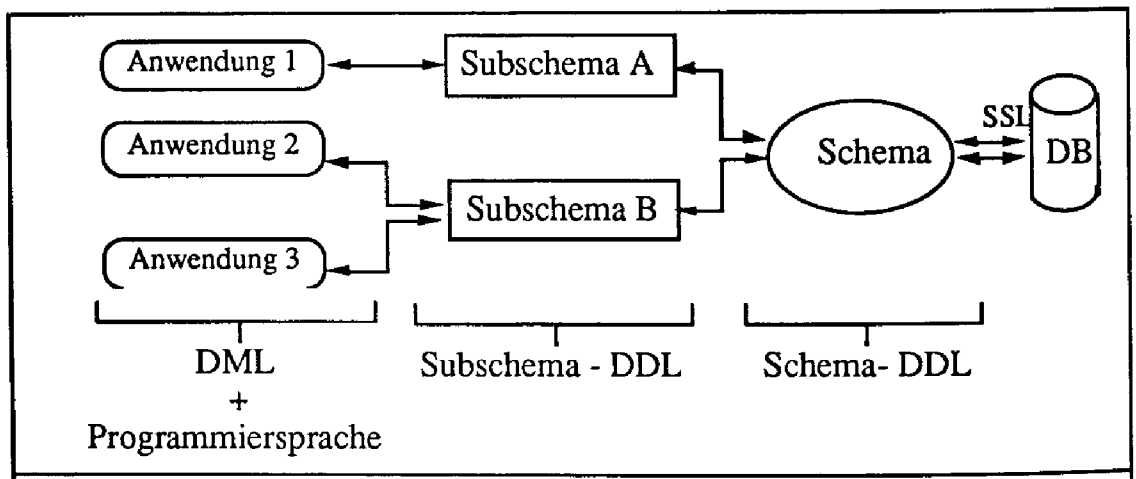


Abb. 2.1.2/1: CODASYL-Vorschlag einer Netzwerk-Datenbank

Die Datenbanksprache besteht aus einer Definitionssprache für das konzeptuelle Schema (Schema-DDL) , einer Subschema-Beschreibungssprache und einer Datenmanipulationssprache (DML) sowie einer Speicherbeschreibungssprache (SSL).

Entsprechend dem Netzwerkmodell enthält die CODASYL-DDL zwei Grundelemente:

(1) RECORDS (Datensätze) bilden als Datenobjekte die Knoten des Netzwerks. In einem RECORD-ENTRY wird der Name des RECORD TYPE genannt und es werden dann alle Attribute (RECORD ITEMS) in einer COBOL ähnlichen Schreibweise spezifiziert:

```
RECORD NAME is STUDENT
  01 Matr.-Nr.        PICTURE "9(6)"
  01 Studname         TYPE CHARACTER 20
  01 Geb.-datum
     02 Tag           PICTURE "99"
     02 Monat         PICTURE "99"
     02 Jahr          PICTURE "99"
  01 Semester         PICTURE "99"
  01 Studienfach      TYPE CHARACTER 15
  01 Studien-KFZ      PICTURE "X(4)"
```

(2) SETS beschreiben die Beziehungen zwischen den RECORDS. Die Bezeichnung hat nichts gemein mit der mathematischen Menge (= Set), sondern bezeichnet im wesentlichen einen benannten Baum mit zwei Ebenen. Ein SET hat genau einen OWNER-RECORD (Vatersatz) und ggf. mehrere MEMBER RECORDS. Sie werden durch einen Namen gekennzeichnet und durch die beteiligten Sätze spezifiziert. Ein Satz wird AUTOMATIC/ MANUAL in den SET eingefügt und die Mitgliedschaft zu einem SET ist OPTIONAL oder MANDATORY (zwingend).

Beispiel: *(vgl. Schlageter/Stucky (1983), S. 103)*

```
SET NAME IS hoert
ORDER IS SORTED BY DEFINED KEYS
MEMBER IS SV
INSERTION IS AUTOMATIC
RENTENTION IS OPTIONAL
KEY IS ASCENDING SVID
SET SELECTION IS THRU hoert-SV
```

OWNER IDENTIFIED BY Matr.-Nr.

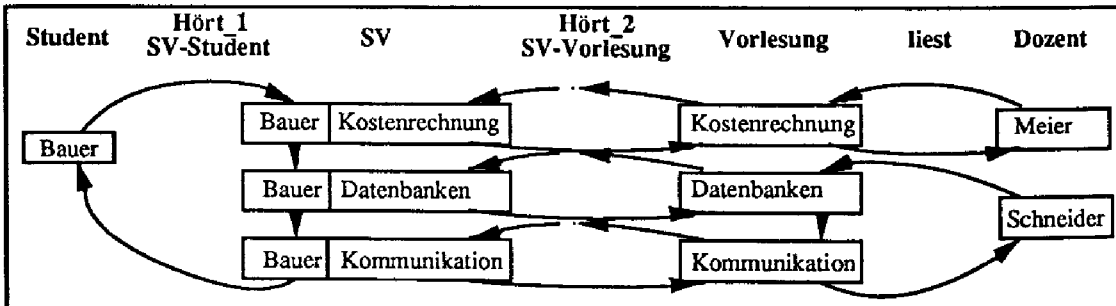
Ein SET hat somit drei Funktionen

- es definiert einen Zugriffspfad vom Owner-Record zu den Member-Records,
- es unterstützt die referentielle Integrität (keine Sohn-Datensätze ohne Vater-Datensätze),
- es trägt die Information, daß zwei Datensätze in einer Beziehung stehen.

Die CODASYL DDL besteht weiterhin aus der SCHEMA-Identifikation und der AREA-Spezifikation (von physischen Speicherbereichen).

Beispiel: (vgl. Martin, J. (1988), S. 181)

```
SCHEMA NAME IS Studium
AREA NAME IS Studbereich
RECORD NAME IS Student
  ...
RECORD NAME IS SV
  01 SVID              PICTURE "9(9)"
  01 Matr.-Nr.         PICTURE "9(6)"
  01 Vorlesungs-Nr.    PICTURE "9(4)"
  ...
RECORD NAME IS VORLESUNG
  01 Vorlesungs-Nr.    PICTURE "9(4)"
  ...
RECORD NAME IS DOZENT
  01 Dozenten-Nr.      PICTURE "9(4)"
  ...
SET NAME IS hoert
  OWNER IS Student
  MEMBER IS SV OPTIONAL AUTOMATIC;
  ASCENDING KEY IS Vorlesungs-Nr.
SET NAME IS Hilfe, ORDER IS SORTED
  OWNER IS Vorlesung
  MEMBER IS SV OPTIONAL AUTOMATIC
  ASCENDING KEY IS Matr.-Nr.
SET NAME IS liest, ORDER IS SORTED
  OWNER IS Dozent
  MEMBER IS Vorlesung OPTIONAL AUTOMATIC;
  ASCENDING IS Vorlesungs-Nr.
```



Abfrage: Geben Sie alle Dozenten des Studenten Bauer aus.

Lösung:

```
Student.Name:=Bauer
find Student using database key Matr.Nr.
find first SV within Hört_1

while statuscode <> 'kein Satz mehr' do
 begin
  find owner within Hört_2

  find owner within liest

  get Dozent
  write Dozent.Name
  find next SV within Hört_1

 end
```

Kommentar:
Gehe anhand der Zeigerstruktur von Hört_1 entlang, bis im Kettrecord SV der Name Bauer zum ersten Mal gefunden wird.
Solange weitere Einträge gefunden werden,
suche die Vorlesung anhand der Struktur von Hört_2,
suche den Dozenten anhand der Struktur von liest,

gib den Dozenten aus und
gehe anhand von Hört_1 zum nächsten Studenten

Abb. 2.1.2/2: Abfragen im Netzwerkdatenbankschema

2.1.3. Datenbank-Sprache für das relationale Modell

Die verbreiteste Datenbanksprache für relationale Datenbanken ist SQL (Structured Query Language). SQL umfaßt neben einer Data Definition Language (DDL) eine Data Manipulation Language (DML) und eine Data Control Language (vgl. Schlageter/Stucky, (1983)).

Bereiche	**Abfragen** **QUERIES**	**Datenmanipulation** **DATA MANIPULATION**	**Datendefinition** **DATA DEFINITION**	**Datenschutz** **DATA CONTROL**
Befehle	SELECT	INSERT UPDATE DELETE	CREATE DROP	GRANT

Abb. 2.1.3/1: SQL- Sprachelemente

(1) Definition des konzeptuellen Schemas

Aufgabe ist die Definition der Basisrelationen mit Angabe der Primärschlüssel, der Attribute und der zulässigen Wertebereiche. Der SQL-Befehl heißt *CREATE*. In Verbindung mit dem Befehlswort *TABLE* wird er folgendermaßen eingesetzt: Eine Tabelle mit den Spalten

Vorlesungsnr, Titel, Zeit und Raum wird erstellt. Mit Hilfe von *CREATE TABLE* wurden die folgenden Tabellen erstellt:

```
CREATE TABLE
VORLESUNG (VORLESUNGSNR int, TITEL char(30), ZEIT char(15),
           RAUM char(10));
```

name	schwerpunkt	fach	sprechstunde	raum	telefon	adresse
Meier	Buchfuehrung	5	Fr. 9-11	C4.224	2303	Paderborn
Meier	Kostenrechnung	5	Fr. 9-11	C4.224	2303	Paderborn
Rubens	Anwendungssysteme	5	Do. 10-11	H5.208	3814	Bielefeld
Schneider	Kommunikation	5	Do. 14-15	B3.332	2807	Paderborn
Schneider	Datenbanken	5	Do. 14-15	B3.332	2807	Paderborn
Willms	Marketing	5	Mi. 9-10	C4.201	2504	Detmold

```
CREATE TABLE
STUDENT (MATRNR int, NAME char(15), SEMESTER int,
         STUDIENFACH char(25), ADRESSE char (15));
```

matrnr	name	semest	studienfach	adresse
3172222	Mueller	10	Wirtschaftswissenschaften	Detmold
3255666	Bauer	8	Wirtschaftsingenieur	Paderborn
3617680	Schmidt	12	Wirtschaftswissenschaften	Paderborn
3627343	Fischer	6	Informatik	Soest

```
CREATE TABLE
VORLESUNG (VORLESUNGSNR int, TITEL char(30),
           ZEIT char(15), RAUM char(10));
```

vorles	titel	zeit	raum
5325	Kostenrechnung	Di. 9-11	C1
5328	Datenbanken	Do. 14-16	H5.324
5336	Marketing	Mi. 9-11	H5.324

```
CREATE TABLE
VORLESUNG (MATRNR int, VORLESUNGSNR int);
```

matrnr	vorles
3172222	5328
3255666	5325
3255666	5336
3272222	5325
3627343	5328

```
CREATE TABLE
VORLESUNG (NAME char(30), VORLESUNGSNR int);

+------------+------+
|name        |vorles|
+------------+------+
|Meier       |  5325|
|Schneider   |  5328|
|Willms      |  5336|
+------------+------+
```

Mit dem Befehl *EXPAND TABLE* wird eine Tabelle erweitert.

```
EXPAND TABLE Dozent (vorname char (12));
```

SQL ermöglicht nur eine rudimentäre Angabe der Wertedomänen der einzelnen Attribute, etwa indem Typen wie INTEGER, CHARACTER zugeordnet werden.

Mit dem Befehl *DROP TABLE* wird eine Tabelle gelöscht.

```
DROP TABLE Vorlesung;
```

(2) Definition des externen Schemas

Benutzersichten werden als virtuelle Relationen über den Basisrelationen erstellt, d. h. sie werden anders als die Basisrelationen nicht gespeichert, sondern als Ausschnitt jeweils aktuell erstellt.

```
DEFINE VIEW wirschaftsinformatik (matrnr, name, semester) AS
(SELECT matrnr, name, semester
 FROM student
 WHERE studienfach = "Wirtschaftsinformatik");
```

Die DML baut auf den Grundoperationen des relationalen Modells auf: Selektion, Projektion, Vereinigung, Schnitt und Verbund. Die einzelnen Werte der Relationen bleiben dabei unverändert.

SELECT	*FROM*	*WHERE*
Attribute	*Tabelle*	*Bedingung*

Der Befehl zum Abfragen von Daten heißt *SELECT*. Mit *SELECT * FROM Vorlesung* werden alle Spalten der Tabelle Vorlesung angezeigt.

```
SELECT * FROM vorlesung;

+------+------------------------------+--------------+----------+
|vorles|titel                         |zeit          |raum      |
+------+------------------------------+--------------+----------+
|  5325|Kostenrechnung                |Di. 9-11      |C1        |
|  5328|Datenbanken                   |Do. 14-16     |H5.324    |
|  5336|Marketing                     |Mi. 9-11      |H5.324    |
+------+------------------------------+--------------+----------+
```

Sollen nur bestimmte Spalten ausgegeben werden, so sind die Bezeichnungen der Spalten anzugeben.

```
SELECT Name, Sprechstunde, Raum, Telefon
FROM Dozent
WHERE Adresse = 'Paderborn'

+--------------------+--------------------+---------+---------------+
|name                |sprechstunde        |raum     |telefon        |
+--------------------+--------------------+---------+---------------+
|Meier               |Fr. 9-11            |C4.224   |2303           |
|Meier               |Fr. 9-11            |C4.224   |2303           |
|Schneider           |Do. 14-15           |B3.332   |2807           |
|Schneider           |Do. 14-15           |B3.332   |2807           |
+--------------------+--------------------+---------+---------------+
```

Bedingungen können nach dem Befehlswort *WHERE* angegeben werden. Bei Vergleichen sind die Operatoren =, <>, <, <=, >, >=, *between* und *in* zugelassen.

```
SELECT * FROM vorlesung WHERE titel = 'Kostenrechnung';

+------+-----------------------------+--------------+----------+
|vorles|titel                        |zeit          |raum      |
+------+-----------------------------+--------------+----------+
|  5325|Kostenrechnung               |Di. 9-11      |C1        |
+------+-----------------------------+--------------+----------+
```

```
SELECT Name, Semester, Studienfach
FROM Student
WHERE Semester > 7;

+--------------------+------+-----------------------------------+
|name                |semest|studienfach                        |
+--------------------+------+-----------------------------------+
|Mueller             |     9|Wirtschaftswissenschaften          |
|Schmidt             |    11|Wirtschaftswissenschaften          |
+--------------------+------+-----------------------------------+
```

Weitere Operatoren sind *avg* (Bestimmung des Mittelwertes), *count* (Anzahl von Feldelementen), *max* (Maximum), *min* (Minimum), *not* (Negation) und *sum* (Summe von Spalten).

```
SELECT avg(semester)
FROM student
WHERE studienfach = 'Wirtschaftswissenschaften';

+-----------+
|col1       |
+-----------+
|     10.000|
+-----------+
```

Mit *like* ist die Suche nach Zeichenketten möglich. Dabei ist die Benutzung der Jokerzeichen % für eine beliebige Zeichenkette und _ für ein einzelnes beliebiges Zeichen möglich.

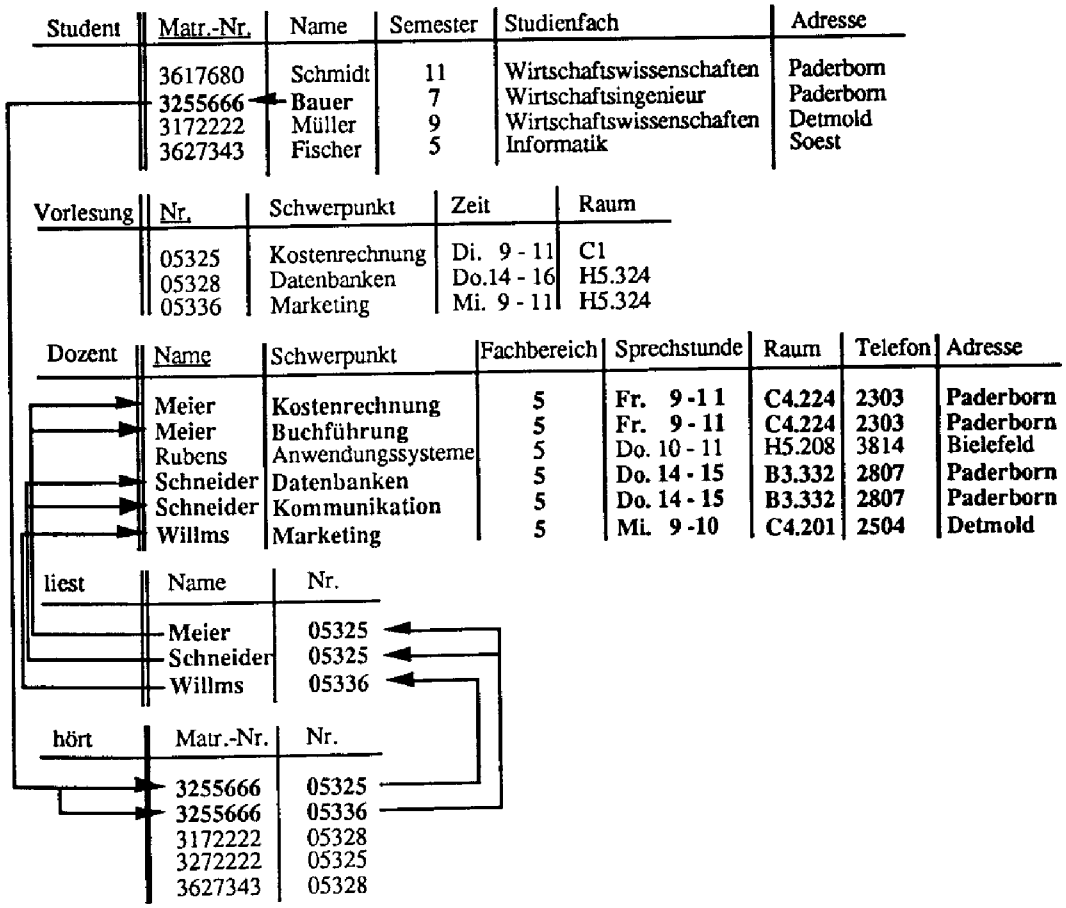
```
SELECT Name, Sprechstunde, Raum
FROM Dozent
WHERE Sprechstunde like 'Do%';

+-------------------+-------------------+----------+
|name               |sprechstunde       |raum      |
+-------------------+-------------------+----------+
|Rubens             |Do. 10-11          |H5.208    |
|Schneider          |Do. 14-15          |B3.332    |
|Schneider          |Do. 14-15          |B3.332    |
+-------------------+-------------------+----------+
```

Abfragen über mehrere Tabellen werden über gleiche Attributausprägungen und geschachtelte Abfragen realisiert. Dabei kann der Fall auftreten, daß die entsprechenden Spalten die gleiche Bezeichnung besitzen.

```
select name from liest
where vorlesungsnr= (
     select vorlesungsnr from hoert
     where matrnr = (
     select matrnr from student where name = 'Schmidt'))

+--------------------+
|name                |
+--------------------+
|Schneider           |
+--------------------+
```



Student	Matr.-Nr.	Name	Semester	Studienfach	Adresse
	3617680	Schmidt	11	Wirtschaftswissenschaften	Paderborn
	3255666	Bauer	7	Wirtschaftsingenieur	Paderborn
	3172222	Müller	9	Wirtschaftswissenschaften	Detmold
	3627343	Fischer	5	Informatik	Soest

Vorlesung	Nr.	Schwerpunkt	Zeit	Raum
	05325	Kostenrechnung	Di. 9 - 11	C1
	05328	Datenbanken	Do.14 - 16	H5.324
	05336	Marketing	Mi. 9 - 11	H5.324

Dozent	Name	Schwerpunkt	Fachbereich	Sprechstunde	Raum	Telefon	Adresse
	Meier	Kostenrechnung	5	Fr. 9 - 11	C4.224	2303	Paderborn
	Meier	Buchführung	5	Fr. 9 - 11	C4.224	2303	Paderborn
	Rubens	Anwendungssysteme	5	Do. 10 - 11	H5.208	3814	Bielefeld
	Schneider	Datenbanken	5	Do. 14 - 15	B3.332	2807	Paderborn
	Schneider	Kommunikation	5	Do. 14 - 15	B3.332	2807	Paderborn
	Willms	Marketing	5	Mi. 9 -10	C4.201	2504	Detmold

liest	Name	Nr.
	Meier	05325
	Schneider	05325
	Willms	05336

hört	Matr.-Nr.	Nr.
	3255666	05325
	3255666	05336
	3172222	05328
	3272222	05325
	3627343	05328

Abfrage: **Geben Sie alle Dozenten des Studenten Bauer aus**

Lösung:

```
Select Name from liest
where Nr. =
        ( select Nr. from hört
        where Matr. Nr. =
                (Select Matr.Nr. from Student
                where Name = 'Bauer'))
```

Abb. 2.1.3/2: Abfragen im relationalen Datenbankschema mit SQL

(3) Manipulation der Daten

Der Befehl *insert* dient zum Einfügen eines neuen Datensatzes. Dazu muß die Reihenfolge der Felder und der korrespondierenden Inhalte der Reihenfolge der Spalten in der Tabelle entsprechen.

```
INSERT INTO Vorlesung(VorlesungsNr, Titel, Zeit, Raum)
VALUES( 05329, 'Kommunikationssysteme', 'Fr. 14-16', 'C2' );

SELECT * FROM vorlesung;

+------+------------------------------+--------------+----------+
|vorles|titel                         |zeit          |raum      |
+------+------------------------------+--------------+----------+
|  5325|Kostenrechnung                |Di. 9-11      |C1        |
|  5328|Datenbanken                   |Do. 14-16     |H5.324    |
|  5329|Kommunikationssysteme         |Fr. 14-16     |C2        |
|  5336|Marketing                     |Mi. 9-11      |H5.324    |
+------+------------------------------+--------------+----------+
```

Eine Aktualisierung der Feldinhalte ist mit dem Befehl *update* möglich.

```
UPDATE Student
SET Semester = Semester + 1;

SELECT Name, Studienfach, Semester FROM Student;

+------------------+-----------------------------------+------+
|name              |studienfach                        |semest|
+------------------+-----------------------------------+------+
|Mueller           |Wirtschaftswissenschaften          |    10|
|Bauer             |Wirtschaftsingenieur               |     8|
|Schmidt           |Wirtschaftswissenschaften          |    12|
|Fischer           |Informatik                         |     6|
+------------------+-----------------------------------+------+
```

Mit dem Befehl delete werden Datensätze gelöscht.

```
DELETE FROM vorlesung where (name = ´Kommunikationssysteme) and
(Raum = ´C2´);

SELECT * FROM vorlesung;

+------+------------------------------+--------------+----------+
|vorles|titel                         |zeit          |raum      |
+------+------------------------------+--------------+----------+
|  5325|Kostenrechnung                |Di. 9-11      |C1        |
|  5328|Datenbanken                   |Do. 14-16     |H5.324    |
|  5336|Marketing                     |Mi. 9-11      |H5.324    |
+------+------------------------------+--------------+----------+
```

(4) Datenschutz

Der Besitzer einer Datenbank kann die Zugriffsrechte auf seine Tabellen mit dem Befehl grant vergeben. Dieser Befehl unterscheidet folgende Teilzugriffsrechte:

- **delete:** Löschen von Datensätzen
- **insert:** Einfügen von Datensätzen
- **select:** Abfrage von Datensätzen
- **update:** Modifizieren von Datensätzen
- **all privileges:** alle Möglichkeiten erlauben

Den Benutzern Dresing, Kern und Walter sollen alle Zugriffsrechte auf die Relation Vorlesung gegeben werden.

```
GRANT ALL PRIVILEGES ON Vorlesung TO Dresing, Kern, Walter;
```

In Verbindung mit dem Zugriffsrecht update können einzelne Spalten, z. B. die Spalten Zeit und Raum der Relation Vorlesung freigegeben werden.

```
GRANT UPDATE (Zeit, Raum) TO Vorlesung Kern;
```

(5) Erweiterung von SQL

SQL ist zwar ein ANSI bzw. ISO-Standard (ISO 9075 aus dem Jahre 1986), doch haben sich inzwischen eine Reihe von Dialekten entwickelt, die den Komfort und die Mächtigkeit steigern.

Attributtypen	- Datums-/ Zeit-Datentyp
Um Integritätsregeln erweiterte DDL	- dynamische Integrität - referentielle Integrität - Entity-Integrität
Erweiterte DML	- Join

Abb. 2.1.3/3: Erweiterungen im SQL 2-Standard

2.2. Kommunikationskomponente

Das Kommunikationssystem hat die Aufgabe, den Anwenderprogrammen bzw. Benutzertypen auf der einen Seite und dem Datenbanksystem auf der anderen Seite eine Schnittstelle anzubieten. Diese soll die logische und physische Heterogenität der verwendeten DV-Komponenten ausgleichen und den Nachrichtenaustausch zwischen einer Vielzahl von (intelligenten und nicht-intelligenten) Datenstationen und den Datenbankkernen abwickeln.

Funktion	Teilaufgaben
Kommunikationsprotokoll	- Vereinheitlichung der Kommunikationsprotokolle der Datenstationen gegenüber dem Datenbanksystem,
Nachrichtenaustausch	- Datenfernübertragung unabhängig von ggf. heterogenen Netzen/ Diensten
Kommunikationssteuerung	- Steuerung der Kommunikation in Abhängigkeit von Prioritäten, Betriebsmittelverfügbarkeit, Kommunika-tionskosten
Datenschutz	- Sicherung von Zugangsregelungen
Benutzeroberfläche	- Bereitstellung einer einheitlichen Benutzer-Kommunikationsoberfläche auch für entfernte Benutzer

Abb. 2.2/1: Funktionen der Kommunikationskomponente

2.3. Steuerungskomponente

Die Steuerungskomponente koordiniert die Aktivitäten von Datenkommunikationssystem und dem Datenbank-Managementsystem, d. h. dem Datenbankkern und der zugehörigen Datenbanksprache.

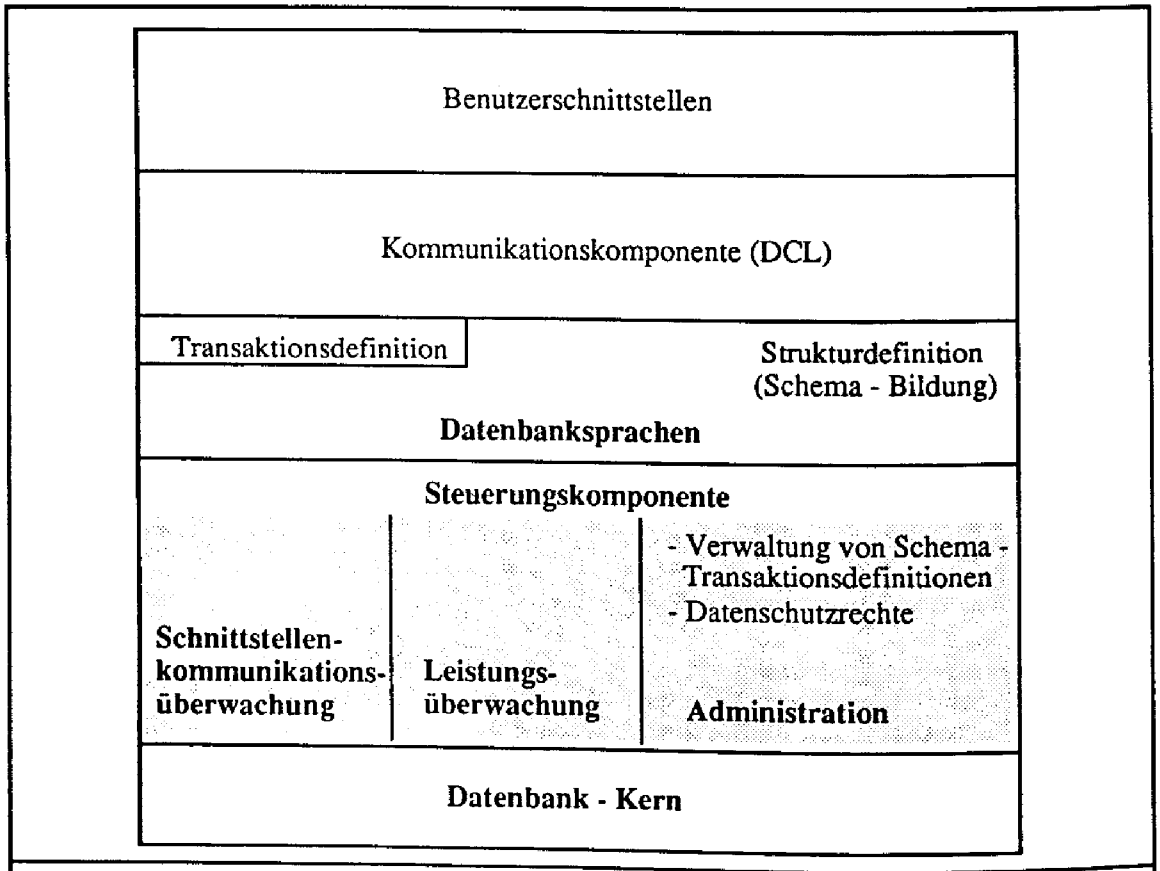


Abb. 2.3/1: Steuerungskomponente als zentraler Bestandteil eines Datenbanksystems

Mit ihren Bestandteilen zur Verwaltung und Überwachung von Kommunikations-, Datenzugriffs- und Datensicherheitsprozeduren bestimmt diese Komponente entscheidend das Leistungsverhalten einer Datenbank.

2.4. Datenbank-Kern (Kernel)

Funktionen

Der Datenbank-Kern realisiert die physische Speicherung (= internes Schema) und ist für den Zugriff auf die Datenbasis verantwortlich. Unter anderem gehören dazu:

- die Darstellung von Attributwerten in Feldern
- der Satz-Aufbau
- die Definition der Zugriffspfade
- die Satzanordnungen

Neben den Speichermedien mit entsprechenden technisch-physischen Eigenschaften (Speicherkapazität, Zugriffsgeschwindigkeit, Adressierbarkeit) gehören zum Datenbankkern die

physische Beschreibungssprache, die die physische Struktur und Anordnung der Daten auf den Speichergeräten spezifiziert. (vgl. Martin (1988), S. 155ff.))

Datenmodell	Datenbanksystem	Ausprägung der physikalischen Beschreibungssprache
Hierarchisches Datenmodell	IBM IMS/DL1	Physical Data Base Description (Physische DBD)
Netzwerk-Datenmodell	CODASYL	Device Media Control Language Data Storage Description Language
	Siemens UDS	Storage Structure Language

Abb. 2.4/1: Übersicht der zu den Datenmodellen verfügbaren DBS und physikalischen Beschreibungssprachen

Speicher- und Zugriffsorganisation

Die Speicherorganisation hängt zunächst einmal von den technischen Eigenschaften der Speichermedien ab. Man unterscheidet üblicherweise sequentielle, blockadressierbare und direkt adressierbare Speichermedien (vgl. Fischer/Herold (1992)). Heute werden die operativ eingesetzten Datenbanken fast ausschließlich auf blockadressierbaren Speichermedien großer Kapazität (vornehmlich Plattenspeichern) verwaltet.

Die logische Organisation eines Datenbestandes wird entweder vom Betriebssystem oder vom Datenbanksystem bestimmt. Im ersten Fall kann das Datenbanksystem die vom Betriebssystem gesetzten Grenzen nicht überwinden.

Logisches Element	Erläuterung	vom Betriebssystem bestimmt	vom Datenbanksystem bestimmt
Zeichen	kleinstes logisch speicherbares Element	x	
Datenfeld	kleinstes logisch zugreifbares Element		x
Datensatz	logisch zusammengehörende Felder		x
Datenblock	logisch oder physisch in einem Arbeitsgang verarbeitetes Element	x	x
Datei	Ansammlung von logischen Sätzen eines Typs		x

Abb. 2.4/2: Elemente der logischen Organisation

Beispielsweise erlaubt das Betriebssystem nur eine bestimmte physische Länge von Feldern, Sätzen oder Blöcken oder es fordert bestimmte Mindestlängen. Im zweiten Fall definiert das Datenbanksystem die zulässigen logischen Konfigurationen.

Die physischen Organisationsformen eines Datenbestandes lassen sich verkürzt in logisch sortierte und nicht sortierte Formen einteilen. Bei logisch sortierten Formen wird ein Datensatz durch ein oder mehrere Merkmale (sogenannte Schlüsselfelder) logisch gekennzeichnet, mit deren Hilfe der Zugriff oder die Speicherung erfolgt.

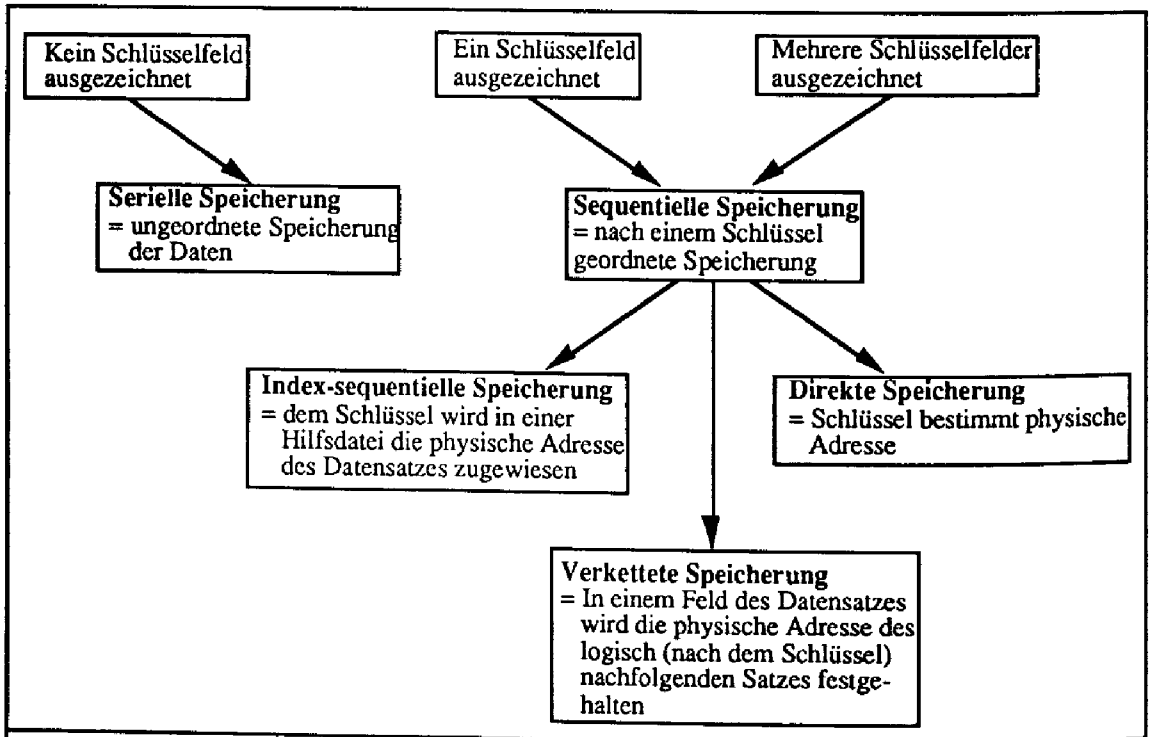


Abb. 2.4/3: Verkürzter Überblick über Speicherorganisationsformen

Bei der physischen Speicherung in Datenbanken ist normalerweise eine Sortierung nach mehr als einem Schlüssel notwendig, so daß entweder mehrere Hilfsdateien (sogenannte Invertierte Listen) verwendet werden oder eine Verkettung nach mehreren Schlüsseln vorzunehmen ist. In einer invertierten Liste findet sich zu einer bestimmten Ausprägung eines Sekundärschlüssels entweder die physische Adresse aller dazu passenden Sätze oder aber die Primärschlüssel der jeweiligen Datensätze. Mit deren Hilfe kann dann über die Indextabellen auf die jeweiligen Datensätze zugegriffen werden.

Auch bei der physischen Speicherorganisation besteht die Möglichkeit, daß das Betriebssystem dem Datenbanksystem Grenzen auferlegt.

Betriebs-system	Zugriffs-technik	Speicherorganisationsform	Physische Grenzen
MS-DOS	FAT (File Allocation Table)	- physikalischer Speicherraum - index-sequentiell - Hilfstabellen auf der äußersten Spur des Datenträgers - unsortierte lineare Listen - chaotische Speicherung	- kleinste Belegungseinheit Cluster von 4 KB - Dateinamen 11 Stellen - Pfadlänge 64
OS/2	HPFS (High Performance File System)	- physikalischer Speicherraum - index-sequentiell - Hilfstabellen in der Nähe der mittleren Spur des Datenträgers - sortierte Bayer-Bäume - Hinweise auf freie Speicherplätze in der Nähe der Dateien	- kleinste Belegungseinheit Sektor von 256 Bytes - Dateinamen 254 Stellen - Pfadlänge 260
UNIX	i-node	- Hilfstabellen für jede Datei mit direkten und indirekten Blockangaben	- kleinste Belegungseinheit 512 Byte (bzw. ein Vielfaches von 512 Byte) - Dateilänge: 14 Zeichen (AT&T-UNIX) bzw. 256 Zeichen (Berkley-UNIX)
IBM/MVS	VSAM (Virtual Sequential Access Method)	- virtueller Speicherraum - Sequentielle Ordnung nach Schlüssel, Speicherungszeitpunkt oder nach relativer Ordnung zum Dateianfang	
	ISAM (Index-Sequential Access Method)	- index-sequentiell	
	HISAM (hierarchical ISAM)	- index-sequentiell - mehrstufige Indizes	

Abb. 2.4/4: Speicherorganisation bei ausgewählten Betriebssystemen

Beispielsweise ist es bei einer virtuellen Speicherorganisation für die Performance eines Datenbanksystems entscheidend, ob dieses die vom Betriebssystem verwendete Blockgröße beim Datenaustausch zwischen Arbeitsspeicher und Peripheriespeichern beeinflussen kann oder nicht. Die modernen Datenbanksysteme bemühen sich, die Grenzen der Betriebssysteme zu umgehen.
Die existierenden Datenbanksysteme unterscheiden sich hinsichtlich der Speicherorganisation in ihrer Leistungsfähigkeit:

Vordefinierte Joins: Kann das DBS die Speicherorganisation nach vordefinierten Abfragestrukturen optimieren?

Clustering: Kann das DBS Sätze, auf die häufig gemeinsam zugegriffen wird, physisch benachbart abspeichern?

Paging: Kann das DBS Sätze, auf die häufig gemeinsam zugegriffen wird, in der gleichen Seite positionieren?

3. Verbreitete Datenbank-Systeme

Produkt	Anbieter	Daten-modell	Betriebssysteme	Daten-bank-sprachen (z. B. SQL)	Bezeichnung der systemeigenen Programmier-sprache
ADABAS	Software AG	relational-orientiert	VAX/VMS, BS 2000, UNIX, VSE, VM/ CMS, OS/2, IBM/ MVS, MS-DOS-Windows 3	SQL	Natural
IMS	IBM	hierarchisch	IBM/MVS	DL/1	
DB2	IBM	relational	IBM/MVS	SQL	
EMPRESS	Empress High Tech Software GmbH	relational	UNIX, VAX/VMS, MS-DOS	SQL	M-Builder
INGRES	Ingres Corp.	relational	VAX/VMS, IBM/MVS, UNIX	SQL	Quel
ORACLE	ORACLE Corp.	relational	VAX/VMS, IBM/ MVS, BS 2000, VISNES, UNIX, OS/2, MS-DOS (Client), Macintosh (Client)	SQL	SQL*Forms, SQL*Report
SYBASE	Sybase	relational	UNIX, VMS, OS/2, MS-DOS	SQL	SQR
UNIFY 2000	Unify	relational	UNIX	SQL	Accell-SQL
SQL-Server	Microsoft	relational	OS/2	SQL	
UDS	Siemens AG	Netzwerk/ relational/ hierarchisch	BS 2000,	SQL, QBE	IQL
VAX DBMS	DEC	Netzwerk	VMS		
INFORMIX	INFORMIX Software GmbH	relational	VAX/VMS, UNIX, OS/2, MS-DOS	SQL	Informix-4GL

Abb. 3/1: Überblick über kommerziell einsetzbare Datenbanksysteme

4. Beurteilungskriterien

Sicht		Gesichtspunkte	Mögliche Alternativen
Logische Sicht	Datenmodell	1. Welche Datenobjekt-Typen werden unterstützt?	- einfache - komplexe
		2. Welche Datenstruktur-Typen sind zugrundegelegt?	- hierarchisch - Netze - Tabellen - Klassen
		3. Welche Datenoperator-Typen werden unterstützt?	- syntaktische - temporale - semantische - dynamische
		4. Welche Integritätsbedingungen werden unterstützt?	- statische - dynamische
	Datenbank-sprache	1. Mächtigkeit der DDL/DML/ DCL	- genormte Sprachen (SQL) - eigene Datenbanksprache - Benutzerkomfort
		2. Verbindung mit Software-Entwicklungstools	- CASE - 4 GL Programmiersprache - Schnittstellen zu 3 GL (Programmiersprachen)
		3. Verfügbarkeit eines Data-Dictionaries	
Technische Sicht	System-architektur	1. Hardwareanforderungen an - Arbeitsspeicher - Peripheriespeicher - Kommunikationseinrichtungen?	- Nutzung unterschiedlicher Speichermedien
		2. Auf welchen Betriebssystemwelten ist das DBS lauffähig?	- herstellerübergreifend - heterogen
		3. Welche Netzstrukturen und -protokolle werden unterstützt?	- ISO/OSI - TCP/IP und andere
	Performance	1. Datenhaltung	- Datenkomprimierung - Anzahl zulässiger Relationen - Anzahl zulässiger Sätze und Attribute je Relation - Techniken zur Zugriffsoptimierung
		2. Rechen- und Kommunikationszeiten	
	Unabhängigkeit	1. Umsystemabhängigkeit	- Schnittstellen zu Programmiersprachen; anderen Datenbanksystemen
		2. Betriebssystemabhängigkeit	- Werden Paging-Prozeduren vom Datenbank-System oder vom Betriebssystem gesteuert? - Werden Datenzugriffe unabhängig vom Betriebssystem durchgeführt?
	Sicherheit	1. Fehlerprozeduren	- Recovery-Routinen bei Systemabsturz
		2. Datenschutzprozeduren	

Abb. 4/1: Gesichtspunkte zur Beurteilung eines Datenbanksystems

Kapitel 4: Verteilte Datenbanksysteme

1. Konzepte

1.1. Kennzeichnung

Von einer „verteilten Datenbank" (distributed database) spricht man, wenn eine logisch einheitliche Datenbank physisch auf mehrere Speicherorte zu verteilen ist. Die Verteilung kann lokal an einem Standort über mehrere Rechner erfolgen und verwendet dann lokale Netzwerke (Local Area Networks - LAN) zur Kommunikation zwischen Nutzer und Datenbeständen bzw. Datenbestand zu Datenbestand. Alternativ kann die Verteilung regional zwischen verschiedenen Standorten erfolgen und Datenfernübertragungsnetze (Wide Area Networks - WAN) nutzen.

Entscheidend ist, daß dem Benutzer der Grad der räumlichen DV-Verteilung verborgen bleibt. Die „verteilte Datenbank" begegnet ihm logisch einheitlich, die Ansprache bestimmter DV-Komponenten ist unnötig.

Die Abkehr von zentralisierten „integrierten Datenbanksystemen" hat DV-technische und organisatorische Vorteile, verursacht auf der anderen Seite aber neuartige Probleme, die bei zentralisierten Datenbanken nicht auftreten. Diese resultieren aus den Zusammenhängen zwischen den verteilten Datenbeständen und den Anforderungen an deren logische Integrität.

Abgrenzung

Datenbanken werden in aller Regel räumlich verteilten Nutzern über Kommunikationsnetze zugänglich gemacht. Bei zentralisierten Datenbanken existiert der Datenbestand für eine bestimmte Nutzergruppe an einem definierten (zentralisierten) Ort.

	DATENBANKEN		
	Zentrale	**Vernetzte**	**Verteilte**
Nutzer	dezentral	dezentral	dezentral
Datenbestand	physisch zentral logisch zentral	physisch dezentral logisch dezentral	physisch dezentral logisch dezentral
	Programmunabhängigkeit	Programmunabhängigkeit	Programmunabhängigkeit/Verteilungsunabhängigkeit

Abb. 1.1/1: Abgrenzung von zentralen, vernetzten und verteilten Datenbanken

Bei vernetzten Datenbanken kann der Nutzer auf mehrere, lokal verteilte Datenbestände zugreifen. Die Datenbestände sind dabei jeweils physisch zu adressieren und logisch anzusprechen; d. h. dem Anwender ist die lokale Verteilung bewußt.

Bei verteilten Datenbanken wird der Anwender von der lokalen Verteilung der Daten demgegenüber nicht berührt, das Datenbanksystem erscheint für ihn als eine physische und

logische Einheit. Man spricht von der sogenannten "Verteilungsunabhängigkeit" (distribution transparency). Etwas paradox ist es, daß in der Literatur auch Systeme als verteilt betrachtet werden, die Daten lokal nur an einem Ort halten.

	Merkmal	Vernetzte Datenbank	Verteilte Datenbank
Nutzer	regionale Verteilung	dezentral	dezentral
	externes Schema	heterogen	heterogen
Datenlokalität	regionale Verteilung	dezentral	dezentral
	logisches Modell	heterogen	homogen

Abb. 1.1/2: Abgrenzung von zentralen, vernetzten und verteilten Datenbanken

Der entscheidende Unterschied zwischen einer vernetzten und verteilten Datenbank liegt somit in dem einheitlichen logischen Bild einer verteilten Datenbank.

Hinsichtlich der Transparenz einer dezentralen Datenbank unterscheidet man drei Stufen:

Bei der Systemunabhängigkeit muß der Nutzer den Ort und den Ausschnitt der Daten spezifizieren, mit dem er zu arbeiten beabsichtigt (vernetzte Datenbank).

Bei der Ortsunabhängigkeit gibt der Nutzer den Ausschnitt der Daten an, nicht jedoch deren lokale Verteilung.

Bei der Dekompositionsabhängigkeit ist schließlich der höchste Grad von Verteilung erreicht; der Nutzer muß weder den Datenausschnitt noch den Ort der Daten spezifizieren.

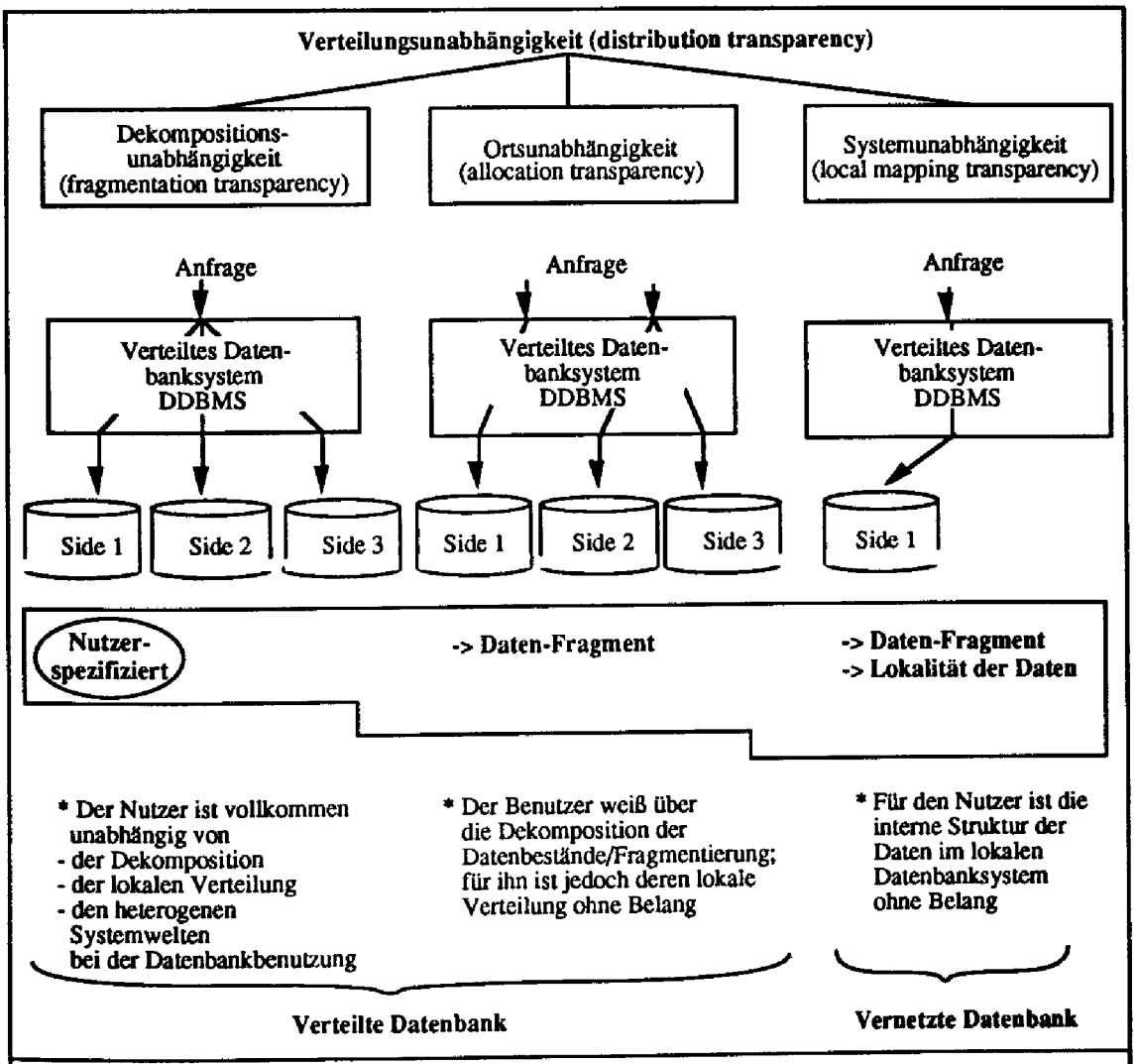


Abb. 1.1/3: Grade der Transparenz bei dezentralen Datenbanken (vgl. Ceri/Pelagatti (1984), S. 47)

Arten

Zur Zeit existieren verschiedene Typen spezialisierter verteilter Datenbanken (meist nur als Prototypen); eine einheitliche Struktur hat sich noch nicht herausgebildet.

Die großen Datenbankanbieter sind alle bemüht, ihre DBS entsprechend zu modifizieren. Modifizierte zentralisierte Systeme ergänzen die Komponenten der zentralen Datenbank um Elemente für die Kommunikation und die Datenbankadministration:

-> database management component (DB) des zentralen Datenbankservers
-> data communication component (DC) als logisch unabhängige Schnittstelle
-> data dictionary (DD) zur übergreifenden Verwaltung der Datenstrukturen
-> distributed database component (DDB), die auf den einzelnen Client-Rechnern läuft

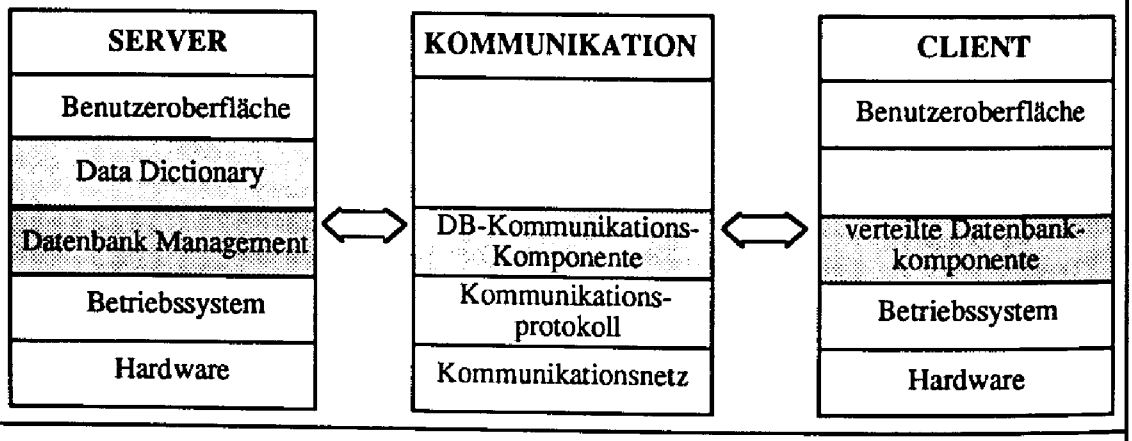


Abb. 1.1/4: Komponenten eines verteilten Systems (modifizierte zentralisierte Systeme)

Bei homogenen Systemen sind alle verteilten Komponenten eines Typs gleich ausgelegt, d. h. Datenschemata und Datenmodell sind auf allen beteiligten Rechnern gleich (z. B. immer relational). Demgegenüber verwenden heterogene Systeme unterschiedliche Datenschemata (Netzwerk, hierarchisch, relational).

Nach dem Grad der Transparenz kann man unterscheiden:

Föderierte Systeme stehen (für den Benutzer) explizit unter lokal getrennter Verwaltung. Es sind häufig nur Leseoperationen auf entfernte Knoten zulässig. Die Datenbestände werden nicht zentral verwaltet, haben jedoch häufig logisch die gleiche Struktur (identisches Datenmodell).

Räumlich getrennte, dezentralisierte Systeme werden (vom Benutzer) über einen dezidierten lokalen Prozessor betreten und machen die Verteilung der Daten über die Unterschiede in den Transaktionszeiten für lokale und entfernte Zugriffe deutlich. Die Verteilung berücksichtigt explizit die Zugriffseigenschaften (langsamer) WAN und neigt daher zu umfassender Datei-Replikation (besonders sich selten ändernder Dateien).

Cluster-Systeme erscheinen (für den Benutzer) als ein System. Transaktionen und Daten werden so auf die Knoten verteilt, daß die Systemverwendung optimiert wird. Der Benutzer adressiert seine Transaktionsanforderungen an das Gesamtsystem, nicht an einen einzelnen Prozessor.

Hinsichtlich des Integrationsgrades des Datenmodells kann differenziert werden (vgl. Dudler (1986), S. 35f):

Vollständig integriert verteilte Systeme sind dadurch charakterisiert, daß alle Daten und Konsistenzbedingungen in einem einzigen globalen Datenschema beschrieben werden.

Föderativ verteilte Systeme sind dadurch gekennzeichnet, daß eine übergreifende Kontrolle über das gesamte Datenschema fehlt und neben lokalen Datenschemata diverse übergeordnete Schemata für den Datenverkehr mit den dezentralen Arbeitsstationen existieren.

1.2. Aufgaben

DV-technische Ziele beim Streben nach verteilten Datenbank-Systemen liegen vornehmlich in den Bereichen Zuverlässigkeit, Verfügbarkeit, Elastizität und Effizienz.

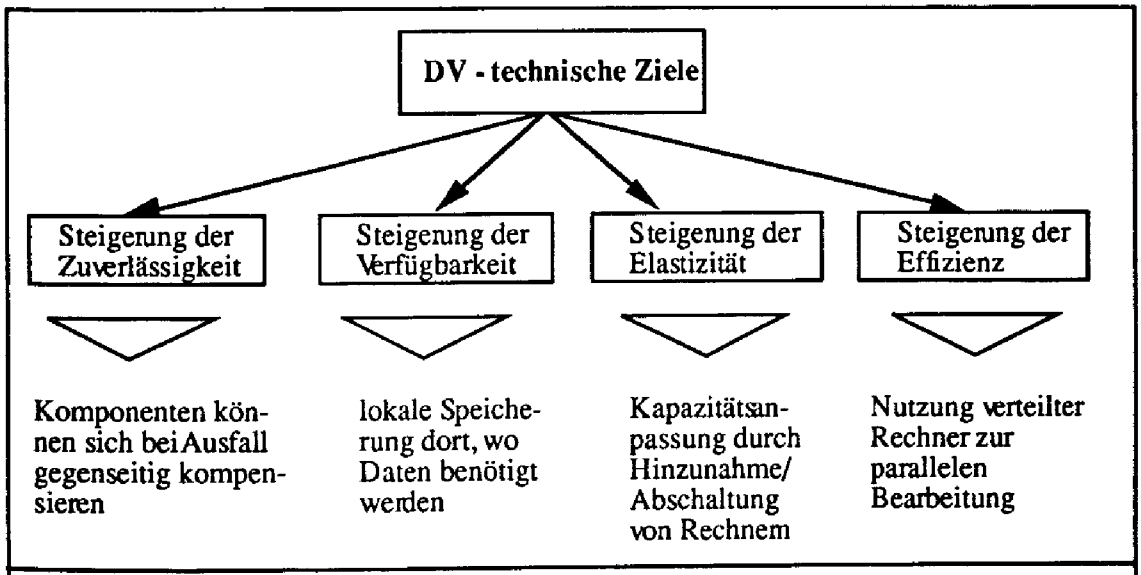


Abb. 1.2/1: DV-technische Ziele einer verteilten Datenbank

Eine Steigerung der Zuverlässigkeit wird dadurch erreicht, daß beim Ausfall eines Rechners andere diesen ersetzen können. Die höhere Fehlerwahrscheinlichkeit durch die größere Zahl von Systemkomponenten wird kompensiert durch die geringere Auswirkung des einzelnen Fehlers. Eine Steigerung der Verfügbarkeit wird erreicht, da Daten dort regional gespeichert werden können, wo sie gebraucht werden. Dies führt zu einer Senkung der Kommunikationsnotwendigkeit und somit zu einer Verringerung der Kommunikationskosten. Die Elastizität wird erhöht, da durch Hinzunahme/Abschalten neuer Rechnerknoten die Rechner- und Speicherkapazität schnell angepaßt werden kann.

Schließlich erfolgt eine Effizienzsteigerung, da mehrere Rechner parallel an einer Abfrage arbeiten können (Lastverteilung und Parallelität).

Organisatorische Ziele einer Verteilung liegen darin, die Datenbestände lokal dort zu halten, wo sie benötigt werden.

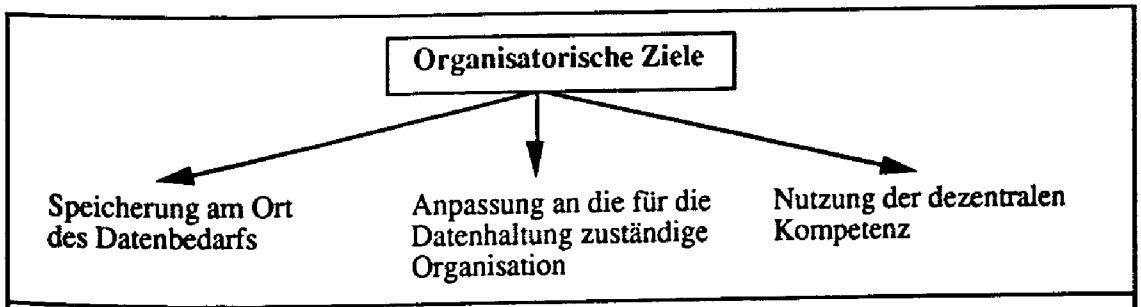


Abb. 1.2/2: Organisatorische Ziele einer verteilten Datenbank

Die organisatorische Anpassung, d. h. die Verteilung der Daten auf die Organisationseinheiten hängt entscheidend von der Kompetenz und den Anforderungen der Nutzer ab. Die Organisationseinheiten sind für dessen Struktur bzw. für die Verbindung dezentraler, bisher isoliert gehlatener Datenbestände zu einem integrierten Datenbankkonzept zuständig.

1.3. Gestaltungsprobleme

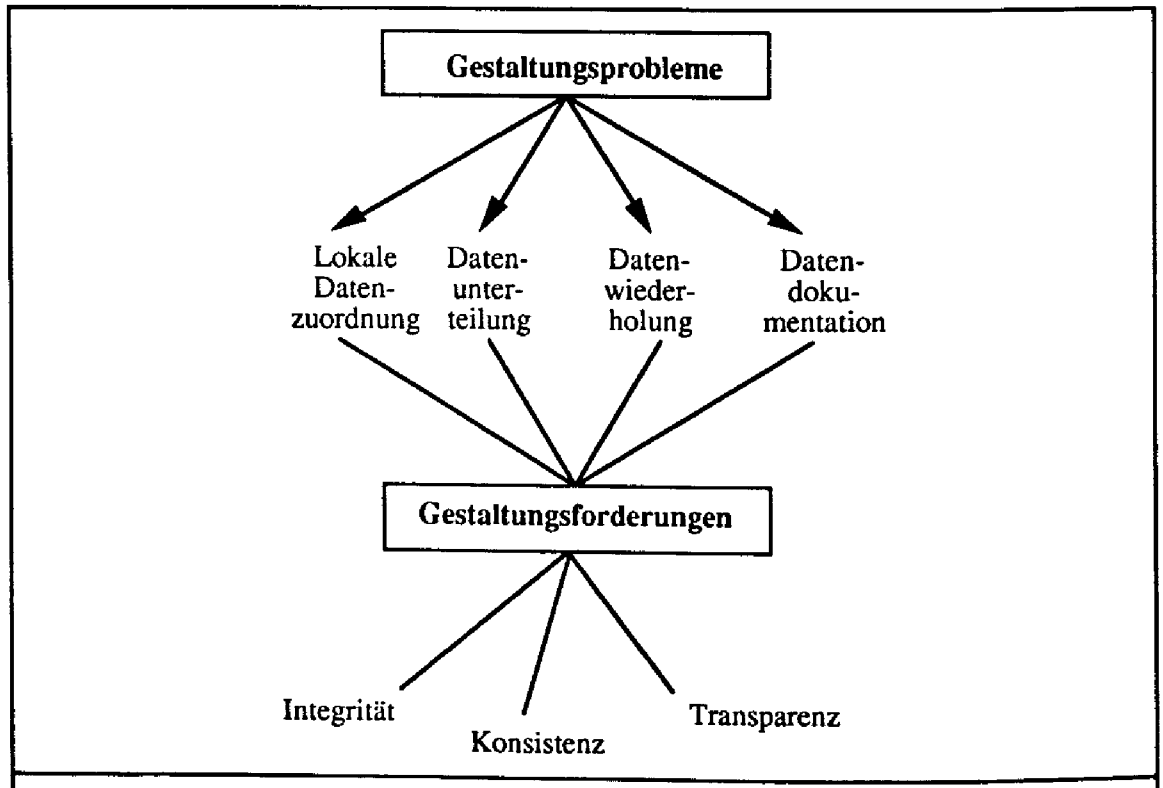


Abb. 1.3/1: Gestaltungsprobleme bei verteilten Datenbanken

Datendokumentation

In einem verteilten Datenbanksystem kommt der Aufgabe der Datendokumentation eine gesteigerte Bedeutung zu:

Zum einen muß die Namensvergabe und -verwaltung der Objekte und Relationen im Gesamtsystem organisiert werden. Dazu muß das Data Dictionary Angaben enthalten über

- das globale Datenschema und dessen Integritätsbedingungen,
- die Kennzeichen der Verteilung im einzelnen über die Fragmentierung, die Allokation, die physische Speicherung in jedem Knoten und über Replikationen.

Das Data Dictionary verliert seine passive Rolle als Lexikon, die es in einem zentralen DBS hat. Es muß aktiv agieren und zwar

=> auf syntaktischer und semantischer Ebene als Übersetzer z. B. als zentraler NAMENSSERVER,

=> auf pragmatischer Ebene als Optimierer bei der (statischen oder dynamisch angepaßten) Fragmentierung, lokalen Verteilung und Replikation durch Erfassung und statistisch-mathematische Auswertung des Benutzerverhaltens,
=> als Instanz für Datensicherheit und Datenschutz.

In einem verteilten System stellt das DATA DICTIONARY die übergreifende Datensicherheitsinstanz dar, die Zugriffsrechte auf bestimmte Datenbestände verwaltet.

Das Data Dictionary stellt eine verteilte Datenbank mit einer Vielzahl von Koordinationsaufgaben dar, die mit voller Information entweder als zentraler Katalog (Meta Data Dictionary) auf dem Server oder einem speziellem Dictionary-Rechner gehalten wird oder als Mehrfachkatalog überall dezentral als Kopie gehalten wird. Möglich ist auch, die benötigten Informationen lokal abzustufen und als lokalen Katalog (local data dictionary) zu halten.

Natürlich sind Kombinationen aus einem zentralen Katalog und einem lokalen Katalog denkbar.

	Meta Data Dictionary		**Local Data Dictionary**
	zentral	dezentral	
Laufzeit	hoch, da immer Zugriff auf den Server erforderlich	gut	beim Zugriff auf nicht lokale Objekte hoch
Änderungsdienst	gut	aufwendig, da Daten in jeder lokalen Kopie	gut bei lokalen Objekten
Speicherausnutzung	gut	hohe Redundanz	gut

Abb. 1.3/2: Darstellung der Kombinationen aus einem zentralen und lokalen Katalog

In der Praxis werden Kombinationen aus zentralen „Meta Data Dictionaries" und dezentralen „Local Data Dictionaries" verwendet. Speziell die Datensicherheitsprozeduren sind auch dezentral zu halten.

Bei der Gestaltung der Datendokumentationskomponente ist weiterhin festzulegen, ob bei jeder Änderung der Datenobjekte/-beziehungen/-attribute jeweils alle dezentralen Datenlexika sofort aktualisiert werden oder ob das „Update" nur in bestimmten Frequenzen erfolgen soll.

Das Problem der Datendokumentation ist eng verbunden mit der lokalen Autonomie der Endnutzer. Mit der zunehmenden Verbreitung preiswerter Arbeitsplatzrechner haben sich die Endnutzer daran gewöhnt, bei der Strukturierung und Verwaltung ihrer Datenbestände sehr eigenständig zu agieren. Eine logische Integration bei Autonomie der Endnutzer läßt sich nur über eine umfassende und komfortable Dokumentation in einem Data Dictionary lösen.

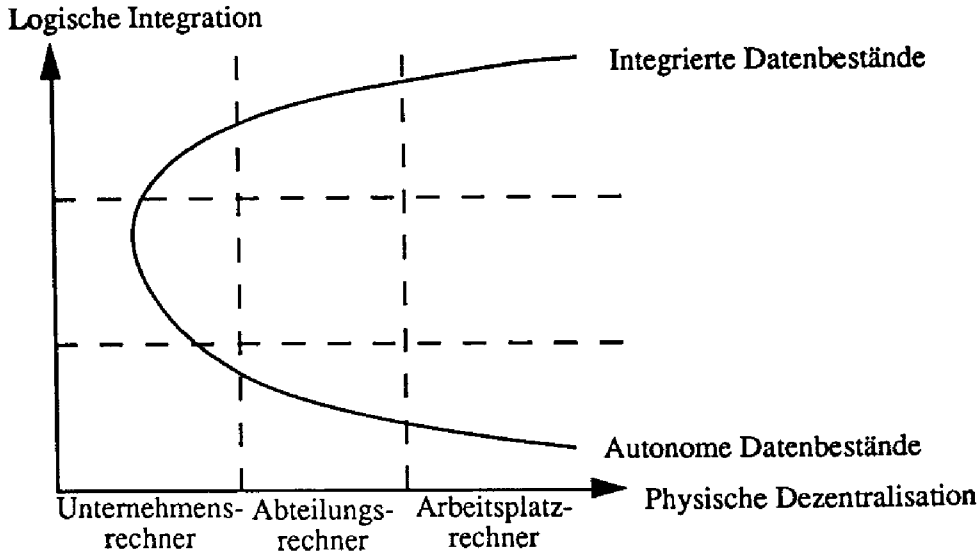


Abb. 1.3/3: Abhängigkeit der Datenbestände zwischen logischer Integration und physischer Dezentralisation

Mit der Verbreitung von Arbeitsplatzrechnern ist deren Datenbestand zunehmend Bestandteil des logisch integrierten Datenbanksystems der Unternehmungen und muß sich den dafür geltenden Dokumentationsprozeduren unterwerfen.

Datenzuordnung

Die Probleme der lokalen Verteilung von Datenbeständen können entweder aus DV-technischer und/oder aus betriebswirtschaftlich-organisatorischer Sicht optimiert werden.

Sicht	Gesichtspunkte	Nebenbedingung
DV-technisch	-> CPU-Verarbeitungszeiten auf Server- und Client-Rechnern -> Zeiten lokaler Ein-/Ausgabeoperation -> Datenübertragungszeiten	-> Höchstantwortzeiten -> Mindestverfügbarkeit
betriebswirtschaftlich-organisatorisch	-> Kommunikationskosten -> DV-Investitionskosten -> DV-Betriebskosten, insbesondere für -> Datenmanagement ---------- -> Nutzen einer Dezentralisierung von Entscheidungsbefugnissen	-> Betriebssicherheit

Abb. 1.3/4: Kriterien der lokalen Verteilung von Datenbeständen

Aus betriebswirtschaftlich-organisatorischer Sicht sind zum einen die direkten Kosten für DV-Investitionen und den DV-Betrieb einer verteilten Datenbank von Bedeutung.

Von unter Umständen größerer betriebswirtschaftlicher Relevanz sind die indirekten Wirkungen auf die Aufgabenerfüllung in einem Betrieb.

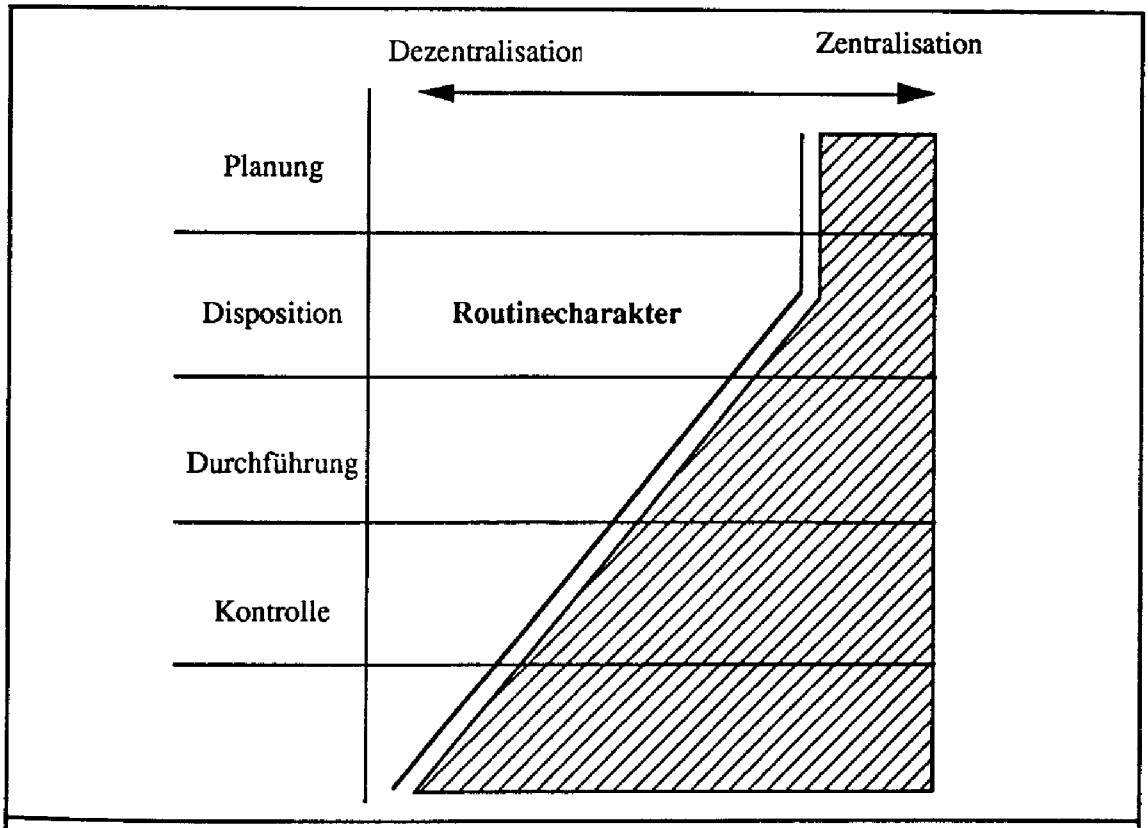


Abb. 1.3/5: Zentralisation und Dezentralisation

Die übergreifende logisch zentralisierte Verfügbarkeit physisch verteilter Datenbestände eröffnet neue Möglichkeiten einer Dezentralisation/Zentralisation in Abhängigkeit vom Aufgabeninhalt.

Aus strategischer Sicht kann die Einführung einer verteilten Datenbank auf der Basis eines Kommunikationsnetzes der erste Schritt zur Abflachung von Managementhierarchien sein. Aus operativer Sicht besteht die Möglichkeit, die jeweilige Aufgabe dort anzusiedeln, wo ein optimaler Aufgabenerfüllungsgrad zu erwarten ist. Üblicherweise sind zwei gegenläufige Tendenzen anzutreffen:

Bei der Steuerung des Materialflusses von der Beschaffung bis hin zum Absatz werden mit der Einführung vernetzter und verteilter Systeme Befugnisse dezentralisiert, dabei allerdings gleichzeitig mit verbesserten Controlling-Prozeduren in ein System vermaschter Regelkreise integriert.

Bei der Steuerung des Informations- und Finanzflusses wird der Grad der Zentralisation hingegen erhöht (vgl. Gunson/Boddy (1989)). Routineaufgaben werden somit verstärkt dezentralisiert und dabei werden verstärkt methodisch einheitliche, integrierte Dispositions- und Informationssysteme verwendet.

Die lokale Datenverteilung kann durch manuelle Eingriffe vor oder während des Betriebs der Datenbank erfolgen oder sie kann automatisch nach DV-technischen Kriterien während

des Betriebs einer verteilten Datenbank ausgeführt werden. Dazu benötigt das System ausgefeilte Prozeduren zur Messung des Benutzer- und Gesamtsystemverhaltens (Antwortzeiten, Kommunikationsverkehr).

Datenintegrität

Die Datenintegrität bei Abkehr vom regional zentralistischen Datenbanksystem ist praktisch äußerst schwierig zu gewährleisten. Es können logische und technische Fehler bei den Transaktionen auftreten. Diese konkurrieren miteinander um den gleichen Datenbestand.

Daher sind Überlegungen hinsichtlich der Steuerungsorganisation und des Steuerungsablaufs anzustellen.

Bei der **Steuerungsorganisation** ist zwischen gleichrangigen und hierarchisch angeordneten Systemen zu unterscheiden. Bei **gleichrangigen Systemen** sind auf jedem einzelnen Knoten die Dienste eines zentralisierten DBMS mit seinem (lokalen) Transaktionsmanager verfügbar. Bei **hierarchisch angeordneten Systemen** existiert ein zentraler Transaktionsverwalter, der die Aktionen der lokalen Transaktionsverwalter steuert. Es kann sich dabei um einen dezidierten zentralen Transaktionsverwalter handeln oder der jeweils lokale Auslöser wird zentraler Transaktionsverwalter (Wurzelknoten-Konzept).

Beim **Steuerungsablauf** existieren Konzepte, deren Ziel die jederzeitige Datenkonsistenz ist und solche, die darauf verzichten.

Konzepte mit dem Ziel der Datenkonsistenz kämpfen mit folgenden Problemen:

- Wie werden replizierte (doppelt gespeicherte) Datenbestände so von der Transaktion betroffen, daß alle Kopien - auch bei Knoten-/Kommunikationsausfall - jederzeit im gleichen Zustand sind (Motto: Alle oder keiner)?
- Wie wird verhindert, daß zwei verschiedene Transaktionen gleichzeitig zwei verschiedene Kopien des gleichen Objektes ändern?
- Wie kann eine eindeutige Sperre verteilter Datenbestände auch angesichts der technischen Kommunikationsverzögerungen erreicht werden?

Die erforderliche Synchronisierung von Transaktionen und Replikationen ist praktisch äußerst schwierig.

Im sogenannten Two-Phase-Commit-Ablauf (vgl. Reuter (1988), S. 43-44) werden Datenbankobjekte vor ihrer Benutzung gesperrt, wenn zwischen den Transaktionen Wartebeziehungen entstehen. Die Koordination übernimmt ein Knoten, der zuerst alle betroffenen Knoten anweist, sich auf eine Änderung einzustellen und die betroffenen Datenbestände sperrt, dann die Änderung durchführt und schließlich in der 2. Phase nach Rückmeldung aller Knoten über eine erfolgreiche Änderung diese abschließt.

Phase	Teilphase	Koordinator (Server)	Teilnehmer (Client)
1	Beginn	Vorbereitung "Teilnehmertransaktion" an alle Teilnehmer senden	
	Mitte		Transaktion vorbereiten und Bestätigung senden
	Ende	Bestätigung der Teilnehmer empfangen	
2	Beginn	Teiltransaktion durchführen, an alle Teilnehmer senden	
	Mitte		Teitransaktion durchführen und Bestätigung an Koordinator melden
	Ende	Bestätigung der Teilnehmer empfangen und Transaktion dann abschließen	

Abb. 1.3/6: Ablauf des Two-Phase-Commitment

Aufgabe des "Two-Phase-Commit" ist es, verteilte Transaktionen über mehrere Client-Rechner zu ermöglichen, ohne daß es im Störungsfall zu Fehlfunktionen kommt.

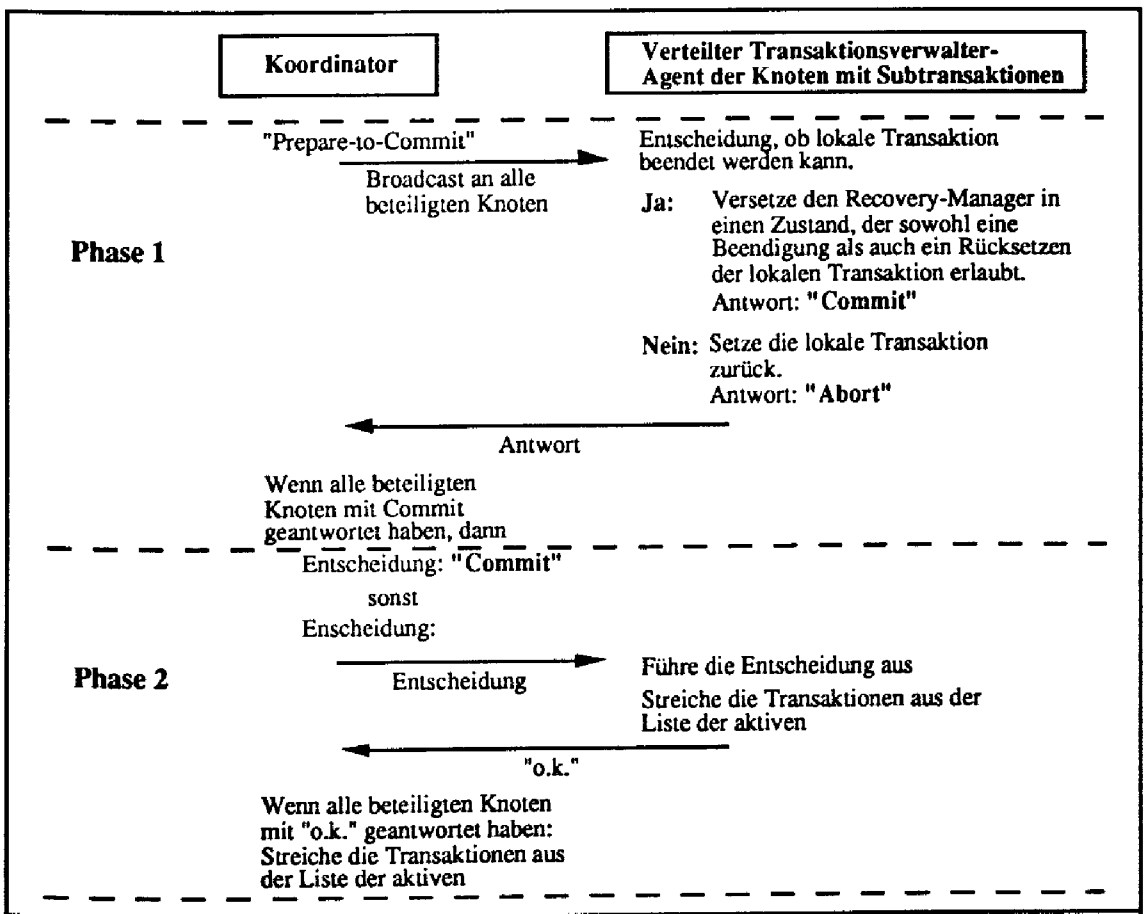


Abb. 1.3/7: Grundzüge des Two-Phase-Commitment

Es existieren unterschiedliche Typen des "Two-Phase-Commit":

- Beim transparenten 2PC ist die eigentliche Anwendung nicht in den Commit-Prozeß einbezogen, der von einem separaten Datenbank-Modul abgewickelt wird.
- Beim einfachen 2PC wird auch die Anwendung vom Commit-Prozeß betroffen.

Konzepte ohne vollständige Datenkonsistenz stellen die Frage, ob Transaktions-Abläufe überhaupt das richtige Konzept für verteilte Datenbanken mit langen und unsicheren Kommunikationswegen und Komponenten sind?

Ist es für die praktische Problemstellung wirklich notwendig, daß die Datenbank jederzeit auf jedem Knoten konsistent ist (Beispiel: Telefonbuch vs. zentrale Auskunft)? Zu beachten sind dabei die resultierenden Kosten (Betriebserfordernisse) und die erforderliche Rechnerleistung allein für die Datenbankaktualisierung.

Inwieweit werden bestimmte Ziele verteilter Datenhaltung wie

- Leistungssteigerung des Gesamtsystems,
- Vernetzung heterogener Hardware-/Software-Welten,
- Steigerung der Ausfallsicherheit,
- wirksame Entkoppelung der Teilsysteme,
- Steigerung von Datenschutz und Datensicherheit,

durch ein striktes Transaktionskonzept herkömmlicher Art wieder gefährdet?

Aus solchen Überlegungen heraus wird die Forderung nach Datenintegrität auf eine abgegrenzte Datenbasis beschränkt, sofern keine Folgeoperationen in einer anderen (verteilten) Datenbasis notwendig werden bzw. ausgeschlossen sind. Die Datenbasis wird in öffentliche und private Bereiche eingeteilt und die Datenintegrität nur im öffentlichen Bereich gesichert.

Datenunterteilung (Fragmentierung)

Dieses Problem betrifft die Bildung von disjunkten Entitätsmengen (= Fragmenten) des logisch integrierten Datenbestandes durch

-> Projektion (vertikale Fragmentierung)
-> Selektion (horizontale Fragmentierung)

Horizontale Fragmentierung

Bei der horizontalen Fragmentierung wird die Menge von Sätzen einer Relation in Untermengen aufgespalten und zwar mit Hilfe der Selektion. Diese erfolgt mit Selektionskriterien (= Attributausprägungen der jeweiligen Relation), mit deren Hilfe entschieden wird, welche Datensätze in das eine oder das andere Fragment gehören.

Bei der abgeleiteten horizontalen Fragmentierung erfolgt eine Aufspaltung der Mengen von Sätzen einer Relation nach Maßgabe von Attributausprägungen einer anderen, korrespondierenden Relation.

Vertikale Fragmentierung

Bei der vertikalen Fragmentierung wird die Menge der Attribute einer Relation in zwei oder mehrere Teilrelationen (split approach) aufgespalten bzw. es werden Attribute zu Teilrelationen (grouping approach) zusammengefaßt. Dabei werden die Teilrelationen so gebildet, daß möglichst viele Applikationen nur eine bestimmte Teilrelation benötigen. Man unterscheidet

-> vertical partioning = überschneidungsfreie Teil-Relationen
-> vertical clustering = Teil-Relationen mit Überschneidungen

(vgl. Bayer/Elhardt u.a.)

Bei Verbundoperationen bestehen dann folgende Möglichkeiten:

=> die Teilrelationen werden als Ganzes zu einem Knoten geschickt und dort wird wie im zentralen Fall der Verbund berechnet (wenig Sendungen, jedoch hohes Datenvolumen)
=> eine Teilrelation wird zu dem Vereinigungsknoten geschickt und fordert für jeden Verbundwert vom anderen Knoten die Tupel mit gleichem Verbundwert (wenig Sendungen, jedoch hohe Anzahl von Sendungen)
=> Semiverbund-Methode ermittelt einen Filter und berechnet damit einen Semiverbund (Optimale Form des Filters?)

Gemischte Fragmentierung

Entsprechend eines Zerlegungsbaums werden entweder horizontale Fragmente vertikal weiter zerlegt bzw. vertikale Fragmente horizontal weiter zerlegt.

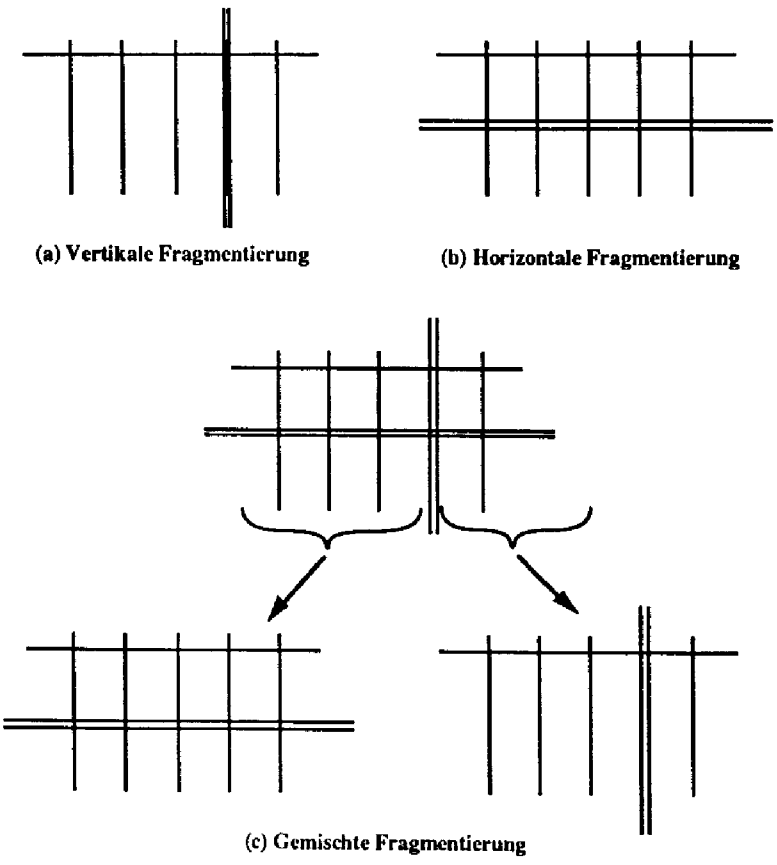


Abb. 1.3/8: Formen der Fragmentierung

Ein Merkmal voll verteilter Datenbanken ist die "fragmentation transparency", d. h. der Benutzer arbeitet mit globalen Relationen und ihm wird nicht bewußt, daß er mit logisch aufgeteilten Datenbeständen hantiert.

Datenwiederholung (Replikation)

Die Wiederholung (Replikation) von Daten in verschiedenen Teilrelationen kann aus diversen Gründen gerechtfertigt sein, so etwa zur Steigerung der Datenkonsistenz und der lokalen Autonomie. Welcher Grad der Wiederholung der Daten in anderen Knoten günstig ist, kann nicht allgemein beantwortet werden: Doppelspeicherung ist für Leseoperationen günstig, für Update-Operationen hingegen ungünstig. Redundante Datenhaltung erspart Datenkommunikationszeiten/-kosten bei Leseoperationen, erschwert jedoch die konsistente Änderung aller Kopien bei Schreiboperationen. Generell gilt: Erhöhe die Verfügbarkeit und Zuverlässigkeit des Systems durch Kopien der Datenbestände (availability and reliability).

Es gibt verschiedene Lösungswege, um die Datenkonsistenz auch in Datenkopien sicherzustellen:

(a) alle Kopien erhalten eine Zeitmarke und werden permanent aktualisiert (vgl. zeitbezogene Datenbanken)
(b) Kopien werden zwar für den lokalen Gebrauch gezogen, dann jedoch nicht wieder aktualisiert.

Die folgende Abbildung zeigt, welche Kombinationen aus Aktualisierungszeitpunkt und Aktualisierungsumfang grundsätzlich möglich sind.

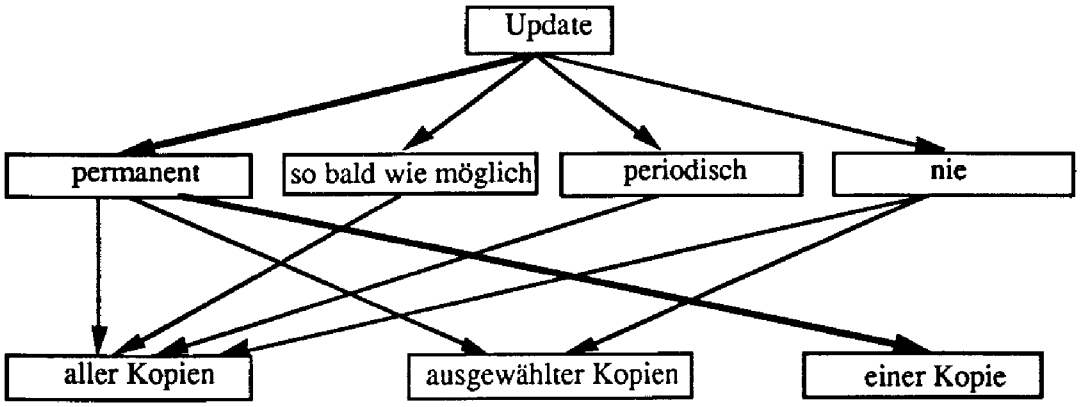


Abb. 1.3/9: Aktualisierungszeitpunkt bei Replikationen in einer verteilten Datenbank

Grundsätzlich gilt, daß mindestens ein Knoten den umfassenden aktuellen Stand der Datenbank halten muß. Dieser "Meisterknoten" kann fixiert sein oder variabel bestimmt werden.

Ein Merkmal verteilter Datenbanken ist "replication transparency", d. h. dem Benutzer wird nicht bewußt, daß er mit doppelt gespeicherten Datenbeständen arbeitet.

Generell gilt für eine voll verteilte Datenbank, daß nach außen (in externer Sicht) der Datenbestand ohne Replikation und konsistent als logische und räumliche Einheit gezeigt wird (fragmentation & allocation & replication transparency).

Technik

Die technischen Probleme einer verteilten Datenbank liegen in der Verbindung heterogener Hardware- und Software-Welten und in der Sicherung eines reibungslosen Betriebs bei technischen Störungen.

	Daten	**Kommunikation**	**Rechner**
Datenbank-implementierung	> Verteilung der Datenbestände nach Aspekten der Last > Heterogene Datenbank-Systeme	> Heterogene/homogene Netzstrukturen und Kommunikationsprotokolle	> Heterogene Betriebssysteme > Rechnerhierarchien
Datenbank-betrieb	> Sende (einer an einen)-Prozeduren > Broadcasting-Prozeduren (einer an viele)	> Logische Ebene der Datenkommunikation (Satz, Block, Seiten, Objekt)	> Prozeduren bei Zuschalten/Abschalten von DB-Knotenrechnern
Datensicherheit	> Prozeduren bei Ausfall eines Teil-Datenbestandes	> Prozeduren bei Teilnetzausfällen	> Prozeduren bei Ausfall von DB-Knotenrechnern

Abb. 1.3/10: Technische Probleme beim Betrieb einer verteilten Datenbank

2. Schritte des Entwurfsprozesses

Der Entwurf einer verteilten Datenbank unterscheidet sich nur im dritten Schritt. Die Schritte 1. und 2.: "Konstruktion und Modellierung des Globalen Schemas" entsprechen dem bei zentralisierten Datenbanken (vgl. Ceri/Pelagatti (1985), S. 38 ff).

Im dritten Schritt wird dann ein Verteiltes Datenschema entwickelt.

Schritt 3.1.: Bildung des fragmentierten Datenschemas

In diesem Schritt erfolgt die logische Aufsplittung der globalen Relationen aus dem Schritt 3 in sich nicht überschneidende (logisch disjunkte) Fragmente mit "gleichen Eigenschaften aus der Sicht der Anwendung", z. B. in Anlehnung an aufbau- oder ablauforganisatorische Gemeinsamkeiten oder an den Datenbedarf der Haupt-Anwendungsprogramme (vgl. Ceri/ Pelagatti, S.41ff und S.72ff).

Dabei gelten folgende Anforderungen:

* Vollständigkeit der Fragmentierung, d. h. die globale Relation wird vollständig in disjunkte Teilmengen zerlegt
* Rekonstruierbarkeit der globalen Relation durch Vereinigung (bei horizontaler Fragmentierung) oder natürlichem Verbund (vertikale Fragmentierung)
* Überschneidungsfreiheit (disjointness condition) der Fragmente (zumindestens bei der horizontalen Fragmentierung)

Schritt 3.2.: Entwicklung des verteilten Datenschema

Aufgabe ist hierbei die lokale Verteilung (zunächst nur in logischer Sichtweise) der Daten-Fragmente und der Daten-Transaktionen auf die verschiedenen KNOTEN unter Beachtung

* der Speicherkapazität der einzelnen Knoten
* maximal zulässiger Antwortzeiten,

Zu treffen sind dabei Entscheidung über Kopien und somit Redundanz sowie die Trennung von Daten und Transaktionen.

Die Zielfunktion enthält als Elemente die Kosten für Speicherplatz, Datenkommunikation, Transaktionen und soweit quantifizierbar den Nutzen aus Datenverfügbarkeit (Schnelligkeit) und Systemsicherheit.

Die Gesamtkosten = Zugriffskosten (lokal + entfernt) + Kommunikationskosten sind nicht nur abhängig von der Datenverteilung sondern auch vom verwendeten Netz.

$$T_{Server} = T_{Server}^{Wartezeit} + T_{Server}^{Bearbeitungszeit}$$

$$T_{Auftrag}^{Übertragung} = T_{Client} + T_{Protokoll}^{Senden} + T_{Kanal} + T_{Protokoll}^{Empfangen}$$

$$T_{Auftrag}^{Abwicklung} = T_{Server} + 2 * T_{Auftrag}^{Übertragung}$$

Abb. 2/1: Zeitbetrachtung T

Die Zeitbetrachtung gibt Anhaltspunkte für die Optimierung des Systemdurchsatzes. Dieser hängt zum einen ab von der Nutzdatenmenge des Auftrags, zum andern von den Zeitbestandteilen (zu möglichen Zuweisungsalgorithmen vgl. Wiederhold (1989), S.494f)).

Im vierten Schritt der Implementierung in ein verteiltes Datenbanksystem sind die existierenden Datenbanksysteme zu integrieren.

3. Architekturen verteilter Datenbanksysteme

3.1. File-Sharing-Architektur (föderative Systeme)

Dezidierte Server halten bestimmte Datenbestände. Dezentrale Workstations greifen mit expliziten "File-Transaktionsbefehlen" auf diese zu und benutzen sie wie lokale Dateien.

Bei einer Transaktion in einer „File-Sharing-Architektur" wird zwischen dem Server und der jeweiligen Arbeitsstation jeweils eine vorher bestimmte Datenmenge übergeben. Im Grenzfall kann dies eine gesamte Teil-Datenbank sein. Die Selektion im Server beschränkt sich also auf die Auswahl des jeweils angesprochenen Teil-Datenbestandes.

Dezentrale Workstations halten keine komplette Datenbank-Software, sondern neben der Applikationssoftware nur noch Elemente für die Anfrageverarbeitung (die die Übersetzung der Anfrage an die eigentliche Datenbank übernehmen). Die Kommunikation mit dem Server erfolgt auf der Basis von Komplexobjekten.

Im eigentlichen Sinne handelt es sich noch nicht um eine verteilte Datenbank, da das Merkmal der lokalen Transparenz der Datenbestände nicht erfüllt ist.

Organisationsformen:

* File locking (Dateisperre): Nur ein Nutzer hat zur Zeit Zugriff auf die jeweiligen Teile der Datenbank, diese wird ihm über das Netz auf seine Workstation geladen (hoher Datentransport)
  -> der Eigentümer der Datei ist für deren Aktualisierung und Konsistenz verantwortlich
  -> jeweiliger Besitzer erhält nur einen "momentanen Schnappschuß" der Datei.
* Record locking (Satzsperre): Mehrere Nutzer können einen Satz lesen, jedoch nur einer kann ihn ändern.

Bei einer **One-Server-File-Sharing-Architektur** hält ein dezidierter Netz-Server bestimmte zentrale Datenbestände, die von den lokalen Arbeitsstationen nach entsprechenden "log in-Kommandos" wie lokale Dateien angesprochen werden können.

Bei einer **Multi-Server-File-Sharing-Architektur** können die lokalen Arbeitsstationen über ein Netz die Dateien anderer Arbeitsstationen ansprechen und dort (durch Berechtigungen) bestimmte Transaktionen vornehmen.

3.2. Client-Server-Architektur

Von Client-Server-Architekturen wird allgemein dann gesprochen, wenn die Prozesse auf den dezentralen Arbeitsstationen unabhängig, doch koordiniert von denen auf den zentralen Datenservern ablaufen können.

Die Backend-Rechner fahren Server-Prozesse, die das Datenbankmanagement (Speicherung, Zugriff, Transaktions-, Recovery-Integritätslogik) übernehmen. Diese Backend-Rechner halten den leistungsfähigen Datenbank-Kern und senden jeweils ausgewählte Sätze an die dezentralen Arbeitsstationen, die die Präsentation und Bearbeitung der Daten (Grafik, Windowing) erledigen und dabei heterogene Endnutzersysteme verwenden können. Ein Unterschied zur „File-Sharing-Architektur" liegt also darin, daß die Datenbanksoftware mittels Server-Prozeß „intelligent" den angesprochenen Teil der Datenbank selektiert und dadurch die zu übertragende Datenmenge beschränkt.

Die dezentralen Workstations fahren Client-Prozesse zur Kommunikation mit der Datenbank und können auf dieser Basis eine unabhängige Anwendungssoftware nutzen. Sie halten neben der Applikationssoftware und einer Anfrageverarbeitungskomponente weitere Elemente des Datenbanksystems, z. B. Komplex- und Basisobjektverwaltung, so daß die Kommunikation zwischen Server-Prozeß und Clients-Prozeß auf der Basis von Seiten (pages) erfolgt. Eine mögliche Organisationsform ist das selektive Record Locking mit selektivem Datentransport.

		File Sharing		Client Server	
		One Server	Multi- Server	One Server	Multi- Server
Nutzer	Regionale Verteilung	dezentral	dezentral	dezentral	dezentral
	externes Schema	heterogen	heterogen	heterogen	heterogen
Datenbestand	Regionale Verteilung	zentral	dezentral	zentral	dezentral
	logisches Schema	heterogen	heterogen	homogen	homogen
Selektion	Durch-führender	ein Rechner	mehrere Rechner	ein Rechner	mehrere Rechner
	Prozeduren	Selektionen pro Datei		Selektionen pro Satz	

Abb. 3.2/1: Abgrenzung File Sharing vs. Client-Server-Architektur

Bei der Betrachtung des Client-Server-Modells muß man zwischen der Hardware-Sicht und der Software-Sicht, die die logische Funktionsweise widerspiegelt, unterscheiden: Dezentrale Anwendungsprogramme auf den jeweiligen Arbeitsplatzrechnern kommunizieren mit den Serverprozessen der Datenbank.

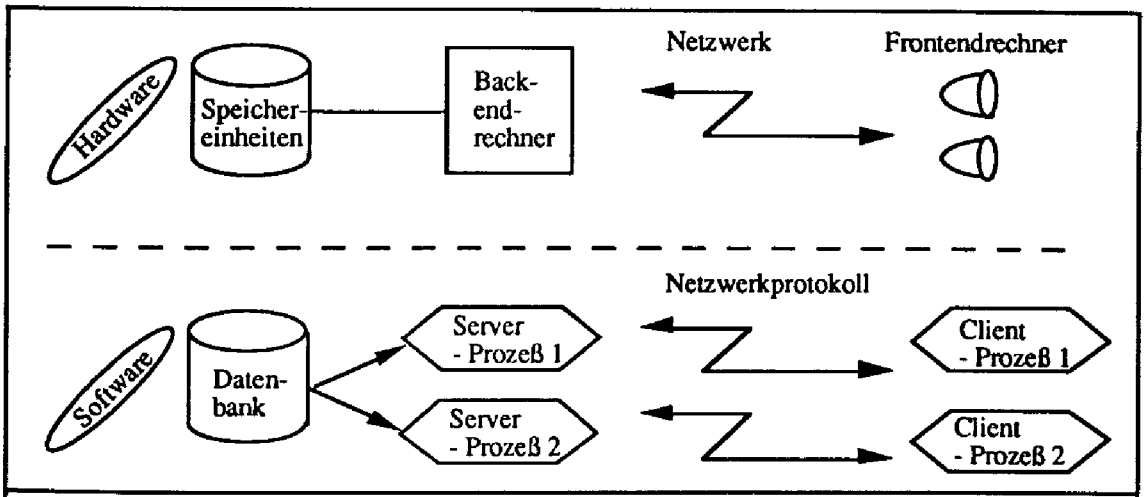


Abb. 3.2/2: Hardware- und Software Sicht einer Client-Server-Struktur

Der wesentliche Bestandteil ist also ein Netzwerk-Datenbank-Server, d. h. eine Software, die die Brücke zwischen dem Backend-Rechner mit der Datenhaltung und den Workstations mit den Client-Prozessen schlägt. Diese Software managt die Selektions-/Kommunikationsprozesse der Frontend-Applikation an die Backend-Datenbank. Sie wird heute oft als SQL-Server bezeichnet und wird von mehreren Herstellern gemeinsam mit einer relationalen Datenbank angeboten, kann aber auch auf Fremddatenbanksysteme zugreifen.

Produkt	Anbieter	Betriebssysteme	Fremddatenbanken
SQL Base Server	Gupta	DOS+NETWARE OS 2 UNIX	DB2 (IBM) ORACLE (Allbase) INFORMIX INGRES SQL-Server
SQL-Server	Microsoft	DOS+NETWARE OS2	
ORACLE-Server	Oracle	DOS+NETWARE OS 2 UNIX	
INGRES-Server	Ingres	UNIX VMS (DEC)	DB2 (IBM) ORACLE SYBASE

Abb. 3.2/3: Beispiele von Server-Software nach dem Client-Server-Konzept

Zwischen den Server-Prozessen und den Client-Prozessen existiert eine Schnittstelle, die die Einkapselung der vom Server angebotenen Funktionalität garantiert, d. h. sie verbirgt vollständig den Clients die Details der internen Implementierung im Server und erlaubt es

dadurch, daß unterschiedliche Anwendungsprogramme auf die Datenbank zugreifen können [Autonomie von Server und Client].

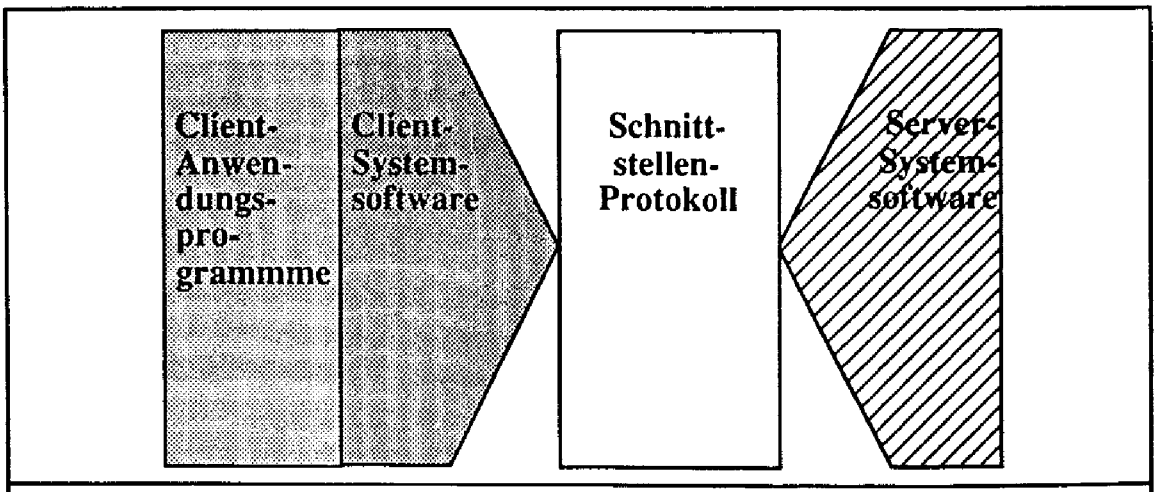


Abb. 3.2/4: Autonomie zwischen Server und Client

Das Schnittstellen-Protokoll kann z. B. entsprechend dem ISO-OSI-Schichtenmodell aufgebaut und auch datenbanksystemübergreifend gestaltet sein (z. B. RDA= Remote Data Acesses).

Architekturen von Client-Server-Konzepten

		Server	
		einer	viele
Client	einer	(1) (1 : 1)-Architektur	(2) (1 : n)-Architektur
	viele	(3) (m : 1)-Architektur	(4) (m : n)-Architektur

Abb. 3.2/5: Architekturen von Client-Server-Konzepten

Eine 1:1-Architektur bedeutet, daß ein Server-Prozeß exklusiv ein Client-System bedient. Für 100 Benutzer sind damit 100 Client-Prozesse und 100 Server-Prozesse erforderlich. Diese Architektur ist selten und praktisch nur von Bedeutung, um den Client von den Details des Servers abzuschirmen.
Eine 1:n-Architektur bedeutet, daß ein Client-System mit mehreren Server-Systemen zusammenarbeitet. Eine solche Architektur ist logisch dann von Bedeutung, wenn ein Client gleichzeitig aus mehreren Servern Daten abfragt und diese zu einem eigenständigen externen Schema komponiert.

In der Praxis verbreitet sind jedoch nur die Fälle (3) und (4). Bei der **Single-Server-Architektur** (m:1-Architektur) kann man folgende Varianten unterscheiden: Ein einziger Server-Prozeß bedient (theoretisch) unendlich viele Frontend-Prozesse (1:n Architektur), was die Anzahl der benötigten Prozesse erheblich senkt (bei 100 Benutzern sind jetzt nur noch 100 Client- und 1 Server-Prozeß notwendig). Der einzige Server-Prozeß kann jedoch schnell zum Engpaß werden, da er eine sequentielle Abarbeitung der Client-Anfragen bedingt.

Alternativ sind mehrere Server-Prozesse, z. B. einer pro Client oder einer pro Auftrag denkbar. Dazu ist im Backend-Rechner ein Steuerprozeß, der sogenannte "listener" realisiert,

der die neu eintreffenden Aufträge der Clients empfängt und deren Abarbeitung verwaltet, in dem er jeweils einen Server-Prozeß kreiert, der anschließend direkt mit dem Client über einen temporären logischen Kanal kommuniziert. Server-Prozeß und dynamischer Kanal bleiben solange bestehen, bis der Client seine Anforderungen beendet, dann werden sie gelöscht.

Die Multi-Prozeß-Serverkonzeption wird unter UNIX entweder mit vollen Prozessen oder mit sogenannten "Threads" durchgeführt. Threads sind "leichtgewichtige" Prozesse, die sich einen gemeinsamen Adreßraum teilen und es daher ermöglichen, daß die Clients auf gemeinsame Daten gleichzeitig zugreifen können.

Die Hersteller von Datenbanken sprechen schon bei einer solchen Multi-Prozeß-Serverkonzeption von einer Multi-Server-Architektur, ohne daß der Datenbestand dazu auf verschiedene Backend-Rechner verteilbar ist. Diese Konzepte erlauben allerdings eine bessere Ausnutzung von Backend-Rechnern mit Multiprozessor-Technologie, in dem für jeden Prozessor bestimmte Server-Prozesse eingerichtet werden.

Datenbank-system	**Hersteller-bezeichnung**	**Erläuterung**
ORACLE Version 6	Dedicated Multi Server-Architecture	Jedem Client-Prozeß ist ein Server-Prozeß dediziert zugeordnet
ORACLE Version 7	Multi Threaded-Multi Server Architecture	Ein Server-Prozeß bedient gleichzeitig mehrere Client-Prozesse; er ist diesem nicht dediziert zugeordnet (non-dedicated-Server). Es gibt eine Anzahl von Server-Prozessen auf einem Backend-Rechner, die unter Umständen auf mehreren Prozessoren laufen können.
SYBASE	Virtual Server Architecture	
INGRES Version 6.4.	Distributed Multi Server Architecture	Aufgrund von Ereignissen bei den Clients oder Anfragen der Client-Prozesse generiert die Datenbanksoftware Serverprozesse, die Arbeitsabläufe zur Datenselektion auf mehreren Rechnern erlauben.

Abb. 3.2/6: Herstellerbezeichnungen für 1:n-Client-Server-Konzepte

One-Server-Konzepte erlauben zwar über die dezentralen Client-Prozesse eine verteilte Datenverwaltung, verwenden aber einen zentralen Datenbank-Server.

Erst bei der **Multi-Server-Architektur (n:m-Architektur)** kann von einer verteilten Datenbank gesprochen werden. Mehrere Server-Prozesse auf regional verteilten Backend-Rechnern mit Datenbanken können mehreren Client-Prozessen frei zugeordnet werden.

Diese Server-Prozesse können in einer Hierarchie zueinander stehen, d. h. ein Server ist der "zentrale Ansprechpartner" für alle Clients (Export-Prinzip). Dieser "Zentralserver" entscheidet, ob er den Auftrag allein oder in Zusammenarbeit mit anderen Servern abarbeitet. Alternativ können die dezentralen Clients autonom über eine einheitlich gestaltete Schnittstelle mit den dezentralen Servern kommunizieren (Import- oder auch Hol-Prinzip).

In einer Multi-Server-Architektur übernimmt einer der Server die Funktion des Namensservers, d. h. die Verwaltung aller durch logische Namen gekennzeichneten vorhandenen Ressourcen und deren Abbildung auf physikalische Einheiten.

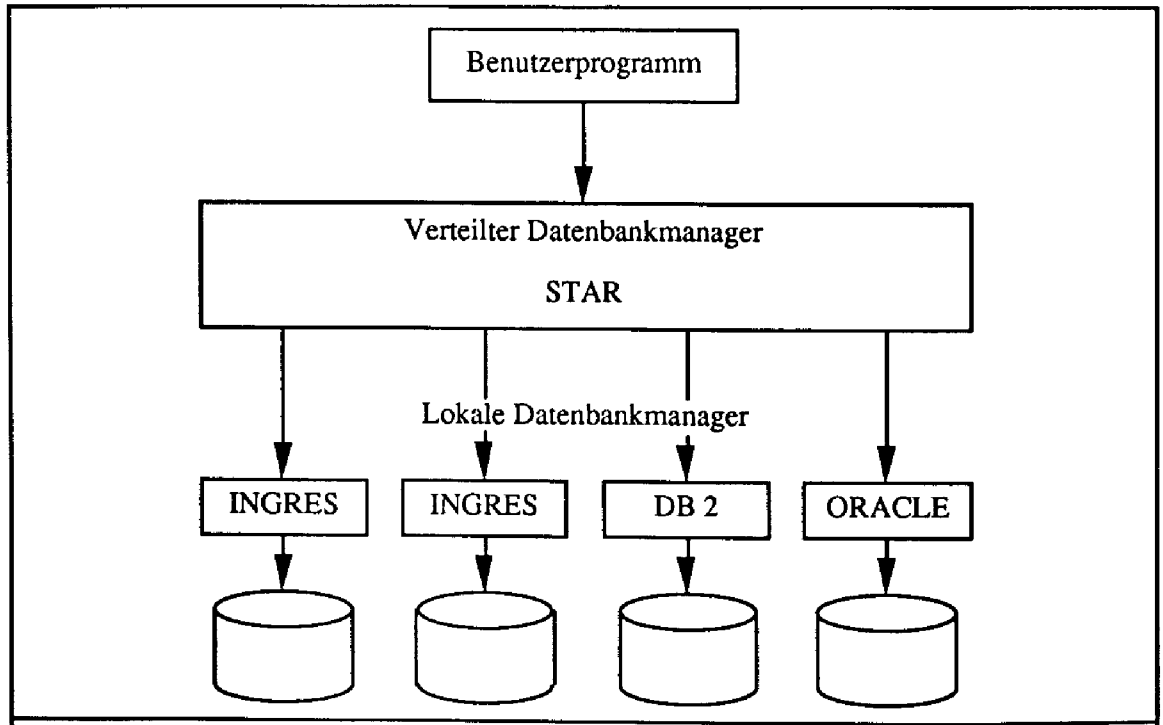


Abb. 3.2/7: Aufbau von INGRES-STAR als Beispiel für eine Multi-Server-Architektur mit einem Zentralserver

Der verteilte Datenbankmanager übernimmt folgende Aufgaben:

- er zerlegt Abfragen in Unterabfragen (Subqueries) entsprechend der Datenverteilung,
- er übergibt die Abfragen an den jeweiligen Knoten, an dem die entsprechenden Daten gespeichert sind,
- er sammelt die Antworten zu den Unterabfragen,
- er gibt die Ergebnisse der verteilten Abfrage an das Benutzerprogramm weiter.

Die Entwicklung von verteilten Datenbanksystemen befindet sich in vollem Fluß; allgemein gültige Konzepte haben sich noch nicht herausgebildet.

System / Eigenschaften	Fragmentierung	Replikation	Kriterien der Optimierung	Anfrageübersetzung	Globale Anfrageoptimierung	Synchronisierung	Deadlockbehandlung	Datenmodell
DDM	Horizontal	ja	Kommunikation CPU	ja	zentral	Sperren	zentrale Erkennung	funktional
DDTS		ja	-	ja	zentral/lokal	Sperren	Vermeidung	Entity-Relationship
Distributed Ingres	Horizontal	nein	Kommunikation CPU	nein	zentral	Sperren	-	relational
ENCOMPASS	Horizontal	nein	-	nein	zentral	Sperren	Time-out	relational
POLYPHEME	-	nein	Kommunikation	nein	zentral	-	-	relational
POREL	Horizontal	ja	Kommunikation	ja	zentral	Sperren	Vermeidung	relational
R*	-	nein	Kommunikation CPU, E/A	ja	zentral/lokal	Sperren	verteilte Erkennung	relational
SDD-1	Horizontal Vertikal	ja	Kommunikation	nein	zentral	Zeitstempel	Vermeidung	relational
SIRIUS-DELTA	Horizontal Vertikal	ja	Kommunikation	ja	zentral	Sperren	Vermeidung	relational
VDN	Horizontal mit Überlappung	ja	-	-	zentral	Explizite Sperren durch den Benutzer	-	relational

Abb. 3.2/8: Übersicht über Prototypen verteilter Datenbanksysteme, (nach Mohan (1984) und Reuter (1988))

4. Beurteilungskriterien für verteilte Datenbanken

Sicht	Kriterium	Gesichtspunkte	Mögliche Alternativen
Logische Sicht	Datenmodell-kennzeichen	1. Welche Datenobjekt-Typen werden unterstützt?	- einfache - komplexe
		2. Welche Datenstruktur-Typen sind zugrundegelegt?	- hierarchisch - Netze - Tabellen - Klassen
		3. Welche Prozeduren zur Integritäts-sicherung werden unterstützt?	- Two Phase Commitment? - Unterscheidung von öffent-lichen und privaten Bereichen
	Transparenz-grad	- Systemunabhängigkeit - Ortsunabhängigkeit - Dekompositionsunabhängigkeit	
	Datenbank-sprache	1. Mächtigkeit der DDL/DML/ DCL	- SQL-Unterstützung - eigene Datenbanksprache
		2. Verbindung mit Software-Entwick-lungstools	- CASE - 4 GL Programmiersprache
		3. Welches Konzept eines verteilten Daten- Dictionaries wird verfolgt?	
Technische Sicht	System-architektur	1. Hardwareanforderungen an - Arbeitsspeicher - Peripheriespeicher - Kommunikationseinrichtungen?	
		2. Auf welchen Betriebssystemwelten ist das DDBS lauffähig?	- herstellerübergreifend - heterogen
		3. Welche Netzstrukturen und -proto-kolle werden unterstützt?	- ISO/OSI - TCP/IP
		4. Welche Fremddatenbanken sind einbindbar?	Relationale Datenbanksysteme Andere Datenbanksysteme
		5. Welche Flexibilität besteht hinsicht-lich der Einbindung neuer Server-Datenbanken und neuer Clients?	
	Performance	1. Kommunikationsperformance	Prozeduren zur Optimierung des Kommunikationsverkehrs Prozeduren zur Optimierung der Datenverteilung
		2. Speicherperformance	Welchen Umfang von Kopien hält das System?

Abb. 4/1: Gesichtspunkte zur Beurteilung eines verteilten Datenbanksystems (DDMBS)

Kapitel 5: Datenbank-Rechner

Viele Datenbanksysteme sind heute physisch und logisch so komplex, daß die konventionellen Hardware-Architekturen immer häufiger ihre Speicher- und Leistungsgrenze erreichen. Der Ausweg liegt im Einsatz von Spezialrechnern für DBMS-Systeme (vgl. Wiederhold (1989), S.460ff, Su (1988)).

Diese Spezialrechner verlagern bestimmte Aufgaben von der DB-Software auf die Hardware. Dabei wird versucht, auch die Intelligenz der peripheren Speicher durch spezielle Prozessoren zu erhöhen.

Architekturelement	Konventionelle Rechner	Datenbank - Rechner
Prozessor	* Prozessor-Ausrichtung auf numerische Operationen	* Prozessor-Ausrichtung auf nicht-numerische Operationen * Parallele Multiprozessor-Architektur mit quantitativer oder qualitativer Arbeitsteilung * Spezialchips für Datenbankoperationen, auch nicht numerischer Art
Periphere Speicher	* Kein direkter inhaltlicher Zugriff auf periphere Speicher möglich, immer Transfer zum Arbeitsspeicher notwendig	* Direkter inhaltlicher Zugriff auf periphere Speicher * Verlagerung der Zugriffsoperationen auf spezielle Zugriffsprozessoren
Controller	* Geschwindigkeit peripherer Speicher bestimmt Input/Output-Geschwindigkeit * Universalcontroller ohne inhaltliche Kompetenz	* Verstärkung der Zugriffsintelligenz des Controllers * Parallele Input/Output-Zugriffsoperationen * Ausrichtung des Controllers auf Datenbank-Operationen
Arbeitsspeicher	* Zugriff nur über Adressen	* Zugriff über Dateninhalte (assoziative Hauptspeicher)

Abb. 1: Unterschiede konventioneller und Datenbank - Rechner

Datenbankrechner bilden den Hintergrund für Host-Rechner. Sie nutzen Möglichkeiten zur Leistungsverbesserung bei Datenbankbetrieb, indem Sie beispielsweise

-> schnellere Plattenzugriffe durch Parallelzugriffe ermöglichen,
-> schnellere Datenübertragung durch frühe Selektion und Attributauswahl der relevanten Daten eröffnen,
-> schnellere Operationen durch Hardware-Realisierung von Algorithmen, Dateizugriffsoperationen und die Optimierung der Speicherbelegung anbieten.

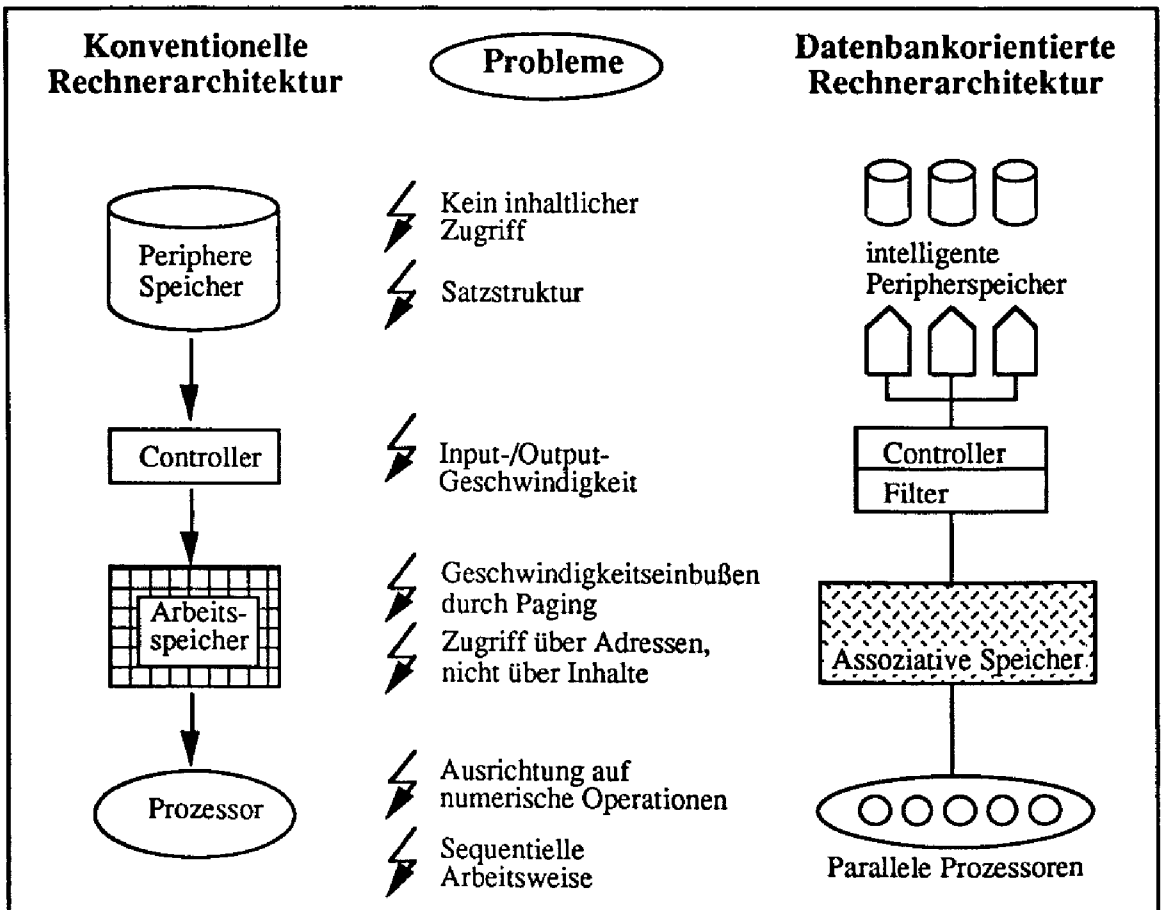


Abb. 2: Unterschiede konventioneller und Datenbank-orientierter Rechnerarchitektur (vgl. Su (1988))

Die verfügbaren Datenbank-Rechner verfolgen bei der Hardware-Architektur jeweils unterschiedliche Wege, so daß die datenbankorientierte Rechnerarchitektur bisher nicht in Reinkultur verwirklicht worden ist.

Datenbankmaschinen mit Standardprozessoren erreichen ihre Geschwindigkeitsvorteile auf unterschiedliche Weise. Eine Reduzierung des Befehlsvorrats des Prozessors kann die Datenbankmaschine beschleunigen, ebenfalls die Optimierung des Betriebssystems (z. B. beim paging) auf die Datenbank-Erfordernisse oder auf typische Datenbankfunktionen wie Abfragen, Datensuche und Integritätsbedingungen. Möglich ist auch die Erweiterung der Hardware-Architektur um intelligente periphere Speicher.

Datenbankmaschinen mit Spezialprozessoren verfügen über eine Prozessorstruktur, die z.B. für nicht-numerische Operationen optimiert worden ist und häufig über eine leistungsfähige parallele Prozessor-Architektur mit zentralen und dezentralen Einheiten. Fortschrittliche Konzepte (z.B. an jeder Lesestation ein control processor oder spezielle Kommunikationsprozessoren) ermöglichen die assoziative Suche, d.h. die Lokalisierung von Datensätzen, indem ein oder mehrere Schlüsselfelder der gespeicherten Datensätze mit einem Suchargument in Übereinstimmung gebracht werden (keine wie auch immer geartete Adressierung), die Suchanforderung gibt nicht das WO sondern das WAS an!

	Prozessor Architektur	**Speicher-architektur**	**Programmie-rung**	**unterstützte Datenmodelle**
TERADATA DBC1012	- Multiprozessorarchitektur, wobei jeder Prozessor gleiche Datenbankfunktionen ausführen kann - Parallele Prozessoren (6 - 1000) - fehlertolerante Auslegung	- Winchesterplatten	SQL-orientierte Programmiersprache	relationales Datenmodell
BRITTON LEE IDM	- Multiprozessorarchitektur mit Verteilung der Datenbankfunktionen auf unterschiedliche Prozessoren - ein Datenbankprozessor, ein Datenbank-Accelerator, bis zu 8 I/O Prozessoren, bis zu 8 I/0-Prozessoren zur Kommunikationsanbindung	- SMD-Platten	Quel-orientierte Programmiersprache	relationales Datenmodell
iDBP von Intel	- Multiprozessorarchitektur mit Verteilung der Datenbankfunktionen auf unterschiedliche Prozessoren - ein Prozessor für das DBMS und das Betriebssystem, ein Prozessor für die Kommunikationsanbindung	- Winchesterplatten - SMD-kompatible Platten		hierarchisches, relationales oder Netzwerkmodell

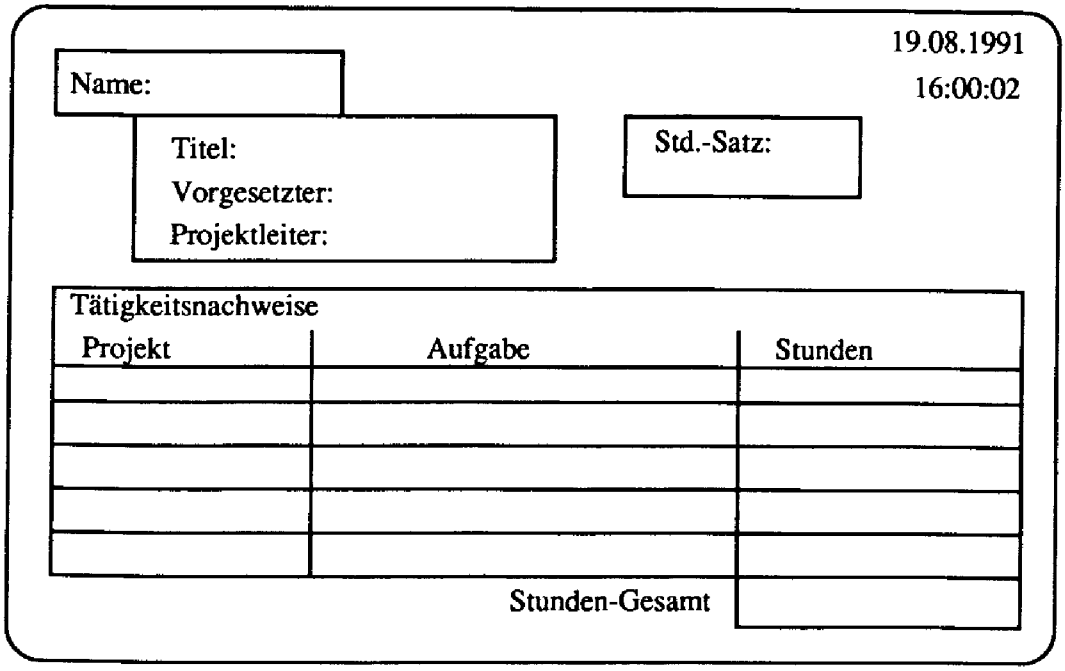
Abb. 3: Beispiele für Datenbankrechner

Übungen

Übung 1: ERM-Datenmodellierung

Ein Automobilhersteller möchte zur Projektsteuerung die beteiligten Mitarbeiter, die Teilaufgaben, die benötigten Arbeitszeiten und die Erfahrungen in einem Datenbanksystem festhalten.

Der erste noch unstrukturierte Entwurf zur Arbeitszeiterfassung geht von folgender Bildschirmmaske aus:



a) Die Bildschirmmaske gehört zum externen Datenbankschema. Beschreiben Sie das 3-Schematamodell und erläutern Sie stichpunktartig in einem Phasenschema die Vorgehensweise beim Datenbankentwurf. Beschreiben Sie in wenigen Worten die wichtigen Merkmale der einzelnen Phasen.

b) Entwerfen Sie aufgrund der Informationen in der Bildschirmmaske ein Entity-Relationship-Modell mit Entities, Relationships, Attributen und Beziehungstypen. Berücksichtigen Sie, daß nur die Informationen aufgenommen werden, die in der Bildschirmmaske enthalten sind.

c) Erweitern Sie das Entity-Relationship-Modell um folgende Punkte:
   c1) Jeder Mitarbeiter ist mit seinen persönlichen Daten zu beschreiben und er ist einer Abteilung zugeordnet.
   c2) Jedes Projekt führt zu einem neuen Produkt, das wiederum aus mehreren Einzelteilen oder Baugruppen bestehen kann. Diese Teilprodukte können in verschiedenen Abteilungen gefertigt werden.
   c3) Das Datenbanksystem soll in der Lage sein, Vergleiche zwischen selbsterstellten und fremderstellten Produkten zu erstellen. Ein Produkt kann von verschiedenen

Lieferanten geliefert werden, die abhängig von der bestellten Menge unterschiedliche Konditionen gewähren.

c4) Bei der Durchführung eines Projektes entstehen häufig gleichartige Probleme. Daher soll jeder Mitarbeiter wöchentlich seine Objekterfahrungen mit einer detaillierten Problembeschreibung für zukünftige Projekte in der Datenbank abspeichern.

d) Überführen Sie Ihr Entity-Relationship-Modell in das relationale Datenmodell und sichern Sie die 1. - 3. Normalformen.

Übung 2: Dateiorganisation und Datenbanken

Ein Futtermittel-Hersteller mischt ca. 150 Futtermittel aus ca. 30 Bestandteilen in einer chargenweisen Produktion (1 Charge pro Tag), die dann an die 20 großen Raiffeisengenossenschaften in Deutschland verkauft werden. Üblicherweise wird einmal im Quartal ein Rahmenauftrag mit jedem Kunden ausgehandelt. Der Versand erfolgt i.d.R. einmal pro Woche. Die Bestandteile werden von ca. 40 Lieferanten bezogen. Die Abwicklung erfolgt analog zum Versand.

Organisatorisch bestehen im Unternehmen fünf Bereiche. Jeder hat sein eigenes Dateisystem aufgebaut, dessen Struktur im folgenden aufgeführt ist:

Versand:	Versandnummer, Versanddatum, Versandmenge, Versandadresse, Produktnummer, Produktname
Verkauf:	Auftragsnummer, Auftragsmenge, Kundenname, Kundennummer, Kundenadresse, Produktnummer, Produktname
Qualitätskontrolle:	Analysenummer, Analysedatum, Analysestoff, Analyseergebnis
Produktion:	Produktnummer, Produktname, Bestandteilname, Bestandteilnummer, Kundennummer, Kundenname, Versandadresse, Lieferantennummer
Einkauf:	Lieferantennummer, Lieferantenname, Bestellnummer, Bestellzeitpunkt, Bestellmenge, Bestandteilname

Aufgabe 1:

Angesichts von Futtermittelskandalen möchte das Management von einem neuen System folgenden neuen Informationsbedarf erfüllt sehen:

(a1) Welches Analyseergebnis hatte die Produktionscharge, die einen bestimmten Kundenauftrag bedient hat?
(a2) Welche Stoffe wurden in welchem Monat beschafft ?
(a3) Welchen Absatz haben wir pro Monat ?
(a4) Welche Lieferanten liefern mangelhafte Bestandteile ?
(a5) Noch nicht erfüllte (offene) Aufträge
(a6) Offene Bestellungen
(a7) Welche Futtermittel wurden an welchen Kunden in welchem Zeitraum geliefert?
(a8) Welche Lieferungen an Kunden XY enthalten den Bestandteil "ABC"?
(a9) Welches Analyseergebnis hatten die Bestandteile der Lieferung an Kunde XYZ?

a) Bitte prüfen Sie, welche dieser Anforderungen sich aufgrund des Dateisystems erfüllen läßt und welche nicht? Bitte ergänzen Sie die Datenelemente so, daß sie den geäußerten Informationsbedarf erfüllen können.
b) Bitte stellen Sie die ergänzte und modifizierte Datenstruktur im Entity - Relationship - Modell dar. Kennzeichnen Sie dabei den jeweiligen Typ der Beziehung.

c) Sie wollen die Dateistruktur mit Hilfe des relationalen Datenschemas abbilden. Bilden Sie bitte die erste bis dritte Normalform auf der Grundlage der unter (a) ergänzten Dateistruktur .

Übung 3:

Aufgabe 1 Neuere Formen von Datenbanken

Zur Zeit wird in der Literatur sehr intensiv über neue Formen von Datenbanken diskutiert:

a) Was versteht man unter einer zeitorientierten Datenbank, welche Arten gibt es dabei und worin unterscheiden sich diese?

b) Was versteht man unter einer objektorientierten Datenbank, welche Arten gibt es und worin unterscheiden sich diese ?

c) Was versteht man unter einer verteilten Datenbank, welche Arten gibt es und worin unterscheiden sich diese ?

Aufgabe 2 Normalisierung von Relationen

Gegeben seien folgende Relationen:

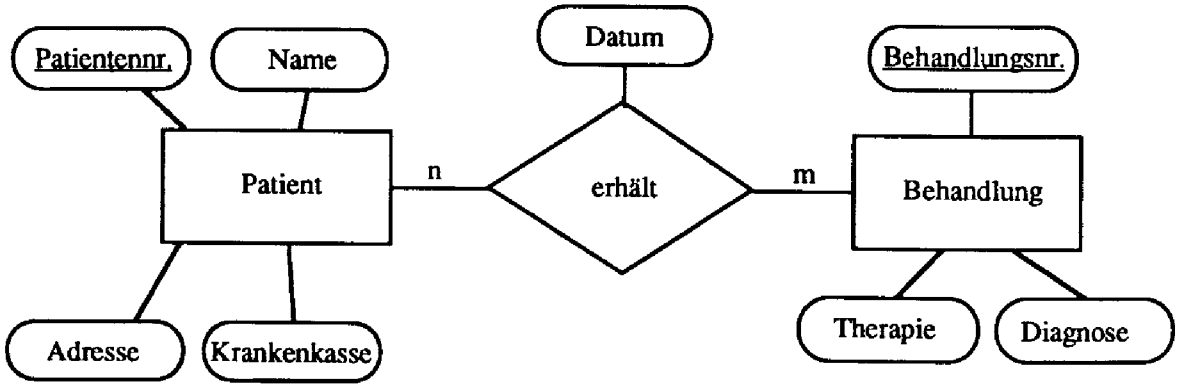
PERSONAL	(PNr., PName, PVorname, PAdresse, Lohn, Lohngruppe, Abt.Nr., Abt.Name, ProjektNr., ProjektName und P-ProjektZeit)
ABTEILUNG	(Abt.Nr., Abt.Name, Abt.Standort, MaschinenNr., Maschinenbezeichnung, Maschinenkapazität, Leiter-Nr., Leiter-Name)

a) Angenommen, Sie würden das Datenbankschema eines Unternehmens analysieren und die folgenden Relationen vorfinden. Welche strukturellen Änderungen sind erforderlich?

b) Geben Sie für die Relation "Personal" und die Relation "Abteilung" die 1. bis 3. Normalform an.

Übung 4: ERM-Datenmodellierung

Im Rahmen einer Diplomarbeit soll ein Entity-Relationship-Modell für die Patientenverwaltung einer Arztpraxis aufgebaut werden.
Ein erstes Grundmodell des Entity-Relationship-Modells ist erstellt worden und sieht folgendermaßen aus:



Aufgabe 1 ERM

Ergänzen Sie das Entity-Relationship-Modell um folgende Aspekte, bzw. erläutern Sie, wie das semantische Ausdrucksvermögen des Entity-Relationship-Modells ergänzt werden muß:

a) Für die Abrechnung mit den Krankenkassen haben alle Diagnosen eine eindeutige Diagnosenummer. Eine Behandlung kann mehrere Diagnosenummern umfassen. Die einzelnen Diagnosen besitzen eine Bezeichnung und eine Leistungsziffer für die Bewertung in DM.

b) Es muß zwischen Kassen- und Privatpatienten unterschieden werden können. Für Privatpatienten gibt es einen Steigerungssatz.

c) Die Krankenkassen geben vor, daß bei jeder Behandlung maximal 5 Diagnosen abgerechnet werden dürfen.

d) Aus der Praxis wird eine Gemeinschaftspraxis mit drei Ärzten, die das Datenbanksystem gemeinsam nutzen wollen. Dazu müssen einerseits die persönlichen Daten der Ärzte vorhanden sein, andererseits muß zu Abrechnungszwecken die Möglichkeit bestehen, für jeden Arzt eine eigene Abrechnung zu erstellen.

Aufgabe 2 Datenmodellierung

a) Überführen Sie das Entity-Relationship-Modell in das relationale Datenmodell. Erläutern Sie Ihre Vorgehensweise.
b) Überführen Sie das Entity-Relationship-Modell in das Netzwerkmodell. Erläutern Sie Ihre Vorgehensweise.
c) Überführen Sie das Entity-Relationship-Modell in das hierarchische Datenmodell. Erläutern Sie Ihre Vorgehensweise.

Aufgabe 3 Zeitorientierte Datenmodelle

Zeitorientierte Datenmodelle haben Elemente zur Berücksichtigung zeitlicher Aspekte. Betrachten Sie die nachfolgende Relation. Für einen Patienten sind die gespeicherten Tupel in einer zeitorientierten Relation aufgeführt.

geändert zum	Pat.-Nr.	Patienten-Name	Geb.-Datum	Diagnose-Kennziffer	Kranken-kasse	eingegeben am
1.2.91	45	Meier	14.7.55	345	AOK	2.1.91
1.9.90	45	Meier	14.7.55	279	AOK	28.8.91
1.9.88	45	Meier	14.7.55	279	DAK	15.8.91
15.2.88	45	Meier	14.7.55	534	privat	13.2.88

a) Zeitorientierte Datenmodelle unterscheiden zwischen zeitabhängigen und zeitunabhängigen Objekten. Zeitabhängige Objekte werden mit Zeitattributen "gestempelt". Welche Zeittypen (Zeitpunkte, Zeiträume), Zeitarten und zeitabhängigen Objekte sind bei der aufgeführten Relation benutzt worden? Begründen Sie Ihre Antwort.
b) Zeitorientierte Datenmodelle lassen sich anhand der Zeitdimension typisieren. Charakterisieren Sie zeitorientierte Datenmodelle! Welcher Typ ist bei der aufgeführten Relation benutzt worden? Begründen Sie Ihre Antwort.

Übung 5: ERM- und Datenmodellierung

Eine Tanzschule möchte für die Verwaltung ihrer Tanzkurse eine individuell angepaßte Datenbank entwickeln. Der Leiter der Tanzschule hat seine persönlichen Wünsche in einem Anschreiben an Sie zusammengefaßt. Der entsprechende Ausschnitt aus diesem Schreiben sieht folgendermaßen aus:

In der Datenbank sollen Teilnehmer, Kurse, Tanzlehrer und Räumlichkeiten verwaltet werden können. Bei den Teilnehmern soll insbesondere festgehalten werden, an welchen Kursen und in welcher Form - als Single oder mit einem bestimmten Partner bzw. Partnerin - er teilgenommen hat. Für die Planung der Kurse ist ein Stundenplan für die Tanzlehrer aufzustellen. Tanzlehrer bevorzugen bestimmte Tanzkurse und Räumlichkeiten. Da sie Ihre Lehrtätigkeit teilweise nebenberuflich ausüben, bevorzugen Sie auch bestimmte Wochentage und Uhrzeiten. Für Räumlichkeiten ist die Größe, die Einrichtung und der jeweilige Kurs festzuhalten.

Über die Datenbank soll die gesamte Kursabrechnung laufen können. Neben den Teilnehmergebühren sollen Rechnungen z. B. über den Verzehr von Getränken und den Verkauf von Schallplatten erstellt werden können. Um die Teilnehmer auf weiterführende Kurse aufmerksam zu machen, werden Sie etwa 4 Wochen vor Beginn solcher Kurse schriftlich darauf hingewiesen.

Für die interne Beurteilung der Tanzlehrer soll die Teilnahme der Kursteilnehmer an den einzelnen Tanzkursen festgehalten werden.

Aufgabe 1: ERM

Entwerfen Sie - aufbauend auf den Anforderungen an das Datenbanksystem - ein Entity-Relationship-Modell mit den ableitbaren Entities, Relationships, Attributen und Beziehungstypen. Machen Sie die Attribute kenntlich, die den Anforderungen zu entnehmen sind und ergänzen Sie das Entity-Relationship-Modell um weitere sinnvolle Attribute (z.B. Schlüsselattribute).

Aufgabe 2: Relationales Datenbankschema

a) Überführen Sie das Entity-Relationship-Modell in das relationale Datenbankschema.

b) Stellen Sie in Ihren Tabellen des relationalen Datenbankschemas folgenden Sachverhalt dar:
Der Kursteilnehmer "Müller" hat an Kursen für Anfänger und Fortgeschrittene teilgenommen. Er hat an allen Terminen teilgenommen. Die Kurse, an denen "Müller" teilgenommen hat, wurden vom Tanzlehrer "Meier" geleitet, der die Abendstunden von montags bis freitags bevorzugt.

Aufgabe 3: Verteilte Datenbanken

Das aufgrund Ihrer Modellierungsentscheidungen entstandene Datenbanksystem wird inzwischen von mehreren Tanzschulen mit Erfolg eingesetzt. Um den Verwaltungsaufwand zu reduzieren, soll das Datenbanksystem so erweitert werden, daß ein logisch und physisch dezentraler Datenbestand aufgebaut wird. Da die Mitarbeiter der Tanzschulen nicht über das dafür notwendige "Know How" verfügen, ist es Ihre Aufgabe, Lösungsvorschläge für eine "verteilte Datenbank" zu erarbeiten.

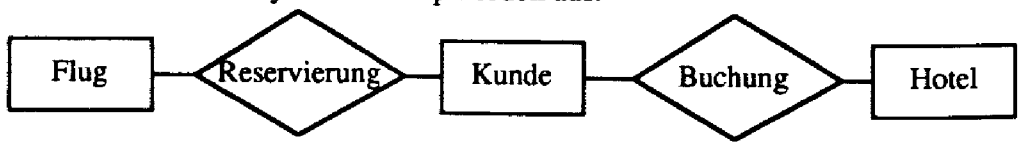
a) Definieren Sie den Begriff "Verteilte Datenbank" und grenzen Sie ihn von anderen Formen der Datenbank ab.

b) Für die Unterteilung der Daten können die Relationen unterschiedlich fragmentiert werden. Zeigen Sie anhand der Tanzschulen, wie die Relationen fragmentiert werden könnten. Gehen Sie davon aus, daß die Abrechnung der Kursgebühren und der verzehrten Getränke zentral erfolgen soll, ebenso die Anschreiben über weiterführende Kurse. Die Verwaltung der Stundenpläne, der benötigten Räumlichkeiten und der Teilnehmerinformationen soll jedoch weiterhin dezentral durch jede einzelne Tanzschule erfolgen.

c) Welches sind die Vorteile einer verteilten Datenbank?

Übung 6: ERM- und Datenmodellierung

Ein Reiseveranstalter möchte ein unternehmensweites Datenbanksystem aufbauen.
In einer frühen Phase des Entwurfs soll ein Entity-Relationship-Modell (ERM) entworfen werden. Es ist Ihre Aufgabe, das Unternehmen dabei zu unterstützen. Gehen Sie dazu von folgendem einfachen Entity-Relationship-Modell aus:



Aufgabe 1 ERM

Ergänzen Sie das ERM um Attribute, Schlüssel und Beziehungstypen.
Hinweis: Es sollten mindestens 5 Attribute pro Entity sein. Vergessen Sie nicht, daß auch Relationships Attribute besitzen können.

Aufgabe 2 Ergänzung eines ERM

Das vorgegebene ERM soll um weitere Entities und Relationships ergänzt werden. Das ergänzte ERM sollte folgende Anforderungen berücksichtigen:
- Personaldaten für Flugpersonal, Bodenpersonal und Hotelpersonal sollte gemeinsam verwaltet werden können.
- Informationen über Flughäfen und damit zusammenhängende Flugrouten und Zwischenlandungen sollten einbezogen werden.

- Aus den bisherigen Erfahrungen hat sich ergeben, daß die Informationen zu Flugzeugen, wenn sie als Attribut von dem Entity Flughafen verwaltet werden, nicht ausreichen.

Ergänzen Sie das E-R-M, so daß die Anforderungen berücksichtigt worden sind. Ergänzen Sie ihr ERM wie in Aufgabe 1 um Attribute, Schlüssel und Beziehungstypen.

Aufgabe 3 Überführung in Datenmodellierung

Überführen Sie ihr ERM in das relationale Datenmodell, das Netzwerkdatenmodell und das hierarchische Datenmodell.

Aufgabe 4 Zeitliche Aspekte

In dem Datenbanksystem eines Reiseveranstalters spielen Zeiten sicherlich eine große Rolle. Zeitbezogene Datenmodelle haben Elemente zur Berücksichtigung zeitlicher Aspekte.

a) Nennen Sie je 3 Zeittypen, Zeitarten und Zeitobjekte. Ergänzen Sie Ihr ERM mit jeweils einem Beispiel.

b) Welche der folgenden Fragen lassen sich in einer Datenbank , die aus dem ERM entstanden ist, beantworten? Begründen Sie Ihre Antworten und ergänzen Sie gegebenenfalls das ERM um zusätzliche Attribute.
   - Welche Mitarbeiter waren am 13.1.1990 in den Hotels beschäftigt und wie hoch war ihr Durchschnittseinkommen?
   - In welchem Zeitraum hat sich ein Kunde in einem bestimmten Hotel aufgehalten und welchen Flug hat er dabei benutzt?
   - Welche Mitarbeiter haben zu welchem Zeitpunkt vom Flug- zum Bodenpersonal gewechselt?

Aufgabe 5 Zeitorientierte Datenmodelle

Zeitorientierte Datenmodelle haben Elemente zur Berücksichtigung der zeitlicher Aspekte. Aus dem Entity Mitarbeiter ist die nachfolgende Relation entstanden. Für einen Mitarbeiter sind die gespeicherten Tupel in einer zeitorientierten Datenbank aufgeführt.

Gültigkeit	Mit.-Nr.	Name	Geb.-datum	Zuordnung	Einkommen	Zeitstempel
1.2.91	45	Meier	14.7.55	Flugpersonal	8500	2.1.91
1.9.90	45	Meier	14.7.55	Flugpersonal	8200	28.8.91
1.9.88	45	Meier	14.7.55	Bodenpersonal	6500	15.8.91
15.2.88	45	Meier	14.7.55	Bodenpersonal	5500	13.2.88

a) Zeitorientierte Datenmodelle unterscheiden unter anderem zwischen verschiedenen Zeittypen, Zeitarten und Zeitmaßstäben, zeitabhängigen Objekten und Zeitfolgen. Welche sind bei der aufgeführten Relation benutzt worden? Begründen Sie Ihre Antwort.

b) Zeitorientierte Datenmodelle lassen sich anhand der Zeitdimension typisieren? Welcher Zeittyp ist bei der aufgeführten Relation benutzt worden? Begründen Sie Ihre Antwort.

c) Durch die Einbeziehung zeitlicher Aspekte in ein Datenmodell oder eine Datenbank ergeben sich zusätzliche Probleme. Nennen Sie drei dieser Probleme und veranschaulichen Sie diese an der aufgeführten Relation.

Aufgabe 6 Verteilte Datenbank

Das von Ihnen entworfene ERM soll in einem verteilten Datenbanksystem abgebildet werden. Beim Entwurf eines verteilten Datenbanksystems hat die Fragmentierung eine wichtige Bedeutung.

a) Welche Arten der Fragmentierung gibt es?

b) Nennen Sie 3 Anforderungen an die Fragmentierung.

Geben Sie zu den Arten und den Anforderungen an die Fragmentierung jeweils ein sinnvolles Beispiel aus ihrem ERM an.

Übung 7: Externes Schema

Die DEWA-Sanitär GmbH verfügt über ein computergestütztes Warenwirtschaftssystem, mit dem sämtliche Bestellabwicklungen bei den Lieferanten durchgeführt werden. In der folgenden Abbildung sehen Sie einen vereinfachten Auszug der Bestellerfassungsmaske:

Bestellerfassung

DEWA-Sanitär GmbH 11.07.91 16.00h Sachbearbeiter: *199*

Bestellnummer: *00123456*
Lieferantennummer: *45315*
Lieferantenadresse: *Meyer KG*
Postfach 1420
4030 Ratingen

Bestellart: *03* Lieferwoche: *30* Bestelldatum: *11.07.91*
Versandart: *10* Termin: *Anfang* Bemerkung: *Eilbestellung*

Pos. Nr.	Artikel Nr.	Menge	ME	Artikelkurz-Bezeichnung	Listen-Preis	Rabatt %	Netto-Pos-Wert
01	239100	4	ST	MONADET weiß	500,00	5,00	1900,00
02	240100	9	ST	KOALA Waschtisch	300,00	3,00	2619,00

Nettoauftragswert: *4519,00*

a) Entwerfen Sie für die oben dargestellte Bestellmaske die Datenstrukturen mit Hilfe des Entity-Relationship-Modells.

b) Erweitern Sie das Entity-Relationship-Modell so, daß es von Bestellung bis Bezahlung der Ware sämtliche Zustände abbilden kann.

c) Erläutern Sie die Unterschiede des konzeptionellen und externen Schemas der 3-Schemata-Architektur nach ANSI/SPARC. Ordnen Sie die Bestellmaske ein und geben Sie drei Beispiele für Unterschiede im konkreten Fall an.

d) Lieferantenkonditionen und Preise ändern sich häufig im Zeitablauf. Erläutern Sie anhand der entsprechenden Relation, inwieweit diese bei der Überführung in ein zeitorientiertes Datenmodell erweitert werden müßte. Welche Zeiten sind in die Relation aufzunehmen und wie werden sie in zeitorientierten Datenbanken bezeichnet? Auf welche Weise können sie dargestellt werden?

Übung 8: Modellierung im Rahmen eines Warenwirtschaftssystems

Die Müsli Absatzgenossenschaft möchte aufgrund des immer stärker zunehmenden Angebots und der verstärkten Nachfrage nach biologisch-ökologisch angebauten Produkten ein datenbankgestütztes Warenwirtschaftssystem entwickeln, um die Verbraucher schneller mit preiswerten und qualitativ hochwertigen Produkten versorgen zu können. Das Warenwirtschaftssystem soll die Bereiche Bestellung, Wareneingang, Lagerung, Warenverkauf und Disposition unterstützen. Darüberhinaus sollen noch warenbezogene Auswertungen bzw. Statistiken generiert werden, um auf Verschiebungen innerhalb der Nachfrage nach Artikeln oder neuen Kauftrends gezielter reagieren zu können.

Aufgabe 1

Ihre Aufgabe ist, die Absatzgenossenschaft bei dem Entwurf der Datenbank für das Warenwirtschaftssystem zu unterstützen. Entwerfen Sie für diese Aufgabe ein Konzept, das die einzelnen Schritte aufzeigt, nach denen Sie vorgehen wollen. Begründen Sie ihre Vorgehensweise.

Aufgabe 2

Geben Sie einen einführenden Einblick, basierend auf dem Bereich des Bestellwesens, in das Entity-Relationship-Modell. Erläutern Sie die Bestandteile des ERM anhand der Objekte **ARTIKEL** und **LIEFERANT** (Grunddaten des Warenwirtschaftssystems) unter Berücksichtigung folgender Aspekte:

a) Entity und Relationship sowie einigen Attributen

b) graphische Darstellung der Elemente des ERMs

c) Beziehungstypen zwischen Entities und Relationships

d) Verdeutlichen Sie anhand der beiden genannten Objekte, welche möglichen Auswirkungen die Veränderung eines Beziehungstyps hat.

Aufgabe 3

a) Entwerfen Sie für das Bestellwesen mit Hilfe des ERMs eine umfassende Datenstruktur für die Objekte **ARTIKEL** und **LIEFERANT** mit allen typischen Attributen und klassifizieren Sie dabei die Schlüsselattribute, die in einem späteren Schritt die Primärschlüssel für das noch zu entwerfende logische Datenmodell bilden sollen.

b) Weiterhin soll das Warenwirtschaftssystem hierarchische Verdichtungsmöglichkeiten hinsichtlich der Zusammenfassung von Artikeln zu Warengruppen unterstützen, um den Anforderungen der Unternehmensführung nach Gesamtstatistiken für gleichartige oder ähnliche Artikel nachkommen zu können. Erweitern Sie entsprechend das ERM um die Warengruppen.

Aufgabe 4

Für die Bestelldurchführung benötigen die Einkäufer der Absatzgenossenschaft eine Übersicht über mögliche Angebotskonditionen, die nach Artikel- und Lieferantenkonditionen unterteilt werden. Die Artikelkonditionen berücksichtigen preisliche Abstufungen durch einen festen Rabattsatz bei einer Mindestabnahmemenge oder gewähren einen Preisnachlaß über eine Mengenstaffelung. Die Lieferantenkonditionen umfassen nur Rabatte auf die Gesamtsumme aller Bestellpositionen. Beim Erreichen oder Überschreiten einer Rabattschranke wird ein prozentualer oder absoluter Rabatt gegeben.
Entwickeln Sie für die beschriebenen Lieferanten- und Artikelkonditionen ein ERM und verbinden Sie das Modell mit dem bisherigen ERM der ARTIKEL und LIEFERANTEN.

Aufgabe 5

a) Überführen Sie das bisher entworfene ERM des Bestellwesens in das relationale DB-Modell. Inwieweit müssen Veränderungen bezüglich der Entities und Relationships sowie hinsichtlich der Beziehungen vorgenommen werden.

b) Beachten Sie bei der Überführung mögliche auftretende Redundanzen, die Sie mittels der Normalformenlehre (1.-3. Normalform) gegebenenfalls abbauen sollen. Beschreiben Sie die durchzuführenden Schritte der Normalisierung ausführlich und erläutern Sie dabei die besonderen Merkmale der einzelnen Normalformen.

Aufgabe 6 Datenbanksprachen

Nach der Datenmodellierung und der Datenbankschemabildung schließt sich die Implementierung des Datenbanksystems an. Diese Implementierung soll mit dBASE IV auf der Grundlage von SQL und QBE vorgenommen werden.

a) Überführen Sie das normalisierte relationale Datenmodell nach dBASE IV. Erläutern Sie an diesem Beispiel, welche Möglichkeiten SQL, QBE und dBASE IV bieten, um eine geeignete Datenstruktur aufzubauen.

b) Formulieren Sie die folgenden Abfragen und Datenmanpulationen in SQL und QBE. Erläutern Sie dabei die Befehle und die Syntax von SQL und QBE.
   b1) Für welchen Artikel liefert welcher Lieferant die günstigste Kondition?
   b2) Hat der Lieferant einen bestimmten Artikel im Programm und wie sehen seine Konditionen aus?
   b3) Welche Lieferanten gewähren neben den Artikelkonditionen auch Lieferantenkonditionen?
   b4) Ein weiterer Artikel wird in die Liste der Artikel aufgenommen und ein Lieferant wird aus der Lieferantenliste gestrichen.
   b5) Ein Lieferant verändert seine Lieferantenkonditionen.

c) Beschreiben Sie die dBASE IV spezifische Eigenheiten und zeigen Sie die Einschränkungen gegenüber "Standard"-SQL und "Standard"-QBE auf.

Aufgabe 7 Ergänzung des ERM

Bislang sind grundlegende Strukturen von ARTIKEL- und LIEFERANTEN-Zusammenhängen aus dem Warenwirtschaftssystem modelliert worden. Dieses Modell wurde anschließend um Aspekte der Artikel- und Lieferantenkonditionen ergänzt. Nun gilt es, die ARTIKEL-Modellierung zu verfeinern.

Erweitern Sie daher das ERM um Entities, Relationships und Attribute, um die folgenden Sachverhalte darstellen zu können:

a) Die Artikel können sich aus unterschiedlichen Gebindeeinheiten zusammensetzen, die noch zusätzlich nach Einkaufs- und Verkaufseinheit getrennt werden müssen. So wird Butter beispielsweise in einer Verpackungseinheit von 10 Stück á 250 g eingekauft, jedoch einzeln (pro Stück) verkauft. Generell lassen sich folgende Typen unterscheiden:
   - Der Artikel wird in der gelieferten Verpackungseinheit gesamt weiterverkauft.
   - Der Artikel wird nicht in der gelieferten Verpackungseinheit sondern nach der Maßeinheit des Verpackungsinhalts verkauft.
   - Der Artikel wird nicht in der gelieferten Verpackungseinheit verkauft, auch nicht in der Maßeinheit des Verpackungsinhalts, sondern es werden eigene Verkaufseinheiten gebildet (z.B. Mehl wird in 50 kg Säcken gekauft und in 5 kg oder 10 kg Selbstverpackung verkauft).

   Beachten Sie bei der Modellierung die Artikelbestandsverwaltung des Lagerwesens.

b) Neben dem Artikelattribut Einkaufspreis müssen auch die verschiedenen Verkaufspreise abgebildet werden. Diese Preise können starr sein oder mit der Verkaufsmenge variieren. Bei variablen Preisen ist eine Preisstaffel in Abhängigkeit von den unterschiedlichen Verkaufseinheiten zu berücksichtigen. Weiterhin kommen noch Verkaufskonditionen wie Rabatte hinzu.

Aufgabe 8 Ergänzung des ERM

Die Organisation der Absatzgenossenschaft basiert auf einem Hauptsitz und mehreren Verkaufsfilialen. Der Hauptsitz ist zentral für die gesamte Bestellabwicklung zuständig und koordiniert entsprechend den Artikeleinkauf und die Verteilung.

a) Erweitern Sie das ERM um die weiteren Filialen. Berücksichtigen Sie dabei eine Artikelzuordnung auf die Filialen.

b) Für die Bestellmengenabwicklung benötigt die Zentrale die monatlichen Bestandsdaten der einzelnen Artikel in den Filialen. Die Daten werden mittels elektronischen Datenaustausches vom Zentralrechner abgerufen und für die Auswertung zwischengespeichert. Entwickeln Sie in Verbindung mit der Datenübernahme ein ERM, um diese Daten für die spätere Bestellung und Bestellverfolgung effizient mit einer Datenbank verwalten zu können. Berücksichtigen Sie dabei auch die Daten der Zentrale.

Aufgabe 9 Normalformen

Überführen Sie die Erweiterung der ERM´s aus Aufgabe 7 und 8 in das relationale Modell und sichern Sie dabei die 1. bis 3. Normalform, um eventuell auftretende Redundanzen abzubauen.

Aufgabe 10 CODASYL-Ansatz (Netzwerk Datenmodell)

Eine Sprache für das Netzwerk-Datenmodell wurde von der CODASYL-Vereinigung entwickelt. Entsprechend dem Netzwerkmodell enthält die CODASYL-DDL (DDL = Data Definition Language) die zwei Grundelemente Records (Datensätze) und Sets (zur Erfassung von Beziehungen).

a) Beschreiben Sie anhand der Entities ARTIKEL und LIEFERANT sowie ARTIKEL und WARENGRUPPE den Zusammenhang zwischen Sets und Records.

b) Für Sets und Records werden häufig graphische Symbole ähnlich dem ERM verwendet. Überführen Sie das ERM aus Aufgabe 3 in die Netzwerk-Struktur.

c) Überlegen Sie sich mögliche Beispieldatensätze (Ausprägungen) und zeigen Sie anhand dieser Ausprägungen die Verkettung der einzelnen Datensätze im Netzwerkmodell.

d) Erläutern Sie anhand ihrer ausgewählten Ausprägungen, wie die Vorgehensweise bei folgenden Abfragen im Netzwerkmodell aussieht:
- Welche Artikel liefert die WESTFALIA GmbH?
- Zu welchen Warengruppen gehören die Artikel, die die WESTFALIA GmbH liefert?

Aufgabe 11 IMS - DL/1 (Hierarchisches Datenmodell)

Der Datenbankansatz von IMS - DL/1 basiert auf einer hierarchisch strukturierten Zusammenfassung von Objekten. Während die grundlegenden Bauelemente einer CODASYL-Datenstruktur der Record und der Set sind, ist das grundlegende Bauelement in DL/1 ein Baum.

a) Stellen Sie die grundlegenden Baumstrukturen anhand der Objekte ARTIKEL und ARTIKELKONDITION sowie LIEFERANT und LIEFERANTENKONDITION der Aufgabe 4 dar. Auf welche Weise werden Beziehungen zwischen den Objekten (Baumstrukturen) dargestellt.

b) Überführen Sie das ERM aus Aufgabe 4 in die hierarchische Datenbankstruktur und beschreiben Sie die Baumstrukturen graphisch.

c) Überlegen Sie sich mögliche Beispieldatensätze (Ausprägungen) und zeigen Sie anhand dieser Ausprägungen die Verkettung der einzelnen Datensätze im hierarchischen Datenmodell.

d) Erläutern Sie anhand ihrer ausgewählten Ausprägungen, wie die Vorgehensweise bei folgenden Abfragen im hierarchischen Datenmodell aussieht:
- Welche Artikel liefert die WESTFALIA GmbH?
- Zu welchen Warengruppen gehören die Artikel, die die WESTFALIA GmbH liefert?

Aufgabe 12

Die Müsli-Absatzgenossenschaft erweitert ihr EDV-Anwendungssystem um ein datenbankorientiertes Kostenrechnungssystem. Dieses Kostenrechnungssystem basiert auf der starren Plankostenrechnung. Einerseits können damit die Kostenstellen und Kostenarten des Unternehmens verwaltet werden, andererseits kann eine Verpflechtung zwischen den Kostenstellen und den Kostenarten aufgebaut werden. Weiterhin lassen sich mit diesem System die Kostenstellen zu Verdichtungskostenstellen aggregieren, um den Anforderungen der Geschäftsleitung nach Gesamtstatistiken nachkommen zu können.

a) Entwerfen Sie für das Kostenrechnungssystem mit Hilfe des ERM die Datenstrukturen für die Stammdatendateien der Kostenstellen und Kostenarten. Modellieren Sie die Kostenstellen/Kostenarten-Zuordnung.

b) Erweitern Sie das bisherige Modell um die Verdichtungsstrukturen der Kostenstellen.

c) Überführen Sie das entworfene ERM in das relationale Datenbankmodell. Beachten Sie bei der Überführung mögliche auftretende Redundanzen, die Sie mittels der Normalformenlehre (1. - 3. Normalform) gegebenenfalls abbauen sollen.

Aufgabe 13 Zeitorientierte Datenbanken

Bei den bisher entwickelten Datenmodellen handelt es sich mehr oder weniger um statische Datenmodelle bzw. Datenbanken (snapshot-databases). Sie berücksichtigen zwar die Änderung von Ausprägungen von Attributen, jedoch können die verschiedenen Veränderungen im Zeitablauf nicht nachvollzogen werden, da bei der Aktualisierung der Datenbestand in der Regel überschrieben wird.
Neuere Entwicklungsrichtungen in der Datenbankforschung versuchen den zeitlichen Aspekt im Datenmodell zu berücksichtigen, um unterschiedliche Versionen eines Datenbestandes halten und nachvollziehen zu können.

a) Kennzeichnen Sie den Unterschied zwischen zeitorientierten und statischen Datenbanken. Welche Zeitkonzepte werden unterschieden? Worauf können sich diese beziehen?

b) Klassifizieren Sie am Beispiel des Kostenrechnungssystems der Aufgabe 12 mögliche Zeittypen zeitorientierter Datenmodelle. Entwerfen Sie dazu beispielhaft einige Ausprägungen und zeigen Sie mögliche Änderungen dieser Ausprägungen im Zeitablauf.

c) Beschreiben Sie am Beispiel der Kostenstellenverdichtung folgende Arten zeitorientierter Datenmodelle und heben Sie eindeutig hervor, welche Zeittypen das jeweilige Modell verwendet und welcher zusätzliche Informationsgehalt für die Verdichtungsstrukturen gewonnen werden kann. Unterscheiden Sie zwischen
(b1) rollback database
(b2) historical database
(b3) temporal database.

d) Bei der Modellierung zeitorientierter Datenmodelle werden elementare Anforderungen an die Systeme gestellt, die gleichbedeutend für alle Arten dieser Modelle sind. Beschreiben Sie mögliche Anforderungen und Probleme bei der Umsetzung dieser Modelle in konkrete Datenbanksysteme.

Aufgabe 14 Verteilte Datenbanken

Die Müsli-Absatzgenossenschaft (vgl. Aufgabe 8) möchte Ihr Datenbanksystem um eine Personaldatenverwaltung erweitern. Dazu ist vorgesehen, daß die einzelnen Filialen die Personaldaten dezentral in den Filialen verwalten.

a) Ergänzen Sie das ERM um die dafür benötigten Entities, Relationships und Attribute und überführen Sie Ihr ERM in das relationale Datenmodell.

b) Die Verwaltung der Personaldaten soll mit einem "verteilten" Datenbanksystem realisiert werden. Dabei könnten einerseits die gesamten Personaldaten zentral in einem Rechenzentrum der Absatzgenossenschaft verwaltet werden, andererseits könnten die Daten der jeweiligen Filiale auch in der Filiale gehalten werden.
   b1) Was versteht man unter einem "verteilten" Datenbanksystem. Unterscheiden Sie dabei zwischen File-Sharing-Architekturen, Client-Server-Architekturen sowie verteilten Datenbanken.
   b2) Beim Entwurf einer verteilten Datenbank wird die Konstruktion und Modellierung des Globalen Schemas entsprechend zentralisierten Datenbanken vorgenommen. Danach erfolgt die Umsetzung in ein verteiltes Datenbankschema erweitert um die Bildung des fragmentierten Datenbankschemas. Stellen Sie anhand der Personaldatenverwaltung und der Artikelstruktur der Müsli-Absatzgenossenschaft Möglichkeiten der Fragmentierung dar.

b3) Nehmen Sie eine Bewertung der verschiedenen Fragmentierungsmöglichkeiten vor. Unterscheiden Sie bei der Bewertung zwischen der Datenunterteilung, der Datenzuordnung und der Datenwiederholung.

Literatur

Abbod, T./Brown, K./Noble, N.: Providing time-related constraints for conventional databases, in: Proceedings of 13th International. Conference Very Large Data Bases, Brighton 1987, S. 167 - 175

Alagic, S.: Object-Oriented Database Programming, New York - Berlin - Heidelberg 1988

Al-Fedaghi, S. S.: An Entity-Relationship Approach to Modelling Petroleum Engineering Database, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.): Entity Relationship Approach to Software Engineering, Amsterdam 1983, S. 761-779

Ariav, G.: Towards a multi homogeneous model for a temporal database, in: Proceedings 2nd International. Conference. Data Engineering, Los Angeles 1986, S. 390 - 397

Ariav,G.: Preserving the time dimension in information systems, Diss. Univ. of Pennsylvania 1983

Bachman, C. W.: The Structuring Capabilities of the Molecular Data Model, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.): Entity Relationship Approach to Software Engineering, Amsterdam 1983, S. 55-68

Bancilhou, F.: Object-Oriented Database Systems, in: ACM Database Systems (1988), S. 152-162

Bayer, R./ Elhardt, K./Kiessling, W./ Killar, D.: Verteilte Datenbanksysteme, in: Informatik-Spektrum 7 (1984), 1, S. 1-19

Beha, H.J./Huy, H.D: Strategische Planung der Informationsverarbeitung als Komponente der Unternehmensplanung, in: Handbuch der Modernen Datenverarbeitung HMD 154, S. 61ff.

Ben-Zvi, J.: The Time Relational Model, Diss. Univ. of California Los Angeles 1982

Berner, C.: Flexibilität statt Taylorismus, in: Computer Zeitung (1991), 7, S. 58

Blokdijk, A. und P.: Planning and Design of Information Systems, Academic Press, 1987

Böhnlein, P.G./Nittel, S./Dittrich, K.R.: Semantische Datenmodelle, in: Handbuch der modernen Datenverarbeitung HMD - Theorie und Praxis der Wirtschaftsinformatik 152 (1990), S. 116-127

Brodie, M.L./ Myolopoulos, L.J./ Schmidt, J.W. (eds): On conceptual modeling: Perspectives from artificial intelligence, databases and programming languages, Berlin etc. 1984

Brodie, M.L./Ridjanovic, D.: On the Design and Specification of Database Transactions, in: Brodie, M.L./Mylopoulos, J./Schmidt, J.W. (eds.): On Conceptual Modelling, Berlin etc. 1984, S. 277-312

Bubenko, J.A.: Information Modelling in the Context of System Development, in: Lavingtion, S.H. (ed.): Information Processing 80, Amsterdam 1980, S. 395-411.

Bubenko, J.A.: The Temporal Dimension in Information Modeling, in: Nijssen, G.M. (ed.); Architecture on Models in Database Systems, Amsterdam 1977, S. 93-117.

Cash,J.I./ Kosynski, B.R.; IS redraws competitive boundaries, in: Harvard Business Review (1985),March/April, S.135-142

Ceri, S./Pelagatti, G.: Distributed Databases-Principles & Systems, New York 1985.

Chen, P.P.S.: Database Design Based on Entity and Relationship, in: Bing, Y. (ed.): Principles of Database Design, Vol. 1: Logical Organizations, 1985, S. 174 - 210

Chen, P.P.S.: The Entity-Relationship Model - A Basis for the Enterprise View of Data, in: Proceedings of the National Computer Conference 1977, S. 77-84

Clifford, J./ Croker, A.:Objects in time; in: Data Engineering Bulletin (1988), S.189-196

Clifford, J./Warren, D.S.: Format Semantics for Time in Databases, in: ACM Transaction on Database Systems 8(1983), 2, 214-254

Codd, E.F.: A Relational Model for Large Shared Data Banks, in: Communications of the ACM 17(1970), 6, S. 377- 387

Czap, H.: Beständigkeit versus Flexibilität von Informationsstrukturen, Manuskript Trier 1991

Dabrowski, C.E./Fong, E.N./Yang, D.: Object Database Managment Systems: Concepts and Fearures, in: NIST Special Publication 1990

Dadam, P./Linnemann, V.: Advanced Information Management (AIM): Advanced Database Technology for integrated applications, in: IBM Systems Journal 28(1989), 4, S. 660-681

Dadam, P./Lum, V./Werner, H.D.: Integration of Time into a Relational Database System, in: Dayal, U./Schlageter, G./Seng, L.H. (eds.), Proceedings of the 10th International Conference on Very Large Data Bases, Singapore 1984, S. 510-522.

Dampney, C.N.G.: Specifying a Semantic Adequate Structure for Information Systems and Databases, in: March, S.T. (ed.), Entity Relationship Approach, Amsterdam 1988, S. 165-188.

Daniel, D.R.: Management Information Crisis, in: Harvard Business Review 39 (1961), 5, S.111-121

Detemple, W./Mack, R.: Die On-Line Recherche und was dann?, in: DJ (1991), 12, 25

Dittmann, E.L.: Datenunabhängigkeit beim Entwurf von Datenbanksystemen, Darmstadt 1977.

Dittrich, K.P.: Objektorientierte Datenmodelle als Basis komplexer Anwendungssysteme - Stand der Entwicklung und Einsatzperspektiven -, in: Wirtschaftsinformatik 32(1990), 3, S. 228-237

Dittrich, K.P.: Objektorientierte Datenbanksysteme, in: Informatik-Spektrum 12(1989), S. 215-218.

Dittrich, K.P.: Object-oriented Database systems - A workshop report, in: Spaccapietra, S. (ed.), Entity Relationship Approach, Amsterdam 1987, S. 51-

Dittrich, K.P.: Object oriented Database Systems: the notion and issues, in: Dittrich,K./Danjal,U.(eds.): International Workshop on Object - Oriented Database Systems, Pacific Grove 1986, S.2-4

Dudler, A.: Entwurf verteilter Datenbanken, Diss. ETH Zürich 1986

Dürr, M./Radermacher, U.: Einsatz von Datenbanksystemen - Leitfaden für die Praxis, Berlin - Heidelberg - New York 1990

Eder, J./Kappel, G./Tjoa, A.M./Wagner, R.R.: BIER - The Behaviour Integrated Entity Relationship Approach, in: Spaccapietra, (ed.), Proceedings of the 5. Entity Relationship Approach Conference Nov. 1986 (Amsterdam), S. 147-166

Eftimie, R.A./Nikles, J.C.: Information Architecture and Entity-Relationship Approach, in: March, S.T. (ed.), Entity Relationship Approach, Amsterdam 1988, S. 427-438

Everest,G.C./ Weber, R.: A Relational Approach to Accounting Models, in: The Accounting Review 52(1977),2,S.340-359

Ferstl, O. K./Sinz, E. J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im semantischen Objektmodell SOM, in: Wirtschaftsinformatik 33 (1991), 6, 477-491

Ferstl, O. K./Sinz, E. J.: Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM), in: Wirtschaftsinformatik 32 (1990), 6, S. 566-581

Fischer, J.: Qualitative Ziele in der Unternehmensplanung, Berlin 1989

Fischer, J./Herold, W.: Einführung in die Datenverarbeitung, Skript des Schwerpunktes Wirtschaftsinformatik & Operations Research der Universität Paderborn, April 1992

Flavin, M.: Fundamental Concepts of Information Modeling, New York 1981

Gray, J.N.: An Approach to Decentralised Computer Systems, in: IEEE Transactions on Software Engineering, Vol. SE-12(1986), 6, S. 684 - 692

Grochla, E./Meller, F.: Datenverarbeitung in der Unternehmung, Reinbek (bei Hamburg) 1974/77.

Gunson, N./Boddy, D.: Computer network systems managerial problems and opportunities, in: Journal of General Management, 15(1989), 2, S. 41 - 56

Gutenberg, Erich: Grundlagen der Betriebswirtschaftslehre, Bd. 1: Die Produktion: 24. Auflage, Berlin 1983

Härder, Theo: Überlegungen zur Modellierung und Integration der Zeit in temporalen Datenbanken, Arbeitspapier Universität Kaiserslautern (1984), 19

Hammer, M./McLeod, D.: The Semantic Data Model: A Modelling Mechanism for DataBase Applications, in: Proceeding SIGMOD Conference 1978, S. 26-36

Hax, A.C./Majluf, N.S.: Strategic Management - An integrative perspective, Englewood Cliffs 1985

Horvath, P.: Controlling, 3. Aufl. München 1990

Hull, R./King, R.: A Tutorial on Semantic Database Modeling, in: Cárdenas, A.F./McLeod, D.V. (eds.); Research Foundations in Object Oriented and Semantic Database Systems, Englewood Cliffs 1988, S. 1-33.

Iivari,J./ Koskela, E.: An extended EAR approach for Information System Specification, in: Davis,C.G. u.a. (eds): Entity Relationship Approach to Software Engineering, Amsterdam 1983, S. 605 - 636

Jackson, M.A.: System Development, Englewood Cliffs 1983

Kappel, G./Schreffl, M.: A Behavior Integrated Entity - Relationship Approach for the Design of Object-Oriented Databases, in: Batini, C. (ed.), Entity Relationship Approach, 1989, S. 311-328

Kaufman, F.: Data systems that cross companies boundaries, in: Harvard Business Review (1966),Jan/Febr., S.141

Klopprogge, M.R. : TERM: An approach to include the time dimension in the entity-relationsship-model, in: Chen, P. (ed.): Entity - Relationsship Approach to Information Modeling and Analysis, Amsterdam 1983, S. 475-508

Klopprogge, M.R. : Gegenstands- und Zustandsgeschichten: Ein Konzept zur Beschreibung und Verwaltung zeitveränderlicher Informationen in Datenbanken, Diss. Universität Karlsruhe 1983.

Klopprogge, M./Lockemann, P.C.: Modeling information preserving databases: Consequences of the concept of time, in: Proceedings of the 9th International Conference Very Large Data Bases, Florenz 1983, S. 399-416

Knolmayer, G.: Die Berücksichtigung des Zeitbezugs von Daten bei der Gestaltung computergestützer Informationssysteme, in: Hax, H./ Kern, W./ Schröder, H.H.: Zeitaspekte in betriebswirtschaftlicher Theorie und Praxis, Stuttgart 1989, S. 77-88

Knolmayer, G./Bölzel, S./Disterer, G: Zeitbezogene Daten in betrieblichen Informationssystemen - Ein Vergleich alternativer Modellierungsansätze an einem Beispiel der Finanzplanung, in: Rückle, D.; Aktuelle Fragen der Finanzwirtschaft und der Unternehmensbesteuerung, Wien 1991, S. 287-319.

Kosiol, E.: Organisation der Unternehmung, Wiesbaden 1962

Lindgreen, P.: Entity sets and their Description, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.), Entity-Relationship Approach to Software Engineering, Amsterdam 1983, S. 91-110

Ling, T.W.: A Three Level Schema Architecture ER-based Data Base Management System, in: March, S.T. (ed.), Entity Relationship Approach, Amsterdam 1988, S. 205-220

Ling, D. H. O./Bell, D. A.: Taxonomy of time models in databases, in: Information and Software Technology 32(1990),3, S.215-224

Lockemann, P.C./Dittrich, K.R.: Architektur von Datenbanksystemen, in: Lockemann, P.C./Schmidt, J.W. (Hrsg.); Datenbank-Handbuch, Berlin-Heidelberg-New York 1987, S. 85-161

Lockemann, P.C./Radermacher, K.: Konzepte, Methoden und Modelle zur Datenmodellierung, in: Handbuch der Modernen Datenverarbeitung - Theorie und Praxis der Wirtschaftsinformatik 152 (1990) , S. 3-16

Lum,V./ Dadam,P./ Erba,R. etc.: Designing DBMS support for the temporal dimension, in: Proceedings ACM-SIGMOD International Conference Management Data, 1984, S.115-130

Mann, R.: Wir brauchen den offenen Führungsstil; in: Blick durch die Wirtschaft (1984), 30. Nov.

Martin, J.: Einführung in die Datenbanktechnik, München - Wien 1988

Mauri, G. A./Walter, W.: Strategien als Verwaltungsalgorithmen - Das Client/Server-Modell als Basis-Architektur zur Realisierung von verteilten Systemen, in: Computer-Zeitung (1991), 7, S. 42

Mayr, H.C./Dittrich, K.R./Lockemann, P.C.: Datenbankentwurf, in:Lockemann, P.C./ Schmidt, J.W. (Hrsg.), Datenbank-Handbuch, Berlin-Heidelberg-New York 1987, S. 482-557

Mees,M./Put,F.: Extending a dynamic modelling method using data modelling capabilities - the case of JSD, in: Spaccapietra,P.(ed.) Entity Relationship Approach, Amsterdam 1987, S.

Mertens, P./Bodendorf, F./König,W./Picot, A./Schumann, M.: Grundzüge der Wirtschaftsinformatik, Berlin etc., 1991

Mertens, P./Holzner, J.: Gegenüberstellung von Integrationsansätzen der Wirtschaftsinformatik, Arbeitspapier Universität Erlangen-Nürnberg-Abteilung Wirtschaftsinformatik (1991), 5,

Mertens, P./Schuhmann, M./Hohe, U.: Nutzeffekte strategischer Informationsverarbeitung, in: Angewandte Informatik 30(1988),12, S.515 - 528

Mertes, H./Klonki, U: Vorgehensweise für die Erstellung eines unternehmensweiten Datenmodells bei der Hoesch AG, in: Wirtschaftsinformatik 33 (1991), 4, 308 - 315

Müller-Merbach, H.: Komprehensive Informationssysteme und Allgemeine Betriebswirtschaftslehre, in: Zeitschrift für Betriebswirtschaft 59(1989), 10, 1023-1045

Niedereichholz, B.: Datenbanksysteme - Aufbau und Einsatz, 3. Aufl. Würzburg-Wien 1983

Österle, H.: Entwurf betrieblicher Informationssysteme, München 1981

Österle, H.: Techniken zum Entwurf betrieblicher Anwendungsprogramme, in: Angewandte Informatik 6(1977), S. 249-257

Olivé, A.: Analysis of Conceptual and Logical Models in Information System Design Methodologies, in: Olle, T.W./Sol, H.G./Tully, C.J. (eds.); Information Systems Design Methodologies, Amsterdam 1983, S. 63-85.

Ortner, E.: Informationsmanagement - wie es entstand und wohin es sich entwickelt, in: Informatik-Spektrum 14 (1991 c), S. 315-327

Ortner, E.; Ein Referenzmodell für den Einsatz von Dictionary-/Repository-Systemen in Unternehmen, in: Wirtschaftsinformatik 33 (1991 b), S. 420-430

Ortner, E.: Unternehmensweite Datenmodellierung als Basis für integrierte Informationsverarbeitung in Wirtschaft und Verwaltung, in: Wirtschaftsinformatik 33(1991), 4, S. 269-280.

Ortner, E.: Semantische Datenmodellierung - Datenbankentwurf auf der Ebene der Nutzer, in: Informatik-Spektrum 8(1985), S. 20-28.

Ortner, E./Rössner, J./Söllner, B.: Entwicklung und Verwaltung standardisierbarer Datenelemente, in: Informatik-Spektrum 13 (1990), S. 17-30

Ortner, E./Söllner, B.: Konzept und Einsatz eines Data Dictionary bei DATEV, in: Informatik - Spektrum 12(1989b), S. 82-92

Ortner, E./Söllner, B.: Semantische Datenmodellierung nach der Objekttypenmethode, in: Informatik-Spektrum 12(1989), S. 31-42

Pohle, K.: Kritische Analyse des Management-Informationssystems aus der Sicht des Vorstandes, in: Küpper, H.U./ Mellwig, W./ Moxter,A./ Ordelheide,D. (Hrsg): Unternehmensführung und Controlling, Wiesbaden 1990, S.1-17

Porter, M.E.: Competition in Global Industries: A conceptual framework, in: Porter, M.E. (ed.): Competition in Global Industries, Boston (Mass.) 1986

Porter, M.E./Millar, V.E.: How Information gives you Competitive Advantage, in: Harvard Business Review 63(1985), July/August, S. 149-160

Put, F.: The ER Approach Extended with the Action Concept as a Conceptual Modelling Tool, in: Batini,C.(ed.): Entity Relationsship Approach, Amsterdam 1989, S.423-440

Reuter, Andreas: Verteilte Datenbanksysteme, in: Computer-Magazin (1988), 11, 41-50

Reuter, A./Schiele, G./Zeller, H.: Nichtprozedurale Datenbanksprachen, in: Informationstechnik 30 (1988),6,S. 387 - 403

Riebel, P.: Probleme der Abbildung zeitlicher Strukturen im Rechnungswesen, in: Hax, H./ Kern, W./ Schröder, H.H.: Zeitaspekte in betriebswirtschaftlicher Theorie und Praxis, Stuttgart 1989, S. 61-76

Robinson, K.A.: An entity / event data modelling method, in: The Computer Journal 22(1970),3, 270-281

Rockart, J. F.: Chief Executive Define Their Own Data Needs, in: Harvard Business Review 57 (1979), 2, S. 81-92

Sakai, H.: A Method for Entity Relationship Behavior Modeling, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.), Entity Relationship Approach to Software Engineering, Amsterdam 1983, S. 111-129

Scheer; A.W.: Architektur integrierter Informationssysteme - Grundlagen der Unternehmensmodellierung, Berlin-Heidelberg-New York 1991

Scheer, A.W.: Modellierung betriebswirtschaftlicher Informationssysteme, in: Wirtschaftsinformatik 32 (1990c), 5, S. 403-421

Scheer, A.W.: EDV-orientierte Betriebswirtschaftslehre, 4. Auflage, Berlin etc. 1990b

Scheer, A.W.: Wirtschaftsinformatik - Informationssysteme in einem Industriebetrieb, 3. Aufl. Berlin 1990

Scheer, A.W.: Entwurf eines Unternehmensdatenmodells, in: Information Management (1988b), S. 14-23.

Scheer, A.W.: Unternehmensdatenmodell als Grundlage integrierter Informationssysteme, in: Zeitschrift für Betriebswirtschaft 58(1988), 10, 1091-1113

Schiel, U.: An Abstract Introduction to the Temporal Hierarchic Data Model, in: Proceedings of the 9.th International Conference on Very Large Data Bases, Florenz 1983, S.323-330

Schlageter, G./ Stucky, W.: Datenbanksysteme: Konzepte und Modelle, 3. Aufl., Stuttgart 1983

Schmidt, J.W.: Datenbankmodelle, in: Lockemann, P.C./ Schmidt, J.W. (Hrsg.), Datenbank-Handbuch, Berlin-Heidelberg-New York 1987, S. 4-82

Schönthaler, F.: Grundlagen des konzeptuellen Datenbankentwurfs, in: Software-Werkzeuge- Konzeption und Realisierung, Arbeitstagung zum DFG-Schwerpunktprogramm: Interaktive betriebswirtschaftliche Informations- und Steuerungssysteme - Tagungsbericht Karlsruhe 1989, S. 15-43

Scholl, M.H./Schek, H.J.: Evolution von Datenmodellen, in: Handbuch der modernen Datenverarbeitung HMD - Theorie und Praxis der Wirtschaftsinformatik 152 (1990), S. 103-115

Scholz-Reiter, B.: CIM-Informations- und Kommunikationssysteme, München-Wien 1990

Schueler, B.M.: Update reconsidered, in: Nijssen, G.M.(ed.): Architecture and Models in Database Management Systems, Amsterdam etc. 1977, S. 149-164

Schuldt, G.: ER-Based Acess Modelling, in: Spaccapietra, S. (ed.); Entity Relationship Approach, 1987, S. 233-251

Schumann,M.: Abschätzung von Nutzeffekten zwischenbetrieblicher Informationsverarbeitung, in: Wirtschaftsinformatik 32 (1990),4, 307-319

Sinz, E.J.: Das Entity-Relationship-Model und seine Erweiterungen, in: Handbuch der Modernen Datenverarbeitung HMD - Theorie und Praxis der Wirtschaftsinformatik 152 (1990), S. 17-29

Smith, J.M./Smith, D.C.P.: Database Abstraction: Aggregation, in: Communication of the ACM (1977b), June, S. 405-413

Smith, J.M./Smith, D.C.P.: Database Abstraction: Aggregation and Generalization, in: ACM Transactions on Database Systems (1977), February, S. 105-133

Sneed, M./Gawron, W.R.: The Use of the Entity/Relationship Model as a Schema for Organizing the Data Processing Activities at the Bavarian Motor Works, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.), Entity Relationship Approach to Software Engineering, Amsterdam 1983 , S. 717 - 730

Snodgrass, R.: The temporal query language TQUEL, in: ACM Transactions on database systems 12(1987),2, S. 247-298

Snodgrass, R./Ahn,I.: Temporal Databases, in: Computer 19(1986b), 9, S.35-42

Snodgrass, R./Ahn, I.: Performance Evaluation of a Temporal Database Management, in: Proceedings International Conference Management of Data, ACM SIGMOD, Washington 1986, S. 96-107

Spitta, T.: Software Engineering und Prototyping, Berlin-Heidelberg-New York 1989

Studer, R.: Modeling Time Aspects of Information Systems, in: IEEE Computer Society (eds); Proceedings International Conference on Data Engineering, Washington 1986, S.364-373

Studer, R./Horndasch, A.: Modeling Static and Dynamic Aspects of Information Systems, in: Meersman etc.(eds.): Proceedings of the Working Conference Database Semantics, 1985

Su, S.Y.W.: Database Computers - Principles, Architectures and Techniques, New York etc. 1988.

Tansel, A.U.: An extension of relational algebra to handle time in relational databases, in: Proceedings International Conference Management of Data, ACM-SIGMOD, Austin 1985, S. 247-265

Tasker, D.: An Entity/Relationship View of Time, in: Marock, S.T. (ed.), Entity Relationship Approach, Amsterdam 1988, S. 237-247

Thome, R.: Wirtschaftliche Informationsverarbeitung, München 1990

Tulowitzky, U.: Anwendungssystemarchitekturen im strategischen Informationsmanagement, in: Wirtschaftsinformatik 33 (1991), 2, S. 94-99

Urban, S.D./Delcambre, L.M.L.: An Analysis of the Structural Dynamic and Temporal Aspects of Semantic Data Models, in: International Conference on Data Engineering, Denver 1986, S. 382-389

Vetter, M.: Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung, 4. Aufl. Stuttgart 1987

Vetter, M.: Strategie der Anwendungssoftware-Entwicklung - Planung, Prinzipien, Konzepte, Stuttgart 1988

Vossen, G./ Witt, K.U.: Zur Klassifikation von strukturellen Erweiterungen des relationalen Datenmodells, in: Informationstechnik 32(1990),2,S.97 - 106

Wedekind, H.: Datenbanksysteme, 2. Aufl. Zürich 1981

Wedekind, H./Ortner, E.: Systematisches Konstruieren von Datenbankanwendungen, München-Wien 1980

Wenner, T.: Datenbankunterstützung für CASE-Entwicklungsumgebungen, in: Wirtschaftsinformatik 33(1991), 1, S. 33-39.

Wiederhold, G.: Database Design, 2. ed., Hamburg, New York [u.a.] 1983.

Wiederhold, G.: Dateiorganisation in Datenbanken: Hamburg etc. 1989

Winkler, J.: The Entity-Relationship Approach and the Information Resource Dictionary System Standard, in: Batini, C. (ed.): Entity Relationship Approach, Amsterdam 1989, S. 3-19

Zachman, J.A.: A Framework for Information Systems Architecture, in: IBM Systems Journal 26(1987), 3, S. 276-292

Zehnder, C.A.: Informationssysteme und Datenbanken, 5. Aufl. Stuttgart 1989

Ziekursch, W.: Was das relationale Modell leistet? in: CHIP-Spezial „INGRES-Datenbankmanagement der Zukunft," (1989), S. 7-10

Zimmermann, R. P.: Phases, Methods and Tools - A Triad of System Development, in: Davis, C.J./Jagodia, S./Ng. P.A./Yeh, R.T. (eds.), Entity Relationship Approach to software Engineering, Amsterdam 1983, S.131-151

Indexverzeichnis