Der praktische Erfolg einer Theorie: das relationale Modell bei den Datenbanken

von Holger Maaß, maass@maasster.de, 06.06.2010

Mein beruflicher Weg der letzten Jahre hat mich dahin geführt, den Hauptteil meiner täglichen Arbeit mit Datenbankentwicklung im medizinischen Bereich zu verbringen. Nach und nach habe ich mir hier einen Erfahrungsschatz aufbauen können und eine gewisse praktische Expertise erreicht. Dabei habe ich immer mehr den Eindruck gewonnen, dass die heute gängigen relationalen Datenbanksysteme von der Sache her einer sinnvollen Idee folgen und sich für die Erfordernisse der Praxis sehr gut eignen. Dazu kam die Erfahrung, dass die relationalen Systeme das Feld fast zu 100% beherrschen und dass eine Ablösung durch etwas anderes nicht in Sicht ist. Für mich stellte sich daher ganz natürlicherweise die Frage, warum das eigentlich so ist.

Eine mögliche Antwort könnte sein, dass es im Grunde ein Zufall ist, der zur marktbeherrschenden Stellung der relationalen Systeme geführt hat und nun neuere Entwicklungen und Alternativen abblockt. Dies scheint mir jedoch unwahrscheinlich. Zwar ist es in gewisser Weise Zufall, dass gerade Oracle das erste kommerzielle relationale Datenbanksystem entwickelt hat, das bis heute große Teile des Marktes beherrscht. Aber dass die relationalen Systeme den Markt überhaupt in Besitz nehmen konnten, muss nach meiner Auffassung in der Grundidee des relationalen Modells begründet liegen. So kam ich dazu, nach einem tieferen Verständnis des relationalen Modells zu suchen und bin dabei auf zwei Artikel von Edgar F. Codd gestoßen, der das relationale Modell für Datenbanken eingeführt hat:

A) A Relational Model of Data for Large Shared Data Banks (1970)

http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

B) Relational Database: A Practical Foundation for Productivity. Turing Award Lecture (1981)

http://awards.acm.org/images/awards/140/articles/2485527.pdf

Das erste Paper gilt gemeinhin als das Gründungsdokument für das relationale Modell und enthält wichtige Hinweise auf die Motivation zur Einführung dieses Modells. Das zweite Paper wurde jedoch 11 Jahre später veröffentlicht und enthält in größerer Deutlichkeit Hinweise darauf, warum gerade das relationale Modell den praktischen Anforderungen an Datenbanken in idealer Weise gerecht wird.

Beide Papers zeugen aus meiner Sicht von einem großen Weitblick Codds, der als studierter Mathematiker viel Sinn für theoretische Fundierung mitbringt und diesen aber mit einem erstaunlichen Gespür für die Erfordernisse der industriellen Praxis vereint. Beides zusammen hat m.E. zur Idee des relationalen Modells geführt.

1.) Die Relation als logische Grundstruktur für die Anordnung von Daten

Jedes System, das Daten in strukturierter Form vorhält und zur Verfügung stellt, also auch jedes Datenbanksystem, muss für die Anordnung der Daten eine elementare Grundform festlegen, wie die Daten vorgehalten werden sollen. Dabei zieht Codd eine scharfe Trennlinie zwischen zwei Schichten des Systems. Die eine Schicht ist für ihn die physische Speicherung der Daten samt der komplexen technischen Mittel, die dazu nötig sind. Die andere Schicht ist die logische Sicht auf die Daten, die ganz unabhängig von der physischen Speicherung sein soll und die für Codd absoluten Vorrang hat.

Für Codd hat also die logische Schicht der Datenanordnung einen Vorrang im Aufbau des Datenbanksystems, und die physische Schicht der Speicherung muss so aufgebaut werden, dass sie das Gerüst stellt, auf der die logische Schicht aufsitzt. Ziel dieser konsequenten Trennung ist es, die physische Schicht komplett vor dem Benutzer der Datenbank zu verbergen, so dass dieser keine Mühe darauf verwenden muss, sich mit dieser Seite der Daten zu befassen. Der Benutzer - und natürlich auch der Anwendungsentwickler - sollen sich voll auf die logische Seite der Daten konzentrieren können, was einen großen Produktivitätsgewinn für die beteiligten Personen bedeutet.

Codd geht nun also daran, seine Idee eines Datenbanksystems rein auf der logischen Ebene zu entwerfen und überlässt es anderen, geeignete Software zur technischen Realisierung des so definierten Datensystems zu entwickeln. Dass dies keine einfache Aufgabe ist, würde Codd nie bestreiten, aber er würde immer betonen, dass der technische Aufbau im Dienste der logischen Struktur stehen muss und keinen Einfluss auf die Festlegung der logischen Schicht haben darf. Damit ist schon ein grundlegender Aspekt des relationalen Datenbankmodells benannt: der konsequente Primat der logischen Sicht auf die Daten.

Wie aber will nun Codd die Daten logisch anordnen? Was soll die Grundstruktur des Datenmodells sein und was die elementare Einheit dieser Struktur?

Bekanntlich hat sich Codd für die Relation als Grundelement der Datenstrukturierung entschieden, was ja auch den relationalen Datenbanksystemen ihren Namen gegeben hat. Was aber eine Relation eigentlich ist, dürfte vielen alles andere als klar sein. Um das besser zu verstehen, kann man sich zunächst einmal an die verbreitetste Veranschaulichung und Realisierung einer Relation halten, und das ist eine Tabelle. In erster Annäherung kann man also sagen: eine Relation ist eine Tabelle. Sie besteht aus Spalten und Zeilen, wobei die Spalten Namen haben und in den Zeilen den Spalten Werte zugeordnet werden. Hier ein Beispiel:

Name	Vorname	Wohnort
Schmidt	Erich	Hamburg
Dubois	Jean	Paris
Rossi	Adriana	Florenz

Diese Tabelle hat drei Spalten (Name, Vorname, Wohnort) und drei Zeilen. Dass sie gewissermaßen quadratisch ist, ist reiner Zufall. Es könnten auch vier Zeilen oder nur zwei sein oder auch mehr als drei Spalten. Aber dass sie rechteckig ist, gehört zu ihrer Grundstruktur. Eine Tabelle ist immer ein Rechteck mit einer bestimmten Breite (Anzahl der Spalten) und einer bestimmten Höhe (Anzahl der Zeilen). Man kann sich dieses Rechteck auch mit einem regelmäßigen Gitter versehen vorstellen, wo Zellen abgegrenzt sind, in die dann die Werte eingetragen werden.

So weit, so gut. Das scheint eigentlich ganz simpel zu sein, und man mag sich fragen, worin da die Leistung Codds bestehen soll, so etwas einfaches als Grundelement für die Datenstruktur gewählt zu haben? Zwei Aspekte sind hier nach meiner Auffassung festzuhalten.

Einerseits ist diese Grundstruktur "Tabelle" in der Tat für uns Menschen gut überschaubar und übersichtlich. Eine Tabelle wie in unserem Beispiel können wir mühelos überblicken, was natürlich für größere Tabellen mit vielen Spalten und Zeilen nicht mehr so leicht möglich ist. Aber im Grunde stellen auch große Tabellen kein grundsätzliches Problem für unsere Anschauung dar, wenn wir wissen, dass auch sie die typische rechteckige Form haben, und wir können uns anhand eines Ausschnitts der Tabelle (einige wenige Spalten und Zeilen) schnell ein anschauliches Bild davon machen.

Andererseits - und das ist der zweite Aspekt - ist eine Tabelle natürlich nichts Atomares, sondern in sich selbst bereits ein komplexes Gebilde. In gewisser Weise kann man sich die Werte in den Zellen der Tabelle als atomare Bestandteile vorstellen (was natürlich auch nicht ganz stimmt), aber die Tabelle ist nicht einfach nur die Zusammensetzung der Zellenwerte zu einem Ganzen, sondern die Anordnung von Zellenwerten in der Weise, dass sie sich zu einer rechteckigen Tabelle mit vorgegebenen Spaltennamen vereinigen. Die Zellenwerte haben also keinen Vorrang, sondern sie sind nur dann ein Grundelement der Datenstruktur der Datenbank, wenn sie in einer Tabelle angeordnet sind.

Wir halten also fest: die Tabelle als elementarer Bestandteil einer relationalen Datenstruktur ist in sich schon etwas Komplexes, Vielfältiges. Durch ihre Regelmäßigkeit ist eine Tabelle aber dennoch etwas leicht Überschaubares für die Benutzer der Datenbank, was ihnen einen relativ komfortablen Umgang mit den Daten ermöglicht.

Wenn die Tabelle der Grundbestandteil des Modells ist, warum nennt Codd es dann relational und nicht z.B. tabellarisch? Diese Frage drängt sich natürlich auf, und Codd stellt sich diese Frage auch explizit im zweiten Paper: "Why call it the relational model? Why not call it the tabular model?" (B, 395). Die Frage zielt auf das Verhältnis der Begriffe "Tabelle" und "Relation". Ohne uns hier in begrifflicher Haarspalterei zu verlieren, wollen wir dennoch einige Anhaltspunkte nennen, wie die unterschiedlichen Rollen der beiden Begriffe zu verstehen sind.

Zunächst einmal ist festzuhalten, dass eine Tabelle der anschauliche Prototyp einer Relation ist. Die Tabelle ist gewissermaßen der anschauliche Einstiegspunkt, über den man sich dem Begriff "Relation" nähern kann. Überdies sind Tabellen gespeicherte Relationen, d.h. die Relationen eines Datenbanksystems, die eine - wenn man so will - materialisierte Existenzform in der Datenbank haben. Neben den Tabellen gibt es nämlich z.B. auch definierte Relationen in einer Datenbank, die aber nur virtuellen Charakter haben, und das sind die Views (Sichten). Views werden durch Grundoperationen definiert, die sich aus Struktur und Inhalt von einer oder mehreren Tabellen ableiten. Views können dabei zum einen reichhaltiger sein als die ihnen zugrunde liegenden Tabellen, z.B. wenn mehrere Tabellen verknüpft werden. Sie können aber auch eine reduzierte Komplexität darstellen, z.B. wenn aus einer Tabelle nur bestimmte Spalten und Zeilen ausgewählt werden. Wenn wir unsere Tabelle mit den Namen und Wohnorten "Person" nennen würden, könnte man in der verbreiteten Datenbanksprache SQL eine eingeschränkte View wie folgt definieren:

```
SELECT
Name
FROM Person
WHERE
Wohnort = 'Florenz'
```

Darin würde nur die Spalte "Name" erscheinen und nur die Zeilen, wo im Wohnort Florenz steht.

"Relation" ist damit gegenüber "Tabelle" der abstraktere Begriff, und Codd entlehnt ihn aus der Mathematik. Im ersten Paper schreibt er:

"The term relation is used here in its accepted mathematical sense. Given sets S1, S2, ... , Sn (not necessarily distinct), R is a relation on these n sets if it is a set of n-tuples each of which has its first element from S1, its second element from S2, and so on." (A, 379)

Wir sehen hier eine der beiden theoretischen Hauptquellen, aus denen Codd für das relationale Modell schöpft, und das ist die mathematische Mengenlehre. Die zweite Hauptquelle ist die Prädikatenlogik, wie wir später noch sehen werden. Eine Relation im Sinne Codds ist auf jeden Fall so definiert, dass aus n Mengen zunächst jeweils ein Element ausgewählt wird und diese Elemente zu einer Liste verknüpft werden (E1, E2, ... En). Damit erhält man ein n-Tupel. Dieses Auswahl- und Verknüpfungsverfahren kann nun beliebig oft wiederholt werden, so dass man eine Menge von n-Tupeln erhält, d.h. eine Menge von Mengen bzw. eine zweidimensionale Menge.

Charakteristisch für eine Relation im Sinne des Datenbanksystems ist auf jeden Fall, dass sie auf der Achse der Spalten, also der x-Achse - wenn man so will - eine geschlossene Menge darstellt und auf der Ebene der Zeilen, also der y-Achse, eine offene Menge. Die Spalten sind nämlich durch aufzählbare Namen genau definiert und übersteigen in der Praxis normalerweise nicht eine überschaubare Anzahl. Die Zeilen jedoch sind nicht einmal unbedingt genau gekennzeichnet - es sei denn durch einen Primärschlüssel - und können in ihrer Anzahl einen überschaubaren Horizont

schnell überschreiten. Millionen von Zeilen sind in Datenbanken durchaus keine Seltenheit, Millionen von Spalten dagegen eine Unmöglichkeit.

2.) Mit Relationen operieren: die relationale Algebra

Eine Datenbank ist nun nicht allein dadurch relational, dass sie viele Tabellen und weitere Relationen enthält, sondern sie stellt auch ein System von Rechenoperationen zur Verfügung, die man auf Relationen anwenden kann. Wenn wir hier von Rechenoperationen sprechen, ist dies zunächst einmal eine durchaus sinnvolle Analogie etwa zum Rechnen mit Zahlen. Denn ähnlich wie man z.B. zwei Zahlen addiert und dabei wieder ein neue Zahl als Ergebnis erhält (2 + 3 = 5), kann man auch zwei Relationen durch eine Operation verknüpfen und erhält als Ergebnis eine neue Relation. Ein wichtiges Charakteristikum des relationalen Operationssystems - der relationalen Algebra - können wir also hier schon festhalten: wenn man mit Relationen operiert, erhält man als Ergebnis wieder eine Relation.

Ganz analog zum Rechnen mit Zahlen gibt es dabei zum einen den Fall, dass eine Operation auf nur eine Relation angewandt wird und zum zweiten den Fall, dass eine Operation mehrere Relationen verknüpft. Bekannteste Operation, die auf eine Relation angewendet wird, ist ein SELECT-Befehl, der gewisse Spalten und Zeilen aus nur einer Tabelle auswählt. Wenn aber bei einer SELECT-Abfrage mehrere Tabellen durch einen JOIN-Befehl verknüpft werden, handelt es sich um eine Operation mit mehreren Relationen aus Ausgangswerten, d.h. auf mehreren Argumenten, um hier einen funktionstheoretischen Ausdruck zu verwenden.

Entscheidend ist hierbei der Punkt, dass die relationale Algebra nicht aus der Struktur der Relationen herausführt und somit das Ergebnis einer Operation wieder als Ausgangspunkt einer neuen Operation verwendet werden kann. Das führt z.B. zur Möglichkeit, SELECT-Befehle zu verschachteln, womit sehr weitreichende und komplexe Operationen möglich werden. Andererseits kann man prinzipiell jedes Ergebnis bzw. Zwischenergebnis einer Operation wieder als Tabelle veranschaulichen und zur Grundlage von Prüfungen und Diskussionen machen. Die Komplexität der Operationen bleibt damit zumindest vom Prinzip her beherrschbar.

Die Analogie zu den Rechenoperationen mit Zahlen hat bei den Relationen aber auch eine Grenze. Es wird hier zwar auch im engeren Sinne gerechnet, d.h. addiert, subtrahiert usw., aber das Rechnen ist eher als eine Nebenmöglichkeit zu verstehen, nicht als Hauptoperation. Die wichtigsten Operationen sind nämlich Mengenoperationen und damit eher logische Operationen. In der relationalen Algebra werden z.B. Teilmengen gebildet, Durchschnittsmengen oder auch verknüpfte Mengen. Im engeren Sinne gerechnet wird dann z.B. nur, um die Menge der ausgewählten Zeilen zu bestimmen, wenn etwa Bedingungen mit Zahlenberechnungen oder -vergleichen formuliert werden.

Um die von Codd entworfene relationale Algebra - d.h. das relationale Operationensystem - richtig zu verstehen, muss man sich nun aber neben der ersten mathematischen Hauptquelle für seine Idee, der Mengenlehre, noch die zweite Quelle vergegenwärtigen, und das ist die Prädikatenlogik. Wenn man

hier präziser argumentieren wollte, müsste man eigentlich von Prädikatenlogik erster Stufe sprechen, aber solche Feinheiten würden hier zu weit führen und uns vom eigentlichen Problem nur ablenken. Inwiefern spielt für Codd die Prädikatenlogik also eine wichtige Rolle?

Die Prädikatenlogik entwickelte sich im 19. Jahrhundert als Erweiterung der Aussagenlogik. Letztere hatte sich vor allem damit befasst, wie die Wahrheit oder Falschheit von zusammengesetzten Aussagen aus der Wahrheit bzw. Falschheit der Teilaussagen mit rein logischen Mitteln abgeleitet werden kann. Die Prädikatenlogik geht nun einen Schritt weiter bzw. setzt einen anderen Fokus, indem sie die innere Struktur einer Aussage genauer unter die Lupe nimmt. Einfache Aussagen wie "Die Alpen sind ein Hochgebirge." lassen sich nämlich als Zuordnung eines Prädikates ("... ist ein Hochgebirge") zu einem Eigennamen ("die Alpen") auffassen. Ein Prädikat im Sinne der Prädikatenlogik ist damit ein sprachlicher Ausdruck mit einer Leerstelle, in die ein Eigenname eingesetzt werden muss, damit eine Aussage entsteht, die dann wahr oder falsch sein kann. Prädikate in diesem Sinne sind z.B. auch Eigenschaften wie "... ist grün", aber auch Relationen wie "... ist größer als ...", so dass auch mehrstellige Prädikate möglich sind, d.h. Prädikate mit mehreren Leerstellen, in die Eigennamen eingesetzt werden müssen, um eine Aussage zu bilden.

Wichtig für das relationale Modell ist nun aber der Punkt, dass man Prädikate in diesem Sinne als zweifache Mengen interpretieren kann, nämlich einerseits als die Menge der Individuen bzw. Individuenbezeichner, die das Prädikat zu einer wahren Aussage vervollständigen und andererseits als Menge der Individuen, die das Prädikat zu einer falschen Aussage machen. Einfach ausgedrückt kann man sich das Prädikat "... ist ein Hochgebirge" als Menge aller Hochgebirge vorstellen und "... ist grün" als Menge aller grünen Gegenstände. Dies klingt zunächst recht simpel und ist es eigentlich auch. Offensichtlich hat man damit auch den Bezug zur Mengenlehre hergestellt, und wir müssen nun im Grunde nur noch genauer hinschauen, wie diese Idee im relationalen Modell nutzbar gemacht wird.

Hierzu müssen wir nun darauf achten, wie etwa in der Abfragesprache SQL die Auswahl der Zeilen eingeschränkt wird. Wir können dazu auf unser Beispiel mit der Tabelle "Person" zurückkommen und auf die Einschränkung der Ergebniszeilen nach dem Wohnort Florenz. Wir definieren nämlich hier unsere Ergebnismenge auf Zeilenebene dadurch, dass wir ein bestimmtes Prädikat formulieren, das von gewissen Zeilen erfüllt wird und von anderen Zeilen nicht. Das Prädikat war hier "... wohnt in Florenz". Nur eine Zeile, d.h. eine Person ergänzte dieses Prädikat zu einer wahren Aussage. Im relationalen Modell formulieren wir also Prädikate als Anforderungen an das Datenbanksystem, die Ergebnismenge zurückzuliefern, die das Prädikat zu einer wahren Aussage ergänzen. Dabei kann das Prädikat natürlich auch mehrstellig sein, was etwa bei JOINs von mehreren Tabellen sogar immer der Fall ist, aber es muss so formuliert werden, dass die Spaltenstruktur der beteiligten Tabellen dem System erlaubt, für jede Zeile eine eindeutige Entscheidung zu fällen, ob dadurch das Prädikat erfüllt wird oder nicht.

Entscheidend ist hier der Punkt, dass im relationalen Modell Zielmengen durch Prädikate definiert werden, die einfach strukturiert, aber auch sehr komplex sein können. Wichtig ist aber dabei, dass die

Zielmenge bzw. Zielrelation nicht durch Auflistung ihrer einzelnen Elemente definiert werden muss. Welche Zeilen es genau sind, die in der Ergebnismenge erscheinen sollen, z.B. welche Zeilennummern, muss der Benutzer der Datenbank oder der Programmierer nicht mehr angeben. Er muss lediglich noch auf einer - aus Sicht des technischen Systems - hochabstrakten Stufe ein passendes Prädikat formulieren, um die Ergebnismenge zu spezifizieren. Den Rest erledigt das Datenbanksystem von selbst. Und was für das technische System hochabstrakt ist, ist es für den Menschen eben gerade nicht, weil er die Spezifikation im Grunde in der begrifflichen und inhaltlichen Logik der jeweiligen Daten formulieren kann, ohne sich um den technischen bzw. physischen Weg der Ergebnisfindung kümmern zu müssen.

Während die Festlegung der Ergebnismenge einer SQL-Abfrage auf Zeilenebene durch Prädikate erfolgt, bleibt aber die Spaltenstruktur eine Einzelauflistung der Namen. Man hat hier lediglich die Möglichkeit, alle überhaupt verfügbaren Spalten in die Ergebnismenge einzubeziehen, indem man Konstrukte wie SELECT * nutzt, wo das Sternchen für "alle Spalten" steht. Das ist dann aber gar keine Einschränkung und führt bei großen Datenmengen im Ergebnis denn auch eher zu Problemen als dass es helfen könnte. Die Vermutung liegt jedoch nahe, dass die Möglichkeit einer prädikativen Festlegung auch bei den Spalten das Vorstellungsvermögen der meisten Menschen schnell überfordern würde. Dass auf dieser Ebene also eine eher feste und leicht überschaubare Namensauflistung erfolgen muss und nur auf der Zeilenebene eine Auswahl mithilfe der eher beweglichen Prädikate geschieht, garantiert gewissermaßen, dass eine bestimmte Grenze von Komplexität nicht überschritten wird. IT-Systeme können ja nur dann einen einigermaßen weiten Verbreitungsgrad finden, wenn die Schnittstellen für die menschliche Benutzung zwar mächtig, aber nicht zu komplex sind. Genau dies ist nach meiner Auffassung bei der Relation als Grundelement des relationalen Modells gut erfüllt und hat den breiten Erfolg desselben mitbedingt.

3.) Über Relationen reden: der gemeinsame Blick auf die Daten

Mit dem Problem der Überschaubarkeit sind wir bereits bei einem weiteren wichtigen Aspekt des relationalen Modells angekommen: die Möglichkeit der Kommunikation über die Daten in einer Datenbank. Im zweiten Paper von 1981 kann man dazu erstaunliche Worte bei Codd lesen, wenn er über Gründe spricht, warum das relationale Modell als Grundlage für Datenbanken vorgeschlagen wurde. Neben der Unabhängigkeit der logischen Datenebene von der physischen nennt er hier als zweiten Grund folgenden:

"A second objective was to make the model structurally simple, so that all kinds of users and programmers could have a common understanding of the data, and could therefore communicate with one another about the database. We call this the communicability objective." (B, 394)

Dass jede Operation in der relationalen Algebra wieder eine Relation als Ergebnis hat, führt also nicht nur zur strukturellen Geschlossenheit des Operationssystems, sondern auch dazu, dass man prinzipiell in jede Relation hineinschauen und darüber kommunizieren kann. Relationen - und man

kann sich diese ja immer als Tabellen veranschaulichen - sind also gewissermaßen die Fenster der Datenbank, durch die man in die Daten hineinschauen kann. Selbst die komplexeste Aufbereitung von Daten, die ein Datenbankentwickler meist in gespeicherten Prozeduren hinterlegt, kann also nur von Relation zu Relation fortschreiten und prinzipiell jedes Zwischenergebnis als einsehbare Tabelle oder Sicht zur Verfügung stellen, auch wenn dies nicht immer sinnvoll und notwendig ist.

Damit steht das relationale Modell nach meinem Verständnis z.B. im Kontrast zur Idee der Kapselung, die etwa in der objektorientierten Programmierung eine herausgehobene Rolle spielt. Die Kapselung verbirgt ja absichtlich Komplexität und beschränkt die Fenster von Objekten auf gewissen Abfrage- und Änderungsmethoden (get- und set-Methoden). Die Schnittstellen, die ein objektorientiertes System nach außen freigibt, sind also wohldefiniert und eingeschränkt. Sie sind im System eher die Ausnahme, was natürlich seine Gründe hat und für bestimmte Zwecke eine angemessene Lösung ist. Das relationale Modell dagegen ist aus Prinzip gewissermaßen offen, wenn nicht der Zugriff auf bestimmte Tabellen durch Berechtigungsvergaben eingeschränkt ist, was natürlich auch wichtig ist. So nennt auch Codd Benutzer und Programmierer bei seinen Ausführungen oft in einem Atemzug. Sie haben zwar verschiedene Aufgaben im Umgang mit den Daten eines Datenbanksystems, aber sie arbeiten und kommunizieren auf breiter Basis gemeinsam an den Daten und über die Daten. Im objektorientierten Modell dagegen werden Benutzer und Programmierer tendenziell eher auf verschiedenen Seiten situiert und durch einen stark eingeschränkten und wohlumgrenzten Schnittstellenbereich miteinander verbunden.

Der Vergleich des relationalen Modells mit objektorientierten Systemen soll hier auch nicht einen Gegensatz akzentuieren, den es eigentlich nicht gibt. Man kann dadurch aber zeigen, dass die Kommunikationsmodelle unterschiedlich sind, die im Ansatz der Systeme zugrundegelegt werden. Relationale Datenbanken haben demzufolge u.a. also dort ihren sinnvollen Einsatzort, wo Benutzer als inhaltliche Experten für die Daten und Datenbankentwickler als technische Experten des Datensystems eng miteinander zusammenarbeiten müssen, um komplexe Datenstrukturen gemeinsam beherrschen zu können. In anderen Einsatzbereichen, wo auch grundlegend andere Kommunikationsstrukturen vorherrschen, können ganz andere informationstechnische Modelle besser geeignet sein.

4.) Mengen deklarativ verarbeiten: die Vermeidung von algorithmischen Verfahren

Eine grundlegende Fähigkeit von Computern ist es, schnell hintereinander besonders viele Rechenoperationen ausführen zu können. Die meisten Programmiersprachen bieten deshalb die Möglichkeit, den Computer zu veranlassen, eine bestimmte Operation beliebig oft zu wiederholen und dabei von Schritt zu Schritt einen Parameter zu verändern oder das Ergebnis des letzten Schrittes zum Ausgangspunkt des nächsten Schrittes zu machen. Auf diese Weise ist es z.B. möglich, einen Computer so zu programmieren, dass er Schach spielen kann. Man weist dabei den Computer an, in einer bestimmten Stellung alle möglichen Züge durchzugehen, für jeden Zug alle möglichen Folgezüge usw. bis zu einer bestimmten Tiefe. Die Züge werden nach bestimmten Kriterien geprüft

und bewertet, und ein Zug wird vom Computer als bester berechnet und ausgeführt. Diese Fähigkeit von Computern hat von jeher viele Menschen beeindruckt und vermutlich auch bei vielen das Bild dessen bestimmt, was Computer so alles leisten können.

Computer können also Problemlösungen finden, die nur durch die Ausführung vieler verschiedener Schritte hintereinander möglich sind, d.h. durch algorithmische Verfahren, wo das menschliche Denken mit seinem langsamen Tempo bald an seine Grenzen kommt. Der Programmierer muss dem Computer nur beibringen, wie er die sich wiederholenden und aneinander anschließenden Operationen sachgemäß auszuführen hat.

Dies im Hinterkopf habend mag es nun einigermaßen überraschen, wenn Codd darauf drängt, im Operationssystem der Datenbank - der relationalen Algebra also - solche algorithmischen Verfahren wie Iterationen und Rekursionen zu vermeiden, wenn es irgend geht. Im zweiten Paper liest man z.B.:

"The relational model calls not only for relational structures (which can be thought of as tables), but also for a particular kind of set processing called relational processing. Relational processing entails treating whole relations as operands. Its primary purpose is loop-avoidance, an absolute requirement for end users to be productive at all, and a clear productivity booster for application programmers." (B, 396)

Woher kommt aber dieses Drängen, die Steuerungssprache für das relationale Operieren soll ohne den Gebrauch von Iterationen und Rekursionen auskommen - "... without making use of iteration or recursion statements" (B, 397)?

Die Antwort wird schon im eben angeführten Zitat gegeben. Iterationen sind aufwendige Programmkonstrukte, die zwar von Programmierern eingesetzt werden können, aber in jedem Falle sorgfältig aufgebaut werden müssen und ausgedehnte Tests notwendig machen, um fehlerfrei zu funktionieren. Für typische Endnutzer von Datenbanken ist es sogar sehr schwierig, Iterationen zu programmieren, denn dies setzt eine gewisse Übung und Gewöhnung in solchen Verfahrensweisen voraus. Normalerweise sind aber Endnutzer Fachleute für bestimmte Bereiche, z.B. Medizin, und haben keine Programmiererfahrung. Dem normalen menschlichen Denken sind Iterationen, wie man sie in der Programmierung anwendet, eher fremd. Der Alltagsverstand - auch der fachlich gebildete - strukturiert komplexe Sachverhalte nämlich nicht als iterative Prozesse, sondern durch Begriffe bzw. Merkmale oder Prädikate. Mengen von Gegenständen z.B. lassen sich daher leichter und natürlicher durch Prädikate kennzeichnen, die auf sie zutreffen, als durch ein iteratives Verfahren, das die Gegenstände nacheinander durchläuft.

Sicher wird man nicht bestreiten können, dass iterative Verfahren auch im Bereich der Datenbanken nicht immer durch prädikative Charakterisierungen ersetzt werden können. Wenn man sich den Umgang mit Datenbanken in der Praxis näher anschaut, sieht man aber, dass dies nur relativ selten vorkommt. In den weitaus meisten praktischen Fällen ist es durchaus möglich, iterative Verfahren auf

der logischen Ebene des Umgangs mit den Daten gänzlich durch (komplexe) Prädikate zu ersetzen und die Iteration komplett dem technischen System zu überlassen. Endnutzer und Datenbankprogrammierer müssen dann nur die Spezifikation der Probleme, die gelöst werden sollen, in logisch korrekte Prädikatenbündel übersetzen. Auch die Kommunikation zwischen Endnutzer und Programmierer kann sich dann auf diesen Schritt konzentrieren.

Kurioserweise zeigt aber die Erfahrung, dass Endnutzer und Programmierer oft aneinander vorbeireden, weil viele Programmierer sich in ihrem Denken sehr stark auf iterative und andere maschinennahe Verfahren eingestellt haben, so dass ihre Redeweise in der Kommunikation mit den Endnutzern aus deren Sicht kryptisch wirkt und eine gemeinsame Problemklärung behindert. Andererseits sind viele Endnutzer nicht daran gewöhnt, ihre Anforderungen so zu formulieren, dass sich daraus passende SQL-Anweisungen ableiten lassen. Ein Datenbank-Programmierer, der an einen deklarativen Programmierstil gewöhnt ist und die Datenaufbereitung vorzugsweise begrifflich und prädikatenlogisch aufbaut, wird die direkte Kommunikation mit den Endnutzern leichter zielorientiert gestalten können. In der Praxis ist damit ein direkter Dialog zwischen Programmierer und Endnutzer ausreichend und effektiv, was Zeit und Kosten in der Entwicklung einsparen hilft. Wenn man aber die Kommunikation zwischen Programmierer und Endnutzer durch einen zwischengeschalteten Projektmanager verkomplizieren muss, um zu akzeptablen Projektresultaten zu kommen, steigen Aufwand und Kosten enorm an, und es kommt das "stille Post" Problem hinzu: komplexe Kommunikationsstrukturen werden anfällig für Missverständnisse. Solche Fragen werden natürlich heute auch intensiv diskutiert.

Prinzipiell kann man sagen, dass die relationale Algebra, die heute in der Datenbank-Praxis größtenteils in der Sprache SQL zur Anwendung kommt, einem deklarativen Programmierstil verpflichtet ist und sich damit vom imperativen Programmierstil unterscheidet. Während im imperativen Programmierparadigma die Schrittfolge der Lösungsfindung im Vordergrund steht, setzt der deklarative Stil den Hauptakzent auf die Formulierung des Ergebnisses, das erreicht werden soll und überlässt es der Maschine zu großen Teilen, den Weg dorthin zu finden.

5.) Zusammenfassung

Wer im Bereich der Datenbanken tätig ist, merkt schon bald, dass auf diesem Feld die relationalen Systeme eine sehr große Verbreitung in der industriellen Praxis haben. Außerdem habe ich selbst in diesem Bereich über Jahre hinweg die Erfahrung gemacht, dass relationale Datenbanken sich für die Lösung von praktischen Erfordernissen durchaus gut eignen. Dies führte mich auf die Frage, warum das eigentlich so ist, und ich habe nach Gründen dafür in der Ausarbeitung des relationalen Modells selbst gesucht.

Schaut man nun in die Schriften von E.F. Codd, der das relationale Modell begründet hat, so wird man auf der Suche nach Antworten schnell fündig. Folgende Aspekte der Theorie wollte ich hier hervorheben:

- Die Daten einer Datenbank sind nur auf einer fest definierten logischen Ebene zugänglich und verarbeitbar. Die physische Speicherung der Daten wird vollständig an die Maschine übergeben.

- Grundelement des logischen Modells ist die Relation und damit eine zweidimensionale Menge, die man sich als eine Tabelle veranschaulichen kann. Das Grundelement ist damit einerseits übersichtlich und andererseits doch ausreichend komplex, um auch weniger einfache Datenstrukturen abbilden zu können.

- Das Modell definiert ein System von Operationen zur Verarbeitung von Daten - die relationale Algebra. Ausgangs- und Endpunkte solcher Operationen sind wieder Relationen, so dass das System in sich geschlossen ist, was wiederum zur guten Überschaubarkeit beiträgt. Jedes Zwischenergebnis einer komplexen Folge von Operationen kann prinzipiell als Tabelle veranschaulicht werden.

- Theoretische Quellen des relationalen Modells sind vor allem die Mengenlehre und die Prädikatenlogik. Dadurch können komplexe Massendaten verwaltet werden, aber in einer logischen Form, die dem Alltagsverstand vergleichsweise leicht zugänglich ist.

- Alle Grundaspekte des relationalen Modells zielen darauf ab, die Daten auf jeder Ebene einfach veranschaulichen zu können und damit eine effektive Kommunikation über die Daten an jeder Stelle zu fördern.

- Der deklarative Stil der Datenbankprogrammierung in SQL rückt Endnutzer und Datenbank-Entwickler eng zusammen und vermeidet ein Auseinanderfallen ihrer Denkweisen. Dadurch wird es überhaupt erst möglich, die teilweise sehr komplexe inhaltlich-fachliche Logik eines bestimmten praktischen Bereiches wie etwa der Medizin, in einer komplexen Datenstruktur angemessen abzubilden.