Papiere schreiben anno dazumal:
Ein Beispiel aus dem EMISA Forum, Heft 10, 1986

EIN ANSATZ ZU EINEM FORMALISIERTEN DATENBANK-ENTWURF IN DER PRAXIS

Helmut Thoma
CIBA-GEIGY AG
Basel (Schweiz)

ZUSAMMENFASSUNG

Ein Entity-Relationship-Modell als Diskussionsbasis mit den Benutzern beim Datenanalyse-Prozess und Relationen in 3NF zur Integration applikationsspezifischer Datenmodelle zu einem applikationsneutralen Modell des betrieblichen Wirkungsfeldes von Informationssystemen: Das sind die wesentlichen Entwurfsebenen des hier beschriebenen Ansatzes auf dem Weg von der Realität zu den Datenbanken. Die normalisierten Relationen sind hierbei aus dem Entity-Relationship-Modell algorithmisch ableitbar.

Obwohl die Kenntnis der Grundlagen von Relationen und ihrer Normalformen für die Anwendung der Methode nicht benötigt wird, wird für denjenigen Leser ein kurzer Einstieg ermöglicht, der mit Relationen wenig vertraut ist. Anschliessend wird die Vorgehensweise beschrieben und anhand von Beispielen verdeutlicht. Zum Schluss werden einige Anmerkungen zur Praxis diskutiert.

1. EINLEITUNG

Die Ueberprüfung und Neugestaltung der Projektstandards für die Entwicklung kommerzieller EDV-Systeme in den Management Services bei CIBA-GEIGY in Basel bot eine günstige Gelegenheit, die praktizierte Methodik des Datenbank-Entwurfs zu überdenken und neu zu gestalten. Die bisherige und grossenteils noch heutige Praxis war und ist gekennzeichnet durch unterschiedliche, projektgruppenspezifische, nichtformale Vorgehensweisen, die in starkem Masse vom individuellen Kenntnisstand der Mitarbeiter geprägt sind. Intuitive und nichtformale Methoden enthalten die Gefahr fehlender Transparenz und Nachvollziehbarkeit von Entwurfs-Entscheidungen und führen zu einem bereits in frühen Entwurfs-Phasen vorherrschendem Denken in schnittstellenspezifischen Strukturen des verwendeten Datenbanksystems. Dies bedeutet, dass logische Zusammenhänge der Daten bereits in einem frühen Stadium des Modellierungsprozesses physischen Zwängen und dem Gesichtspunkt der Realisierung preisgegeben werden.

Wesentliches Ergebnis der neuen Methodik des Datenbank-Entwurfs ist eine schnittstellenneutrale Ebene zwischen der Realität und dem datenbankspezifischen Schema zur Beschreibung ausschliesslich logischer Zusammenhänge der Daten. Hierzu bedienen wir uns des Relationenmodells. Das betriebliche Wirkungsfeld mit seinen Objekten, Individuen, Ereignissen und Konzepten wird unter rein logischen Gesichtspunkten auf normalisierte Relationen abgebildet, erst anschliessend erfolgt die Abbildung unter Einbezug applikatorischer und systemspezifischer Faktoren auf die Schnittstelle des verwendeten Datenbanksystems (momentan IMS oder TOTAL).

Ziel dieses Beitrages ist es, die verschiedenen Phasen des Datenbank-Entwurfs vorzustellen. Ein Schwerpunkt liegt hierbei in der Beschreibung des Weges vom betrieblichen Wirkungsfeld über ein semantisches Modell (Datenbeschreibung) zu normalisierten Relationen. Im weiteren sollen einige Gesichtspunkte zum praktischen Einsatz dieser Methode besprochen werden.

2. GRUNDLAGEN DER RELATIONENTHEORIE

Im folgenden wird lediglich ein kurzer Einblick in die Grundlagen von Relationen und Normalformen gegeben. Dieser Abschnitt ist nicht als Einstieg in das Relationenmodell für den Anfänger gedacht. Hierzu wird auf entsprechende Lehrbücher verwiesen, z.B. auf [Dat 81]. Sein Ziel ist vielmehr, die verwendete Terminologie festzulegen und dem Leser eine kurze Wiederholung des Relationenmodells zu bieten.

Eine Relation ist eine Menge geordneter n-Tupel (d1,d2,...,dn). Hierbei ist di ein Element einer Menge Di, den sog. Domänen der Relation. Die Domänen müssen nicht unterschiedlich sein. Die Bedeutung einer Domäne in einer Relation wird durch das sog. Attribut festgelegt. Einem Attribut und einer Domäne müssen unterschiedliche Namen gegeben werden, wenn in derselben Relation gleiche Domänen mit unterschiedlicher Bedeutung (oder Rollen) vorkommen (z.B. Stückliste).

Es ist üblich, eine Relation als Tabelle darzustellen: Eine Zeile repräsentiert ein n-Tupel der Relation, eine Spalte ein bestimmtes Attribut. Eine Relation (besser: der Typ einer Relation) wird somit durch ihre Attribute definiert.

Jede Relation besitzt mindestens ein Attribut oder eine Kombination von Attributen, dessen Wert einmalig ist und somit ein Tupel identifiziert (im allgemeinsten Fall das gesamte Tupel). Dieses Attribut resp. die Kombination nennen wir Primärschlüssel. Dabei muss beim kombinierten Primärschlüssel die Kombination minimal sein, d.h. dass bei einem Weglassen einer Komponente die identifizierende Eigenschaft der Kombination verloren geht. In einer Relation kann es noch weitere Attribute resp. Kombinationen mit der Eigenschaft eines Primärschlüssels geben. Alle diese sind wie der Primärschlüssel sog. Schlüsselkandidaten. Fremdschlüssel einer Relation R1 ist ein Attribut, das nicht Primärschlüssel in R1 ist, dessen Werte jedoch Werte des Primärschlüssels einer Relation R2 sind. Primär- und Fremdschlüssel repräsentieren Verbindungen zwischen Tupeln (im allgemeinen zwischen unterschiedlichen Relationen, jedoch müssen R1 und R2 nicht verschieden sein).

Wenden wir uns nun dem Gebiet der Normalisierung von Relationen zu. Zunächst beschreiben wir die beim Datenbank-Entwurf äusserst wichtige funktionale Abhängigkeit in einer Relation: Ein Attribut oder eine Kombination von Attributen B einer Relation R ist von einem Attribut oder einer Kombination von Attributen A derselben Relation R genau dann funktional abhängig, wenn es zu einem bestimmten Wert von A höchstens einen Wert von B gibt. B ist von einer Kombination von Attributen A genau dann voll funktional abhängig, wenn es von A funktional abhängig ist, jedoch nicht funktional abhängig von irgendeiner Teilmenge der Attribute von A. Transitive Abhängigkeit: In einer Relation R sei A der Primärschlüssel, B und C seien zwei weitere Attribute oder Kombinationen von Attributen in R (A, B und C untereinander verschieden). C ist transitiv abhängig von A, falls C von B funktional abhängig ist, B jedoch kein Schlüsselkandidat von R ist (oder anders ausgedrückt: A von B nicht funktional abhängig ist).

Mit diesen Kenntnissen ausgerüstet, können wir die Ideen der Normalisierung und der unterschiedlichen Normalformen diskutieren. Wir sprechen dann ganz allgemein von normalisierten Relationen, wenn jeder Attribut-Wert in jedem Tupel atomar, also nicht zerlegbar ist (auch "flache Relationen" genannt). Dies bedeutet, dass in jeder Spalten-Zeilen-Position einer Tabelle höchstens ein skalarer Wert existiert.

Mit Hilfe normalisierter Relationen soll die Verletzung realitätskonformer Sachverhalte im Datenbank-Schema verhindert werden. Logische Zusammenhänge der Realität werden durch Beachtung funktionaler Abhängigkeiten in der Relation realitätstreu modelliert. Hierzu wurden von verschiedenen Autoren unterschiedliche Grade der Normalisierung definiert (sog. Normalformen), die mehr oder weniger Fakten der Realität in der Datenstruktur vor Verletzung bewahren können.

Codd definierte die erste (1NF), zweite (2NF) und dritte Normalform (3NF) [Cod 72]. Eine nichtnormalisierte Relation kann durch einen Zerlegungsprozess über die 1NF in die 2NF und in die 3NF überführt werden: Eine Relation R ist in 1NF, wenn alle ihre Domänen nur atomare Werte enthalten. Eine Relation R ist in 2NF, wenn sie in 1NF ist und wenn jedes nicht zu einem Schlüsselkandidaten gehörende Attribut von jedem Schlüsselkandidaten voll funktional abhängt (also keine funktionale Abhängigkeit von einer Teilmenge eines Schlüsselkandidaten). Eine Relation R ist in 3NF, wenn sie in 2NF ist und jedes nicht zu einem Schlüsselkandidaten gehörende Attribut nichttransitiv von jedem Schlüsselkandidaten abhängt.

Unsere angestrebte Normalform bei der Normalisierung von Relationen ist die Boyce/Codd Normalform (BCNF) [Cod 74], die wir in Anlehnung an die gängige Praxis im weiteren als Relation in 3NF bezeichnen: Eine Relation ist in 3NF genau dann, wenn jede Determinante ein Schlüsselkandidat ist. Dabei ist eine Determinante ein Attribut oder eine Kombination von Attributen, von denen ein anderes Attribut voll funktional abhängt.

Ein kleines Beispiel soll die Verletzung realitätskonformer Sachverhalte im Datenbank-Schema zeigen, wenn die Realität nicht mit Relationen in 3NF modelliert wird:

Wollten wir den Sachverhalt festhalten, in welchem Land ein Mitarbeiter der Einkommens-Steuerpflicht unterliegt (bei Arbeitnehmern mit Grenzgänger-Status resp. bei kantonalen Steuergesetzen kein absurdes Beispiel), so könnten wir dies in einer Relation folgendermassen beschreiben (Primärschlüssel unterstrichen):

Steuerbarkeit (Mitarb.#, Name, Wohnort, Eink.-Steuerpflicht)

Das Modell hält jedoch den Sachverhalt, dass der Wohnort die Einkommens-Steuerpflicht bestimmt, nicht realitätskonform fest. Beispielsweise könnten zwei Tupel der Relation "Steuerbarkeit" folgendermassen aussehen:

Steuerbarkeit	Mitarb.#	Name	Wohnort	Eink.-Steuerpflicht
	1213	Thoma	Basel	Basel-Stadt
	1812	Müller	Basel	Deutschland

Beide Tupel sind somit modell-, jedoch nicht realitätskonform, die Relation "Steuerbarkeit" verstösst gegen 3NF. Realitätskonform wären für diesen Sachverhalt die beiden Relationen

Mitarbeiter (Mitarb.#, Name, Wohnort)
Steuer (Wohnort, Eink.-Steuerpflicht).

3. SCHRITTE BEIM DATENBANK-ENTWURF

3.1 Ueberblick

Einen Ueberblick über die unterschiedlichen Schritte beim Entwurf von Datenbanken, die zu aufeinander abgestimmten Abstraktionsebenen führen, gibt Abb. 1.

Der hier mit "semantischer Entwurf" bezeichnete eine Schritt vom betrieblichen Wirkungsfeld zur Datenbeschreibung wird grossenteils in der Literatur (z.B. in [Cer 83, Ort 85, AJ 84, MDL 85])in zwei unterschiedlichen Abstraktions-Schritten abgehandelt: einem Schritt der Informationsbedarfs-Analyse zur Umschreibung des betrieblichen Wirkungsfeldes und einem Schritt, der zu einem semantischen Modell führt. Ortner [Ort 85] bezeichnet diese Schritte als Anforderungsspezifikationen auf der fachlichen Ebene und als Objekttypenkonstruktion auf der konstruktiven (in unserem Sinne semantischen) Ebene. Wir vollziehen diesen Schritt in gemeinsamer Arbeit von Experten der Fachabteilung und der DV-Abteilung, wobei der Funktionsumfang des geplanten Informationssystems das betriebliche Wirkungsfeld abgrenzt und als Datenbeschreibung ein Entity-Relationship-Modell - ähnlich dem Konzept von Chen [Che 76] - zum Ziel hat.

Im "konzeptionellen Entwurf" wird die Datenbeschreibung in das konzeptionelle Schema - Relationen in 3NF - überführt. Ergänzt um quantitative Angaben der wichtigsten, die entsprechenden Relationen verwendenden Funktionen, bilden sie das sog. "Datenmodell" (nicht zu verwechseln mit den auch Datenmodelle genannten Modellierungskonzepten von Datenbank-Systemen). Diese Abstraktionsebene fügen wir zwischen das semantische Modell und das logische Datenbank-Schema ein, um eine Basis für die Bildung eines "kanonischen Datenmodelles" [Vet 85] zu erhalten. Das Entity-Relationship-Modell der Datenbeschreibung ist wie unseres Erachtens auch andere semantische Modelle in seinem Umfang von der Anforderungsanalyse eines spezifischen Informationssystems geprägt und deshalb wenig geeignet, anwendungsunabhängiges Modell für die Daten eines weltweit tätigen Grossunternehmens resp. einer, zahlreiche Informationsysteme umfassenden organisatorischen Einheit zu sein. Diese integrierende Wirkung erhoffen wir uns jedoch vom Relationenmodell. Diese Einschätzung wird noch verstärkt durch die Tatsache, dass beim benutzernahen, semantischen Entwurf unterschiedliche Experten aus unterschiedlichen Fachabteilungen tätig sind, während die Bildung des "Datenmodells" hauptsächlich von einer zentralen Organisation getragen werden muss.

Die Abbildung der Daten auf die Daten-Definitions-Schnittstelle eines vorhandenen Datenbank-Systems nennen wir hier den logischen Entwurf. Eine wichtige Rolle spielen hierbei jetzt die funktionalen Ergänzungen, die zusammen mit den Relationen des konzeptionellen Schemas im "Datenmodell" festgehalten sind und die wichtige Informationen zur Bildung eines effizienten logischen und physischen Schemas festhalten. Für den Leser, der mit der IMS-Terminologie vertraut ist, sei hier bemerkt, dass unter dem Terminus "logisches Datenbank-Schema" nicht die sog. "logical data bases" des IMS zu verstehen sind, die zur Bildung netzwerkartiger Strukturen definiert werden müssen. Gemeint ist bezüglich des IMS vielmehr die Modellierung des konzeptionellen Schemas mit den Konzepten, die das IMS zur Datendefinition zur Verfügung stellt: die Segmentierung der Daten und ihre hierarchische Verzeigerung mit einer Menge sog. "physical data bases" und mit einer Menge sog. "logical data bases". Die View-Extraktion für die einzelnen Programme erfolgt dann auf der Basis des hier entwickelten logischen Datenbank-Schemas. Der mit physischem Entwurf bezeichnete Schritt zum physischen Schema ist dann bezüglich des IMS die Vervollständigung der das logische Datenbank-Schema bildenden data bases mit der Festschreibung der Zugriffsmethoden, der Verteilung der Daten auf die Datenträger etc.

Es wäre ohne Probleme möglich, direkt aus dem Entity-Relationship-Modell der Datenbeschreibung das logische Datenbank-Schema zu entwickeln und die Abbildung auf Relationen in 3NF lediglich als Sonderfall zu betrachten, wenn nämlich ein relationales Datenbank-System zur Verfügung steht. Das zuvor beschriebene Streben nach einem anwendungsneutralen kanonischen Modell der Unternehmensdaten, das nach einer hinreichenden Anzahl von Anwendungsentwicklungen Ausgangsbasis für die Daten weiterer Anwendungen sein könnte (z.B für die individuelle Informationsverarbeitung) und die Ueberzeugung, dass Relationen in 3NF nicht immer ein effizientes Datenbank-Schema bilden, führen uns zu dem oben beschriebenen "Datenmodell".

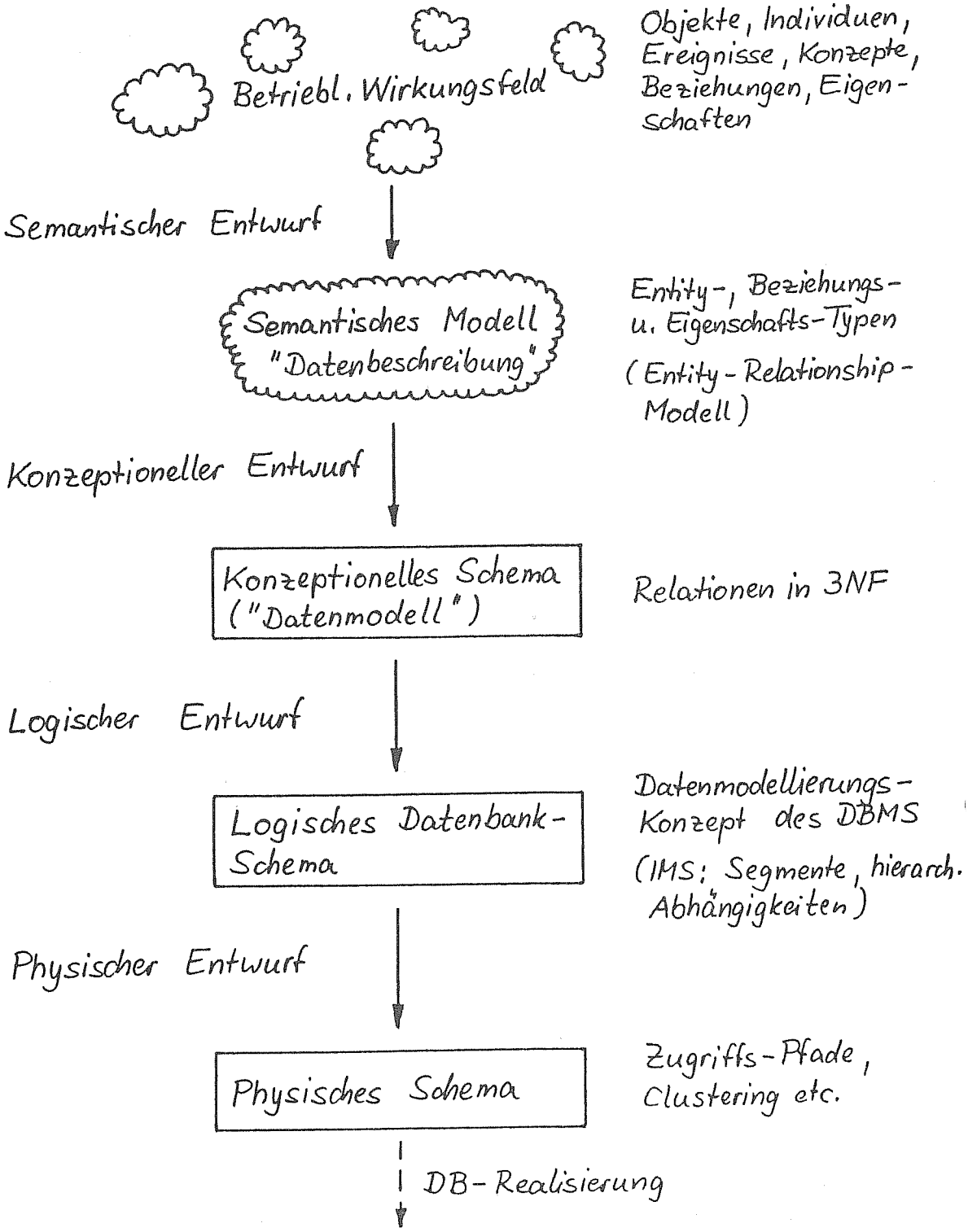


Abb. 1: Schritte des Datenbank-Entwurfs bei Ciba-Geigy

3.2 Semantischer Entwurf

Die Basis für den semantischen Entwurf ist ein wohldefiniertes betriebliches Wirkungsfeld des Informationssystems. Wohldefiniert bedeutet in dem hier beschriebenen Zusammenhang, dass Grenzen zwischen den Objekten, Individuen, Ereignissen oder Konzepten eines Betriebes gezogen werden müssen, je nachdem, ob sie für das betreffende Informationssystem resp. für den zu modellierenden Unternehmensbereich von Bedeutung sind oder nicht. Hierzu werden in [Dav 82] eine Vielzahl unterschiedlicher Strategien und ein Auswahlkriterium vorgeschlagen. Die Wahl der Vorgehensweise wird stark davon beeinflusst, ob für das entsprechende Wirkungsfeld bereits ältere Informationssysteme bestehen, die zu erweitern oder zu ersetzen sind, ob die Modellierung der Daten zum betrachteten Zeitpunkt für genau ein Informationssystem erfolgen soll und damit die Auswahl stark funktionsorientiert ist oder ob die Informationsbasis für einen ausgewählten Bereich eines Unternehmens durch funktionsneutrale Analyse der betrieblichen Realität modelliert werden soll.

In einem Grossteil der Publikatonen wird die Definition des Wirkungsbereiches - wie bereits zuvor erwähnt - als selbständiger Entwurfsschritt beschrieben, der zu einer eigenständigen Abstraktionsebene führt. Dieser Auffassung folgen wir hier nicht. Im Falle der Ergänzung oder Ersetzung von Informationssystemen sowie bei einer funktionsneutralen Analyse der betrieblichen Realität sind diese Grenzen durch bestehende Systeme resp. durch organisatorische Abgrenzungen grösstenteils vorgegeben. Soll die Wahl der Daten für eine Anwendung funktionsorientiert ohne Berücksichtigung vorhandener Systeme erfolgen oder erfolgt die Datenmodellierung zusammen mit Benutzern, denen strukturiertes Denken kaum vertraut ist, so ist eine eigene Abstraktionsebene mit einer Menge informaler Beschreibungen wohl gerechtfertigt. Hierzu wird beispielsweise von De Antonellis und Demo [DAD 83] ein systematischer Weg vorgeschlagen: In 7 Schritten kommt man mit einer Formularhierarchie von einer natürlichsprachlichen Beschreibung von Organisationseinheiten, deren Aufgaben, deren Informationsquellen über eine Einschränkung des Sprachschatzes und eine Satzklassifikation zu Verzeichnissen der Daten, der Operationen und der Ereignisse.

Wir versuchen bei unserer Vorgehensweise, gleichzeitig mit dem Erkennen der Existenz von Einheiten des betriebl. Wirkungsfeldes deren syntaktisch korrekte Aufnahme in das Entity-Relationship-Modell zu vollziehen. Dieser Vorgang erfolgt in Zusammenarbeit von Experten der Fachabteilung, die in der Regel bereits Erfahrung mit der Systemanalyse besitzen, und der Systementwickung. In entscheidenden Phasen wird er von "neutralen" Experten der Modellierungsmethode moderiert.

Objekte, Individuen, Ereignisse oder Konzepte des betrieblichen Wirkungsfeldes, über die wir Informationen sammeln resp. deren Existenz wir festhalten wollen, werden als Entities betrachtet und zu Entity-Typen als Menge von Entities im Sinne der Cantor'schen Mengenlehre zusammengefasst.

Beispiel eines Entity-Typs "Abteilung" (Abb. 2)

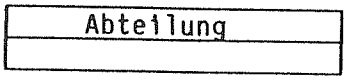


Abb. 2

Ein Entity besitzt mindestens eine Eigenschaft, nämlich diejenige, durch die es innerhalb eines Entity-Types eindeutig identifiziert wird. Der entsprechende Eigenschafts-Typ (Identifikator) wird folgendermassen festgehalten:

Beispiel eines Entity-Typs "Abteilung" mit dem die einzelnen Abteilungen identifizierenden Eigenschafts-Typ "Abt. #" (Abb. 3)

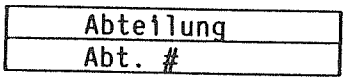


Abb. 3

Jeweils zwischen 2 Entity-Typen können Beziehungen bestehen. Diese Beziehungen bezeichnen wir als einfach oder als komplex, je nachdem, wieviele Entities eines Typs mit wievielen Entities des anderes Typs in Beziehung stehen können:

Beispiel einer Beziehung zwischen den Entity-Typen "Abteilung" und "Mitarbeiter" (Abb. 4)

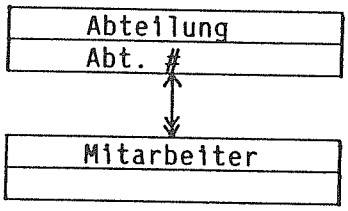


Abb. 4

Interpretation: Eine Abteilung kann mehrere Mitarbeiter beschäftigen, ein Mitarbeiter kann jedoch nur in einer Abteilung beschäftigt sein.

Einer besonderen Modellierung bedürfen beidseitig komplexe Beziehungen zwischen zwei Entity-Typen oder auch bezüglich eines Entity-Types. Diese Beziehungen führen zur Bildung eines Beziehungs-Typs.

Beispiel eines Beziehungs-Typs zwischen den Entity-Typen "Mitarbeiter" und "Projekt" (Abb. 5). Interpretation: Ein Mitarbeiter kann an mehreren Projekten mitwirken, ein Projekt kann mehrere Mitarbeiter beschäftigen.

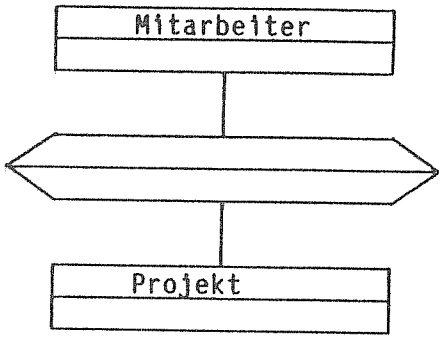


Abb. 5

Beispiel eines Beziehungs-Typs innerhalb des Entity-Typs "Produkt", der mit "Stückliste" bezeichnet wird (Abb. 6). Interpretation: Eine Stückliste, die in der einen "Richtung" der komplexen Beziehung aussagt, dass ein Produkt aus mehreren Komponenten (Zwischenprodukten) bestehen kann und in der anderen "Richtung" dass mehrere Komponenten ein Produkt bilden können.

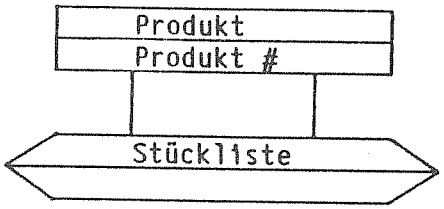


Abb. 6

Diese Beziehungs-Typen erhalten wie die Entity-Typen während des Aufbaus des Entity-Relationship-Modells eine Bezeichnung sowie einen die einzelnen Beziehungen identifizierenden Identifikator: Eine Kombination der Identifikationen der an der Beziehung beteiligten Entity-Typen oder ein Surrogat.

Entities von Entity-Typen und Beziehungen von Beziehungs-Typen besitzen also mindestens eine Eigenschaft resp. eine Kombination von Eigenschaften: den Identifikator, mit dessen Hilfe sie von den anderen Exemplaren desselben Typs unterschieden werden können. Sowohl Entities als auch Beziehungen können jedoch sehr viele weitere Eigenschaften aufweisen. Gleichartige Eigenschaften werden zu Eigenschafts-Typen zusammengefasst. Eigenschafts-Typen können nur mittelbar oder unmittelbar im Zusammenhang mit mindestens einem Entity- oder Beziehungs-Typ existieren. Diese Abhängigkeit (einfach oder komplex) wird in Form von Pfeilen zwischen Eigenschafts-Typ und Entity-Typ, Beziehungs-Typ oder anderen Eigenschafts-Typen dargestellt.

Beispiel für Eigenschafts-Typen "Name", "Wohnort" und "Eink.-Steuerpflicht" und deren Abhängigkeiten (Abb. 7). Interpretation: Ein Mitarbeiter besitzt nur einen Namen, ist nur an einem Ort mit Hauptwohnsitz gemeldet, umgekehrt können jedoch mehrere Mitarbeiter denselben Namen tragen resp. denselben Hauptwohnsitz haben. Der Kanton oder das Land, bei dem Einkommens-Steuerpflicht besteht, wird vom Wohnort bestimmt (z.B. in Basel-Stadt, wenn der Wohnort CH-4000 Basel ist, in Deutschland, wenn in D-7850 Lörrach wohnhaft, in Basel-Stadt Quellensteuer, wenn als Grenzgänger in D-7800 Freiburg wohnhaft etc.).

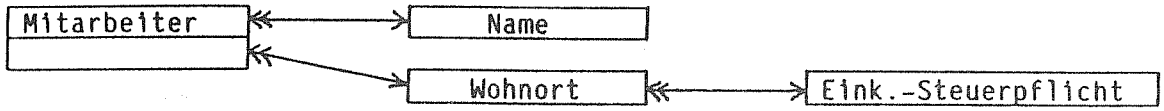


Abb. 7

Eigenschafts-Typen stehen in einem Kontext zu ihren Entity- resp. Beziehungs-Typen. Wenn aus dem (implizit vorhanden) Kontext und der Bezeichnung die Bedeutung des Eigenschafts-Typs zweifelsfrei hervorgeht, bezeichnen wir den Abhängigkeitspfeil nicht, wie sonst im Entity-Relationship-Modell üblich, mit einer Charakteristik: Die Bezeichnung des Eigenschafts-Typs ersetzt diejenige der Charakteristik.

Die Angabe einer Charakteristik ist dann notwendig, wenn unterschiedliche Abhängigkeiten desselben Eigenschafts-Typs festgehalten werden müssen. Ausserdem müssen wir in bestimmten Fällen eine Rollenindikation zu einer Beziehung vornehmen, beispielsweise bei der Stückliste (Abb. 8).

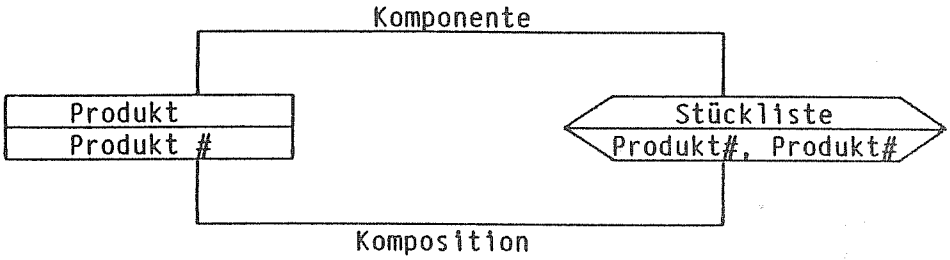


Abb. 8

Aus Gründen der Uebersichtlichkeit unseres Modells müssen wir so viel als zum Verständnis nötig und so wenig wie möglich in der Graphik unterbringen. Da bei einzelnen Anwendungen bis zu 200 Eigenschafts-Typen für einen Entity- oder Beziehungs-Typ anfallen können, müssen wir gegebenenfalls Eigenschafts-Typen mit gleichartigen Abhängigkeiten in einem separaten Formular zusammenfassen und in der Datenbeschreibung lediglich einen Verweis auf dieses Formular eintragen.

Die Entwicklung unseres Entity-Relationship-Modells der Datenbeschreibung vollzieht sich in mehreren Phasen und Iterationen. Zuerst gilt unsere Aufmerksamkeit den Entity- und Beziehungs-Typen. Ihr Netz bildet die Basis-Version des Entity-Relationship-Modells, das von einem bzgl. der Systementwicklung unabhängigen Moderator in der Diskussion mit Benutzern und Systementwicklern verifiziert und gegebenenfalls korrigiert wird. Wo möglich und wo zum Verständnis notwendig, werden die Identifikatoren bereits jetzt definiert. In der zweiten Phase ermitteln wir die Eigenschafts-Typen und - wo notwendig - die Charakteristika resp. die Rollen. Die Bereinigung der Bezeichnungen der Eigenschafts-Typen (Eindeutigkeit) sowie die definitive Festsetzung der Identifikatoren erfolgt jetzt. Identifikatoren können Eigenschafts-Typen sein, die a priori nur für diesen Zweck definiert wurden oder die jetzt definiert werden müssen oder es können Eigenschafts-Typen sein, die in beidseitig einfacher Abhängigkeit zum Entity-Typ stehen.

Als Identifikator eines Beziehungs-Typs ist zweckmässigerweise die Kombination der Identifikatoren der an der Beziehung beteiligten Entity-Typen einzusetzen.

Noch eine Bemerkung zu den Beziehungs-Typen: Der eine oder andere Analytiker wird anstelle einer beidseitig komplexen Beziehung zwischen zwei Entity-Typen einen weiteren Entity-Typ definieren mit zwei korrekten (1:n)-Beziehungen. Als Beispiel (Abb. 9.) sei hier die "Projekt-Mitwirkung" von Mitarbeitern an Projekten aus Abb. 5 aufgeführt. Im Prinzip modellieren Abb. 5 und Abb. 9 denselben Sachverhalt bei korrekter Wahl der Identifikatoren, der bei Abb. 5 beschrieben ist.

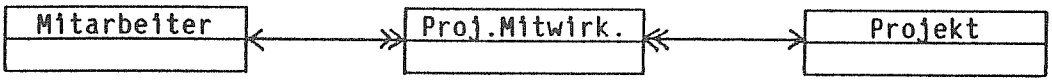


Abb. 9

Beide Modell-Varianten führen beim konzeptionellen Entwurf auch zu derselben Relationen-Menge in 3NF. Was jedoch bei der Variante von Abb. 9 auf den ersten Blick verlorengegangen ist, ist der abhängige Charakter der "Projekt-Mitwirkung" (referential integrity). Mit der Modellierung eines Beziehungs-Typs wird sowohl den Integritäts-Belangen als auch dem Gesichtspunkt der eigenständigen Existenz beim späteren logischen Entwurf mehr Beachtung geschenkt (z.B. Möglichkeit, Wurzelsegment in einer IMS-Struktur zu werden).

3.3 Konzeptioneller Entwurf

Die graphische Beschreibung der Daten des betrieblichen Wirkungsfeldes als Entity-Relationship-Modell in der Datenbeschreibung bildet die Basis, um zum konzeptionellen Schema, dem bei uns sog. "Datenmodell" zu gelangen. Hauptbestandteil dieses "Datenmodells" sind - wie bereits erwähnt - Relationen in 3NF. Aus den Konstruktionselementen der Datenbeschreibung (Entity-, Beziehungs- und Eigenschafts-Typ) müssen hierbei Relationen und Attribute definiert werden. Der Weg hierzu führt nicht über grosse Relationen und einen Zerlegungsprozess in 1NF, 2NF und 3NF, sondern direkt vom Entity-Relationship-Modell mittels geeigneter Transformationsvorschriften zu Relationen in 3NF (Relationen-Synthese):

1. Jeder Entity-Typ und jeder Beziehungs-Typ definiert eine eigene Relation mit dem Identifikator als Primärschlüssel.

2. Den Relationen werden diejenigen Eigenschafts-Typen als Attribute hinzugefügt, die vom entsprechenden Entity- resp. Beziehungs-Typ aus unmittelbar mit einer einfachen Beziehung (Aufnahme des Identifikators eines "benachbarten" Entity-Typs als Fremdschlüssel) oder einer einfachen Abhängigkeit erreichbar sind. Ist diese Abhängigkeit beidseitig einfach, handelt es sich um einen Schlüsselkandidaten. Gegebenenfalls muss die Bezeichnung eines Attributs aus der Charakteristik resp. der Rolle und der Bezeichnung des Eigenschafts-Typs zusammengesetzt werden.

3. Definition weiterer Relationen, wenn Eigenschafts-Typen weitere Eigenschafts-Typen in einfacher Abhängigkeit besitzen (Vermeidung transitiver Abhängigkeiten); die Bezeichnung des Eigenschafts-Typs, von dem die einfachen Abhängigkeiten ausgehen, wird Primärschlüssel.

4. Definition weiterer Relationen bei der Existenz beidseitig komplexer Abhängigkeiten; jeder beteiligte Eigenschafts-Typ führt zu einer Relation (Primärschlüssel wird Bezeichnung des Eigenschafts-Typs), jede beidseitig komplexe Abhängigkeit führt ebenfalls zu einer Relation (Primärschlüssel wird fallweise eine Kombination aus den Bezeichnungen der beiden Eigenschafts-Typen oder aus der Bezeichnung des Eigenschafts-Typs und dem Identifikator des Entity-Typs resp. des Beziehungs-Typs).

5. Definition von Relationen aus Eigenschafts-Typen, die Entity-Typen resp. Beziehungs-Typen in einfacher Abhängigkeit besitzen, selbst jedoch komplex von diesen abhängen; die Bezeichnung des Eigenschafts-Typs wird Primärschlüssel.

6. Zuordnung der Eigenschafts-Typen als Attribute zu den nach Regeln 3 bis 5 gebildeten Relationen sinngemäss nach Regel 2.

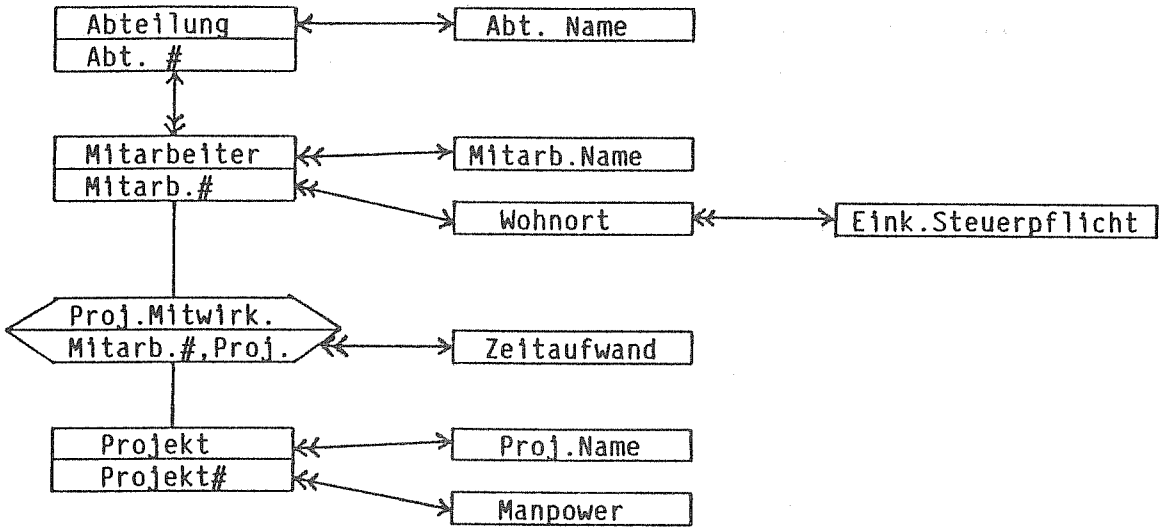
Normalerweise wird man mit der Durchführung der ersten beiden Transformationsregeln, in Spezialfällen noch mit Regel 3 auskommen. Die restlichen Regeln werden in der Hauptsache lediglich dann angewendet werden müssen, wenn Entity- resp. Beziehungs-Typen nicht als solche erkannt und fälschlicherweise als Eigenschafts-Typ in die Datenbeschreibung eingetragen wurden.

Aus der Menge der so gebildeten Relationen muss man unter Umständen noch Relationen streichen, die in anderen bereits enthalten sind. Dies kann der Fall sein bei Relationen aus Beziehungs-Typen, die mit anderen Beziehungs-Typen komplex in Beziehung stehen und selbst keine anderen Attribute als den Primärschlüssel besitzen. Oder es müssen in Relationen mehrfach vorhandene Attribute gestrichen werden (z.B. wenn anstelle eines Beziehungs-Typs ein Entity-Typ mit kombinierten Identifikatoren definiert wurde).

Um ein kanonisches Datenmodell zu erhalten, sind die jetzt erstellten Relationen gegebenenfalls in ein bereits vorhandenes Schema aus Relationen in 3NF zu integrieren. Hierbei gilt es, Redundanzen durch Synonyma resp. durch Relationen, die bereits in anderen Relationen enthalten sind, zu vermeiden. Es gilt aber auch, Homonyme zu erkennen und dann neue und eindeutige Bezeichnungen einzuführen.

Den Transformationsvorgang wollen wir jetzt anhand zweier Beispiele verdeutlichen. Ausgangspunkt ist jeweils ein Entity-Relationship-Modell, das in Teilen bereits besprochene Sachverhalte modelliert. Es werden deshalb für beide Beispiele keine zusätzlichen Erklärungen mehr gegeben, die Transformationen in die Relationen können anhand obiger Regeln leicht nachvollzogen werden.

Beispiel 1: Sachverhalt über Mitarbeiter, Abteilungen, Projekte und Projekt-Mitwirkungszeit (Abb. 10):



Abteilung (Abt. #, Abt. Name)
Mitarbeiter (Mitarb.#, Abt. #, Mitarb.Name, Wohnort)
Projekt-Mitwirkung (Mitarb.#, Projekt#, Zeitaufwand)
Projekt (Projekt#, Proj.Name, Manpower)
Steuerbarkeit (Wohnort, Eink.-Steuerpflicht)

Abb. 10

Beispiel 2: Stückliste (Abb. 11):

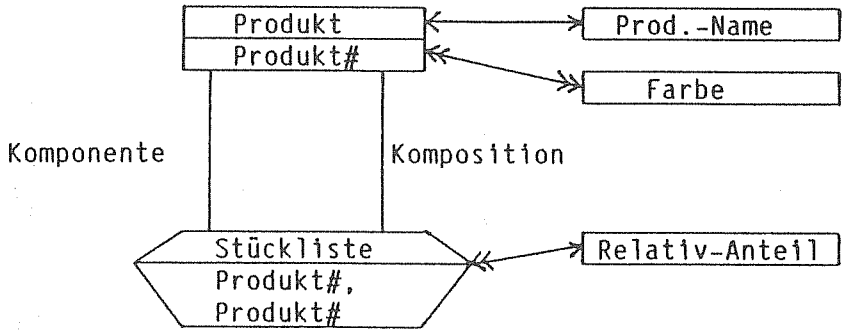


Abb. 11

Produkt (Produkt#, Prod.-Name)
Stückliste (Komponente.Produkt#, Komposition.Produkt#, Relativ-Anteil)
Farbskala (Farbe)
Einfärbung (Produkt#, Farbe)

Dieses Beispiel verdeutlicht die obige Bemerkung, dass Entity-Typen fälschlicherweise als Eigenschafts-Typen modelliert werden können. Die Transformationsregeln bügeln den Schaden zwar wieder aus. Beispiel 2 (Stückliste) hätte mit Abb. 12 als korrektem Entity-Relationsship-Modell zum selben Relationen-Schema geführt wie Abb. 11.

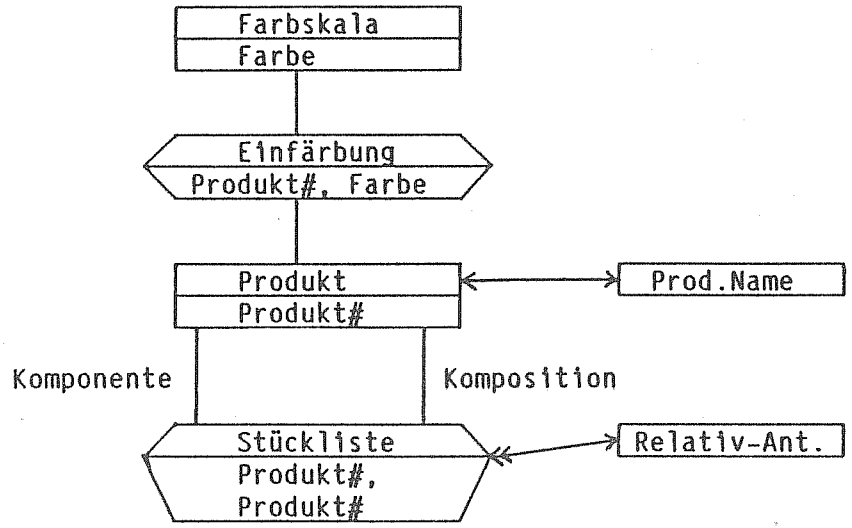


Abb. 12

3.4 Logischer Entwurf

Mit diesem Schritt wird die Abbildung der Realität auf die vom Datenbank-Management-System angebotene Daten-Definition-Schnittstelle vorgenommen. Bei uns ist dies hauptsächlich die Schnittstelle von IMS, in Ausnahmefällen diejenige von TOTAL.

Basis dieses Schrittes ist die im Datenmodell festgehaltene konzeptionelle Struktur der Daten in Form von Relationen. Die Zusammenfassung der Attribute zu Relationen erfolgte ausschliesslich unter dem Gesichtspunkt funktionaler Abhängigkeiten. Auf dieser Grundlage können nun die wichtigsten Schritte dieser Abbildung, nämlich

- Definition von Datenfeldern aus Attributen
- Bildung von Segmenten aus Relationen
- Bildung der DB-Struktur aus den Beziehungen zwischen den Relationen via Primär- und Fremdschlüssel

durchgeführt werden. Die Methode der Umsetzung des Datenmodells in ein Datenbank-Schema unter Beibehaltung der logischen Bedingungen ist in [Vet 85] beschrieben. Die logischen Abhängigkeiten der Realität (im Datenmodell dargestellt) sind jedoch nur ein Gesichtspunkt für den Schema-Entwurf. Weitere Gesichtspunkte sind die Anforderungen an die Performance aus den Funktionen, die Daten aus einer Datenbank benützen und deren Anforderungen kritisch sind sowie die Notwendigkeit eigenständiger Handhabung der Relationen.

Aus diesem Grund werden vor der Abbildung auf das Datenbank-Schema quantitative Angaben zu den Attributen und Relationen für ausgesuchte Funktionen des Informationssystems beschrieben wie

- Typ, Länge und Wertebereich der Domänen
- Umfang der Tupel in einer Relation
- Zugriffe (Häufigkeit, Art) zu Attributwerten
- Umfang von miteinander in Beziehung stehenden Tupeln verschiedener Relationen
- Zugriffe (Art, Häufigkeit, Sequenz) auf Relationen.

Diese Angaben helfen mit, eine optimale Datenbank-Struktur unter Beachtung logischer und applikatorischer Anforderungen zu erreichen. Der Optimalitäts-Kontrolle dienen nach der Segmentierung quantitative Angaben für die Datenfelder und Segmente, wie sie zuvor für die Domänen resp. Attribute und die Relationen zusammengestellt wurden.

Der logische Entwurf wird im wesentlichen in zwei Schritten durchgeführt. Die Abhängigkeiten zwischen den Relationen werden in einem "Relationsgraphen" festgehalten (Relationen als Knoten, Primär-Fremdschlüsselbeziehung als Kanten), im ersten Schritt 1:1 auf das Schema des DBMS abgebildet und anschliessend mit Hilfe der quantitativen Angaben im obigen Sinne modifiziert.

3.5 Physischer Entwurf

In diesem letzten Entwurfs-Schritt werden die Parameter für die physische Speicherung und Verwaltung der Datenbanken festgelegt. Dies bedeutet, dass unter Beachtung applikatorischer Notwendigkeiten der Datenbank-Typ, die Zugriffsmethode, der benötigte Space, Pointers, Blockgrössen etc. ausgewählt werden müssen.

4. ANMERKUNGEN ZUM PRAKTISCHEN EINSATZ

4.1 Terminologie

Wissenschaftliche Fachsprachen erzeugen ausserhalb wissenschaftlicher Arbeitsbereiche Schwellenängste und haben für denjenigen "Praktiker", der sich gerne von den sog. "Theoretikern" abhebt, einen Anstrich von nicht praktizierbarer Theorie. Deshalb hat der Verfasser in [Tho 84] noch nicht dem Wunsch von Praktikern widersprochen, mit den Begriffen Entity, Beziehungen, Datenelement und der Normalform eines Entity sowohl den semantischen als auch den konzeptionellen Entwurf beschreiben zu wollen. Diese exemplarspezifischen Begriffe sollten also auch für Mengen sowie für die Elemente der Relationentheorie benutzt werden. Mit dieser Begriffsarmut wird der Praktiker nach den bisherigen Erfahrungen jedoch

mehr verwirrt als geschont, und demjenigen, der ohnehin mit abstraktem oder strukturellem Denken Mühe hat, wird der Unterschied zwischen Exemplaren und Mengen resp. zwischen Datenbeschreibung und Datenmodell nicht einmal mit unterschiedlichen Sprachkonstrukten offenbart. Auch der Praktiker wird deshalb inskünftig mit den wenigen Begriffen des Entity-Relationship-Modells, wie sie in Kap. 3 eingeführt wurden und mit den Grundbegriffen der Relationentheorie vertraut gemacht. Das Vokabular ist trotzdem klein gehalten - nicht zuletzt durch die Wahl der Methode - und verlangt keine Kenntnisse von unterschiedlichen Normalformen, von funktionaler Abhängigkeit etc. Es sollte jedoch auch vom Praktiker die Bereitschaft erwartet werden, sich beim Erlernen einer neuen Arbeitsmethode auch mit einigen wenigen neuen Begriffen vertraut zu machen.

4.2 Methoden-Einführung

Dass zwischen der ersten, groben Niederschrift von neuen Entwicklungs-Standards für Datenbanken und der Einführung einer adäquaten Methode ca. 2 Jahre vergangen sind, mag auf den ersten Blick erstaunen. Für diese Verzögerung sind jedoch mehrere Faktoren verantwortlich: Die betroffenen Mitarbeiter sind in zeitkritische Projekte eingebunden und deshalb nicht sofort frei für die detaillierte Ausarbeitung und Schulung resp. zum Erlernen einer neuen Methode. Ferner müssen, selbst wenn das obere Management von der Notwendigkeit einer konzeptionellen Design-Methodik und zentral geführter und integrierter Datenbanken überzeugt ist, auch die Mitglieder der übrigen Managementstufen und die Sachbearbeiter bereit sein, sich vom Konzept programmspezifischer Files zur Datenspeicherung zu lösen, wo dies noch nicht erfolgt ist. Dies verlangt Ueberzeugungskraft und einen Erkenntnisprozess und kostet ebenfalls Zeit. Der Erkenntnisprozess wird sicherlich durch die Probleme beschleunigt, die als Folge fehlender Datenintegration und fehlender Dokumentation der betrieblichen Bedeutung der Daten bei zunehmender individueller Informationsverarbeitung in den Fachabteilungen auftreten. Der Endbenutzer in der Fachabteilung kann nämlich seine Informationsbedürfnisse nur dann mit vollem Nutzen befriedigen, wenn er einen unkomplizierten Zugang zu den zentralen Daten realisieren kann und wenn er diese Daten korrekt interpretieren kann (Datenzentrum). Semantische und konzeptionelle Entwicklungsschritte sind hierzu unerlässlich.

Die hier vorgestellte Methode wird momentan erstmals bei der Neuentwicklung eines grossen und bedeutenden Applikations-Systems im Hause eingesetzt und gleichzeitig in Kursen gelehrt. Eine intensive Schulung und die Schaffung einer kompetenten Stelle für das Datenmanagement sowie deren Einbindung in die Ablaufsorganisation der Systementwicklung sind zwar notwendig für die Etablierung eines konzeptionellen Datenbank-Entwurfs, hinreichend hierfür jedoch ist ausschliesslich eine weitverbreitete Akzeptanz der Methode, die die oben beschriebenen Erkenntnisse sowie Erfolge bei ihrem Einsatz in einem anerkannten betrieblichen Wirkungsfeld voraussetzt.

4.3 Methoden-Wahl

Für den konzeptionellen Datenbank-Entwurf bieten sich viele Methoden an (vgl. z.B. [MDL 85]). Die hier vorgestellte Methode führt zu einem konzeptionellen Modell, das sich unseres Erachtens durch die folgenden Eigenschaften auszeichnet:

- Einfach: Wenige aussagekräftige Darstellungselemente und Begriffe sind zur Modellierung notwendig.

- Vollständig: Trotz der einfachen Darstellungsweise sind uns bis jetzt kaum Aspekte bekannt, die wir nicht direkt im Modell beschreiben könnten, sondern die wir in ergänzenden Beschreibungen unterbringen müssen.

- Verständlich: Die graphische Darstellung des Entity-Relationship-Modells bietet ein gutes Arbeitsmittel, um mit den Benutzern die Realität des betrieblichen Wirkungsfeldes diskutieren zu können und bietet somit die Chance, ein korrektes Modell zu erreichen.

- Eindeutig: Sowohl die Konstruktionselemente des Entity-Relationship-Modells als auch das Relationenmodell lassen eine zweifelsfreie Interpretation für die Weiterverarbeitung zu.

- Dokumentationsfähig: Die wichtigsten Konstruktionselemente des Modells sollten zusammen mit ihren Verbindungen zu den Elementen des logischen Schemas und zu den Funktionen computergestützt dokumentierbar sein. Dies ist bei uns momentan nicht möglich, jedoch realisierbar (z.B. im Rahmen unseres Data Dictionary Systems).

- Integrationsfähig: Das Modell lässt die Möglichkeit der Integration konzeptioneller Modelle unterschiedlicher Applikations-Systeme auf der Ebene der Relationen in 3NF zu einem kanonischen Datenmodell zu. Ebenso ist es für die nachträgliche Modellierung und Integration bereits bestehender Datenbanken offen.

Die Methode selbst sollte unseres Erachtens computergestützt realisierbar sein. Sie ist es zwar, jedoch sind uns momentan nur Entwicklungen von Forschungsinstitutionen und keine kommerzielle Realisierung bekannt. Problem bei der Entwicklung eines für den Markt verfügbaren Entwurf-Tools auf der Basis Entity-Relationship-Modell könnte die nur aufwendig realisierbare Portabilität der Graphik sein. Und gerade die Graphik als Kommunikationsbasis zwischen Benutzer und Analytiker ist die Stärke des hier vorgestellten Ansatzes. Wir könnten neben der Evaluation der bekannten Ansätze auch an die eigene Entwicklung eines entsprechenden Tools denken, eventuell auf einem Personal Computer. Hierbei wäre der grösste Aufwand bei der Implementierung der Graphik des Entity-Relationship-Modells zu leisten, während dessen Umsetzung in 3NF-Relationen und in eine logische IMS-Struktur problemlos wäre.

Die uns bekannten kommerziellen Tools stellen die Normalisierung von Relationen in den Mittelpunkt. Normalisierte Relationen können jedoch nicht mehr Diskussionsgrundlage mit Benutzern sein. Wir befürchten deshalb beim Einsatz solcher Tools Einbussen in der Verständlichkeit des Modells mit entsprechenden Auswirkungen auf die Korrektheit, zumal die Gestaltung des Output nur über strukturierte Texte, nicht jedoch über eine Graphik verfügt.

4.4 Detaillierungsgrad der Analyse

Für die Korrektheit und die Lesbarkeit eines semantischen Modelles ist der Detaillierungsgrad der Datenanalyse von ganz entscheidender Bedeutung. Unter Detaillierungsgrad soll hier die Auflösung des betrieblichen Wirkungsfeldes in einzelne Entities verstanden werden. Beispielsweise kann die Adresse eines Mitarbeiters unterschiedlichen Betrachtungsweisen unterzogen werden: Zum einen als Adresse, die ausschliesslich in ihrer Gesamtheit benötigt wird und in sich zu keinem Zeitpunkt und in keiner Anwendung untergliedert wird geschweige irgendwelchen Plausibiliätsprüfungen unterworfen wird. Zum anderen kann eine Adresse selbst in ihre Komponenten Postleitzahl, Ort, Strasse und Hausnummer aufgelöst werden mit der Bedingung, nur gültige Postleitzahlen, Orte, Strassen und Hausnummern in den jeweils gültigen Kombinationen zu verwenden. Im ersten Fall modellieren wir die Adresse als Eigenschaftstyp, im zweiten Fall als Netz zwischen Entity-, Beziehungs- und Eigenschafts-Typen. Wir müssen also darauf achten, dass wir ein betriebliches Wirkungsfeld nur bis zu der von uns benötigten Feinheit auflösen, um die Lesbarkeit des Modells nicht zu gefährden.

Andererseits dürfen wir der Lesbarkeit nicht die Korrektheit eines Modelles opfern. Als Beispiel führe ich hier die Rezeptur von chemischen Produkten an. Dies ist in der einfachsten Betrachtungsweise eine Stückliste über einem Entity-Typ "Produkt". Für ein bestimmtes chemisches Produkt gibt es in einer chemischen Fertigung jedoch nicht nur ein Rezept: Abhängig von der Fertigungsanlage mit unterschiedlichen Ausbeute-Anteilen oder von der Qualität der Komponenten existieren für die Fertigung unterschiedliche Rezepte, die nicht voneinander ableitbar sind. Vereinfachend können wir von einer Basis- und einer Fabrikations-Rezeptur sprechen. Für die Kalkulation der Produkte ist eine weitere Stückliste massgebend, die zwar in bestimmten Perioden mit der Fabrikations-Rezeptur abgeglichen wird, deren Unterschiedlichkeit zwischen den Abgleichszeitpunkten jedoch durchaus sinnvoll ist. Das Fabrikations-Rezept wird durch chemische und technische Indikatoren beeinflusst, während der Takt für die Abstimmung der Kalkulation mit der Fertigung auch von kaufmännischen Gesichtspunkten bestimmt wird. Wir haben also für ein auf den ersten Blick so einfaches Gebilde wie eine Produkte-Stückliste schon drei semantische Ausprägungen; mit einer hätten wir zwar die Lesbarkeit des Modelles erhöht, jedoch zu Lasten der Korrektheit.

Welcher Detaillierungsgrad nun der anzustrebende ist, kann nicht a priori bestimmt werden. Er muss zusammen mit dem Benutzer festgelegt werden und wird dadurch bestimmt, mit welchen Details die Benutzer das betriebliche Wirkungsfeld zur Erfüllung ihrer Aufgaben sehen müssen.

5. SCHLUSSBETRACHTUNG

Das konzeptionelle Modell als Basis für die Wandlung des Rechenzentrums zum Datenzentrum: Ein Ziel, das auf verschiedenen Wegen erreicht werden kann. Der hier vorliegende Ansatz beschreibt einen Weg, der unter der Prämisse ausgewählt wurde, dass die systematische und verständliche Modellierung der Realität des betrieblichen Wirkungsfeldes problematischer ist als die algorithmisierbare Normalisierung von Relationen. Zusätzlich verbindet er die Vorteile applikationsspezifischer Modellierung mit der Notwendigkeit applikationsneutraler Integration zu einem kanonischen Modell und ist von der Informationsbedarfsanalyse bis zu einem ersten Entwurf des systemspezifischen logischen Datenbank-Schemas computerisierbar. Die Auswahl eines anderen Weges wäre denkbar, indem z.B. ein kommerziell erhältliches Tool ausgewählt und die diesem zugrundeliegende Methode eingeführt wird. Wie bereits früher erwähnt, läge nach unserem momentanen Kenntnisstand der Schwerpunkt dann allerdings auf der Synthese von "Elementar-Relationen" zu Relationen in 3NF.

Die Diskussionen über den besseren Weg sind auch in unserem Hause noch nicht abgeschlossen. Ungeschickt wäre jedoch, wenn - wie in der Datenverarbeitung nicht selten - über Tool- und Methodenstreit das Ziel aus den Augen verloren ginge. Dies zu verhindern sowie die Etablierung einer Methode jetzt und für alle sollte unser aller Bestreben sein. In diesem Sinne soll der vorliegende Bericht sowohl den momentan eingeschlagenen Weg beschreiben als auch einen Beitrag zur Methoden-Diskussion liefern.

6. LITERATUR

AJ 84 Agosti, M.; Johnson, R.G.: A Framework of Reference for Database Design. DATA BASE (SIGBDP), 15(4), 1984, 3 - 9.

Cer 83 Ceri, S. (Ed.): Methodology and Tools for Data Base Design. North-Holland, 1983.

Che 76 Chen, P.P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), 1976, 9 - 36.

Cod 72 Codd, E.F.: Further Normalization of the Data Base Relational Model. In: Rustin, R. (Hrsg.): Data Base Systems, Courant Computer Science Symposium 6, May 1971. Prentice Hall, 1972, 33 - 64.

Cod 74 Codd, E.F.: Recent Investigations into Relational Data Base Systems. Proc. IFIP Congress, 1974, 1017 - 1021.

DAD 83 De Antonellis, V.; Demo, B.: Requirements Collection and Analysis. In: [Cer 83], 9 - 24.

Dat 81 Date, C.J.: An Introduction to Database Systems. Addision-Wesley (3. Auflage), 1981.

Dav 82 Davis, G.B.: Strategies for Information Requirements Determination. IBM Systems Journal, 21 (1), 1982, 4 - 30.

MDL 85 Mayr, H.C.; Dittrich, K.; Lockemann, P.C.: Datenbankentwurf. Kapitel 5 von Schmidt, J.; Lockemann, P.C. (Hrsg.): Datenbankhandbuch. Erscheint bei Springer, vorauss. 1986.

Ort 85 Ortner, E.: Semantische Modellierung - Datenbankentwurf auf der Ebene der Benutzer. Informatik-Spektrum, 8(1), 1985, 20 - 28.

Tho 84 Thoma, H.: Datenbank-Design mit Relationen. In: 16. Deutsche G.U.I.D.E Tagung 1984 (Tagungsband). 1984, 218 - 223.

Vet 85 Vetter, M.: Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung. Teubner (2. Auflage), 1985.