3 Datenbankentwurf

Im vorigen Kapitel wurde bereits auf die Notwendigkeit eines logischen und physikalischen Konzeptes für die Datenbank hingewiesen. Gerade unter Berücksichtigung der Aspekte, die zur Qualitätssicherheit beitragen sollen, ist der Datenbankentwurf ein wichtiger Schritt zur erfolgreichen Umsetzung des Gesamtprojektes. Fehler, die in dieser Phase gemacht werden, lassen sich später nur mühsam bereinigen. Die hier zu nennenden Aspekte sind *Vollständigkeit*, *Korrektheit*, *Minimalität*, *Lesbarkeit* und *Modifizierbarkeit*.

Beim Datenbankentwurf geht es in erster Linie darum, aus den gegebenen Anforderungen eine logische und physische Struktur für die Datenbank zu erarbeiten.

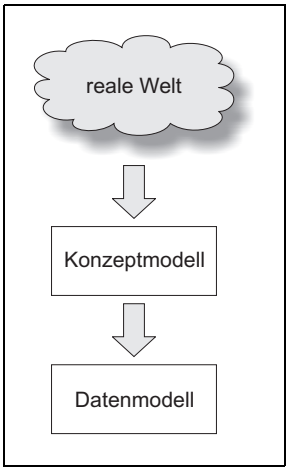

Abbildung 3.1: Von den Anforderungen der realen Welt zum Konzept

Die Normalisierung der Daten und der Entwurf mit dem Entity-Relationship Modell sind zwei Verfahren, die sich anbieten, um entsprechende Konzepte zu erarbeiten.

Die Normalisierung kann ausschließlich bei relationalen Datenmodellen verwendet werden. Das Entity-Relationship Modell hingegen ist unabhängig vom Datenmodell. Zunächst soll jedoch ein Begriff geklärt werden, der in diesem Zusammenhang eminent wichtig ist.

3.1 Ungewollte Vielfalt – Redundanz

Ein wichtiger Begriff im Zusammenhang mit Datenbanken ist der Begriff der *Redundanz*. Redundanz betrifft nicht nur Datenbanksysteme, sondern IT-Systeme im Allgemeinen. Betrachten wir z.B. Geschäftsprozesse. Gerade hier ist es wichtig, redundante Abläufe zu beseitigen, um die Laufzeiten eines Prozesses zu verringern. Bei der Programmierung, insbesondere bei der objektorientierten Programmerstellung, ist es ein wesentliches Ziel, Redundanz zu vermeiden.

Welche Bedeutung hat nun der Begriff *Redundanz*?

Übersetzt bedeutet Redundanz, dass etwas mehrmals und zwar überflüssigerweise vorliegt. Das ist zunächst nichts Verwerfliches, wenn wir an Erbformationen in unseren Zellen denken. In einer Zelle steckt die Erbinformation aller anderen Zellen. Manchmal wird auch ganz bewusst Redundanz erzeugt. In der Werbung meint man manchmal, dass das Label der Firma gar nicht oft genug im Bild erscheinen kann. Welche unangenehmen Folgen Redundanz jedoch haben kann soll das Fallbeispiel verdeutlichen.

Beispiel 3.1:

Ein Buch existiert in der Datenbank eines Buchgeschäftes zweimal als Objekt. Dem Objekt ist jeweils der Preis als Eigenschaft zugewiesen. Nun hat es einen Preisnachlass gegeben. Eine Datenbankprozedur hat dabei den Preis bei einem Objekt angepasst, jedoch nicht bei dem anderen. Ausgehend vom neuen Preis, werden die Bücher neu deklariert. Ein Kunde entscheidet sich, dieses Buch zu kaufen. Er wird an der Kasse jedoch überrascht, als er erfährt, dass das Buch teurer sein soll als die Preisangabe auf dem Etikett es aussagt. An der Kasse wurde der alte Preis berücksichtigt. Die Kassiererin ist verwirrt und der Kunde verärgert (siehe Abbildung 3.2).

Das Problem von redundanten Daten ist also, dass sie unterschiedlich geändert werden könnten, und dass man danach nicht einmal mehr weiß, welche eigentlich die aktuellen Daten sind.

In der Fachsprache bezeichnet man diese Problematik als die Gefahr von *Inkonsistenz*, d.h. Daten können sich widersprechen. Der Ressourcenverbrauch (Speicherkapazitäten) ist im Vergleich dazu ein eher sekundäres Problem.

Aus diesem Grund sollten Anstrengungen unternommen werden, um entsprechende Redundanz in einem IT-System zu vermeiden.

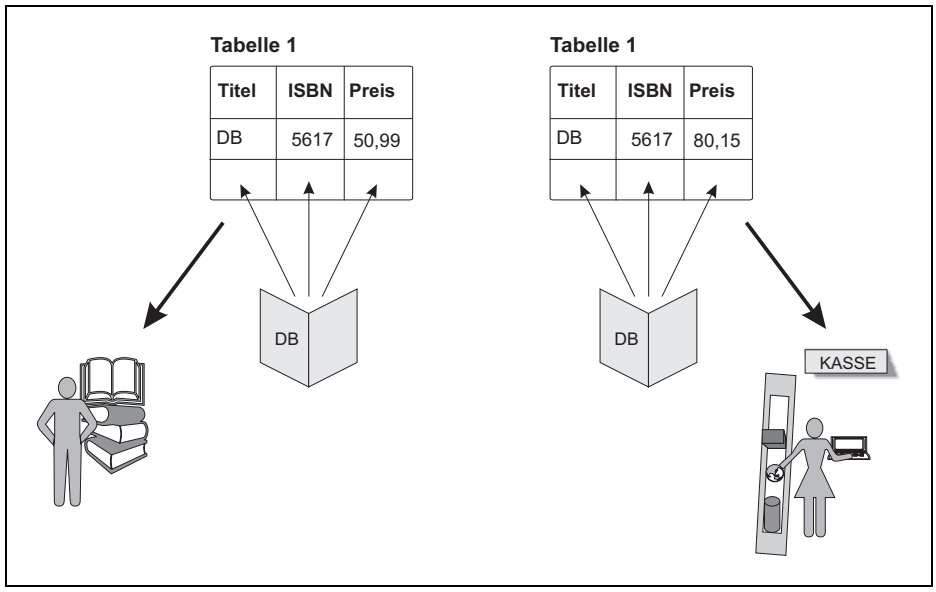

Abbildung 3.2: Fallbeispiel für Redundanz im Buchgeschäft

3.2 Konzepterstellung mit dem ER-Modell

Die Abbildung von Informationsstrukturen der realen Welt auf das konzeptuelle Schema einer Datenbank wird meist nicht direkt vorgenommen, sondern vielmehr über den Zwischenschritt der Abbildung auf ein semantisches Datenmodell, das dann auf das konzeptuelle Schema transformiert wird.

Das am häufigsten zu diesem Zweck eingesetzte Datenmodell ist das *Entity-Relationship Modell*, das in der heute gebräuchlichen Form 1976 von *P. Chen* vorgestellt wurde und in der Folge diverse Erweiterungen erfuhr. Das Entity-Relationship Modell geht grundsätzlich davon aus, dass sich die im Rahmen des Modells zu verwaltende Information als eine Menge von Objekten (Entities) beschreiben lässt, zwischen denen Beziehungen (Relationships) herrschen. Ähnlich wie das relationale Modell, sieht das Entity-Relationship Modell Attribute zur Beschreibung von Objekten und Beziehungen vor und macht Gebrauch vom *Primärschlüsselkonzept*. Darüber hinaus sind jedoch weitergehende Konzepte wie *Generalisierung* oder *Aggregation* vorhanden.

3.2.1 Die Elemente im ER-Modell

Das ER-Modell kommt mit einigen wenigen Elementen aus, die in diesem Abschnitt vorgestellt werden sollen.

Bezeichnung	Englische Bezeichnung	Bedeutung
Entitäten	entities	Eine Entität ist das Abbild eines Objektes aus der Realwelt.
Entitätstyp	entity type	Die Menge der Entitäten, die gleiche Eigenschaften haben und deshalb zusammengefasst werden können.
Beziehungen	relationships	Über die Beziehungen werden die Entitäten sinnvoll verknüpft.
Beziehungstyp	relationship type	Beziehungstypen stellen auch wieder eine Zusammenfassung der Beziehungen gleicher Eigenschaften dar.

Tabelle 3.1: Elemente des ER-Modells

Bei der Darstellung des ER-Modells in einem Diagramm werden die abgebildeten Symbole verwendet:

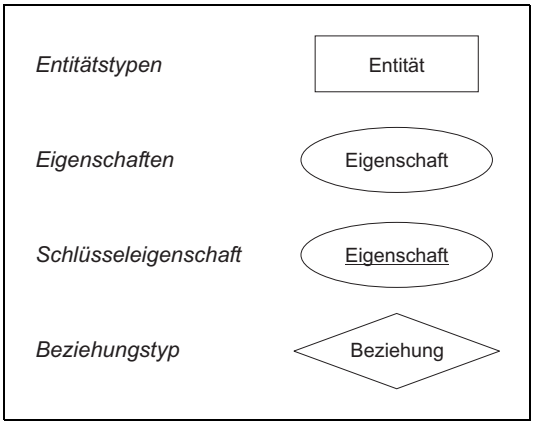

Abbildung 3.3: Symbole des ER-Modells

3.2.2 ER-Modell für die Beispieldatenbank

An dieser Stelle soll nun das Konzept für unsere Beispieldatenbank erstellt werden. Es handelt sich um die Strukturen eines Dienstleistungsunternehmen, welches die internen und externen Geschäftsabläufe automatisieren will. Die Datenbank soll in der Form konzipiert werden, dass sie auch für Firmen mit ähnlichen Geschäftsstrukturen verwendet werden kann.

Dem *Kunden* werden von den *Mitarbeitern* des Unternehmens verschiedene Dienstleistungen angeboten, die wir im Folgenden zusammenfassend *Service* nennen. Dieser einfache funktionale Zusammenhang wird nun in einem *ER-Diagramm* dargestellt.

1. Schritt: Selektion der Objekte

Abbildung 3.4: Selektion der Objekte

Zunächst werden die Entitäten (Objekte) herausgesucht und mit einem Rechteck als Symbol dargestellt. Mitarbeiter, Service und Kunde sind eindeutig identifizierbare Exemplare aus der Realwelt. Die Menge der Entitäten haben gleiche Eigenschaften und gelten dann als *Entitätstyp*. So hat ein Mitarbeiter aus der Buchhaltung, ebenso wie die Mitarbeiterin aus der Marketingabteilung, die Eigenschaften *MitarbeiterNr*, *Name* und *Geburtstag*. Dabei müssen die Werte dieser Eigenschaften nicht identisch sein. Im nächsten Schritt werden also den Objekten die Eigenschaften zugewiesen.

2. Schritt: Zuweisen der Eigenschaften

Üblicherweise sind für die verschiedenen *Entitäten* weitere Informationen, d.h. weitere Eigenschaften vorhanden.

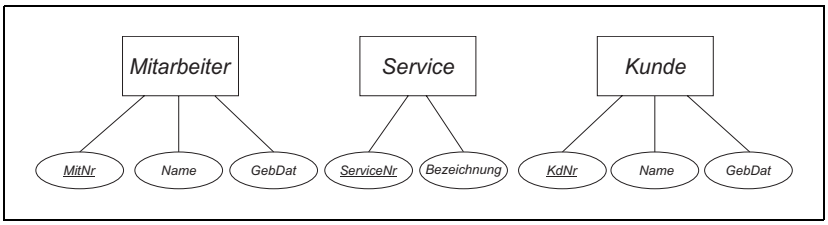

Abbildung 3.5: Zuweisen der Eigenschaften

Eine oder mehrere Eigenschaften eines Entitätstyps müssen als *Primärschlüssel* ausgewiesen werden. Keine Entität darf existieren, die nicht durch seinen Primärschlüsselwert eindeutig von allen anderen Entitäten seines Typs unterscheidbar ist. Bezeichner von Entitätstypen müssen untereinander eindeutig sein, Eigenschaftbezeichner müssen innerhalb eines Typen eindeutig sein. So darf innerhalb des Modells zwar die Eigenschaft *Name* sowohl für den Kunden als auch für den Mitarbeiter vergeben werden, jedoch ist es unzulässig, dem Mitarbeiter eine weitere Eigenschaft mit der Bezeichnung »Namen« zuzuweisen, die z.B. den Namen der Abteilung beinhaltet.

3. Schritt: Objekte mittels Beziehungstypen verknüpfen

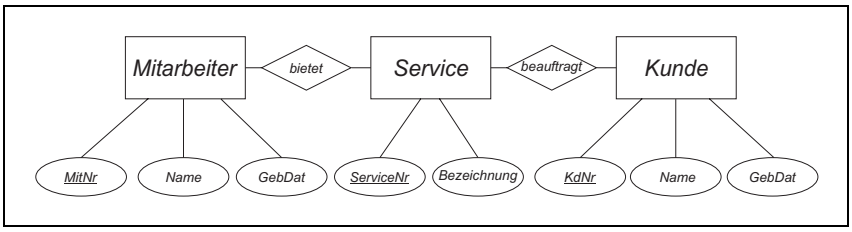

Abbildung 3.6: Objekte mittels Beziehungstypen verknüpfen

Beziehungen zwischen Entitäten werden durch sogenannte Beziehungstypen beschrieben, die ebenso wie Entitäten strukturierte Datenobjekte sind. Ein Beziehungstyp muss wenigstens zwei Entitätstypen zueinander in Beziehung setzen. Im Gegensatz zu den Entitätstypen ist ein Beziehungstyp jedoch kein eigenständiges Objekt, sondern kann nur existieren, wenn auch die referierten Entitätstypen existieren. In dem Beispiel werden die Entitäten *Service* und *Kunde* über einen Beziehungstypen mit der Bezeichnung »beauftragt« verknüpft. Die Bezeichnung des Beziehungstypen sollte so gewählt werden, dass sich daraus eine Aussage über die Art der Beziehung ableiten lässt. Dies erhöht die Lesbarkeit des Diagramms.

4. Schritt: Beziehungstypen kennzeichnen

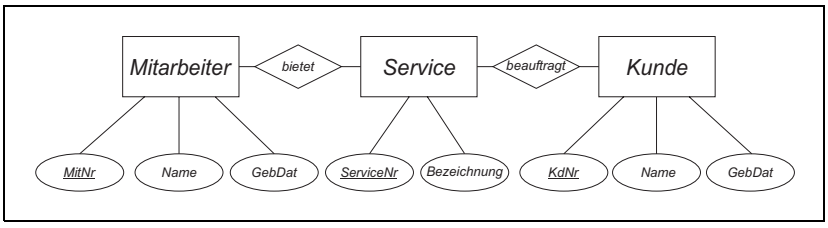

Abbildung 3.7: Beziehungstypen kennzeichnen

Schließlich werden im Diagramm die Beziehungstypen gekennzeichnet. In dem angegebenen Beispiel soll ein Mitarbeiter nur für jeweils eine Art von Serviceleistung zuständig sein. Umgekehrt kann aber ein und dieselbe Serviceleistung von mehreren Mitarbeitern angeboten werden. Daraus ergibt sich eine *1:n*-Beziehung, die entsprechend der Abbildung 3.7 im Diagramm gekennzeichnet wird.

3.2.3 Generalisierung

Dem Konzept der Bildung von *Untertypen* kommt durch den immer stärkeren Einfluss von objektorientierten Ansätzen bei der Datenbanktechnologie eine besondere Bedeutung zu.

Die Attributstruktur wird teilweise von anderen Typen geerbt, deren Elemente zugleich auch Elemente der vererbenden Typen sind. Im erweiterten Entity-Relationship Modell sind solche Zusammenhänge durch einen speziellen Relationship-Typen ausdrückbar, der als *Generalisierung* bezeichnet wird.

Eine Generalisierungsbeziehung besteht zwischen einem ausgezeichneten, im Folgenden als *Obertyp* bezeichneten Entitätstypen, und einer Menge von anderen Entitätstypen, die im Folgenden als Untertypen bezeichnet werden. Der Obertyp fasst dabei die Entitäten aller Untertypen zusammen. Er kann Eigenschaften besitzen, die dann den Untertypen gemein sind, und er kann wie jeder andere Entitätstyp bei der Bildung von Beziehungstypen verwandt werden.

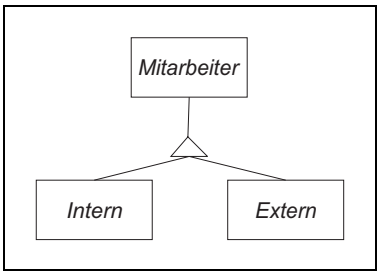

Abbildung 3.8: Beispiel der Generalisierung

Das Bestehen einer Generalisierungsbeziehung zwischen *Entitätstypen* wird durch ein Dreieck dargestellt, von dessen Spitze eine Kante zum Obertypen und von dessen Unterseite Kanten zu Untertypen ausgehen.

Das angegebene Beispiel stellt dar, dass Mitarbeiter eine Obermenge von internen und externen Mitarbeitern ist. »Interne« und »Externe« teilen gewisse Eigenschaften (z.B. Namen), weisen andererseits aber auch spezifische Eigenschaften auf, die sich aus der Art des Beschäftigungsverhältnisses ergeben.

Im konzeptuellen Entwurf spielen diese Verschärfungen meist aber eine geringe Rolle. Bedeutsam ist vielmehr, dass die Generalisierung eine Möglichkeit bietet, beim Entwurf von mehreren, speziellen Entitätstypen zugunsten eines Obertyps, der gemeinsame Merkmale bündelt, zu abstrahieren.

3.2.4 Aggregation

Da Datenobjekte oft aus anderen zusammengesetzt sind, ist die Darstellbarkeit von *aggregierten Objekten* eine wichtige Voraussetzung für die Modellierung von Datenstrukturen auf einem hohen, abstrakten Niveau. Während das ursprüngliche Entity-Relationship Modell keine speziellen Konzepte zur Darstellung von *Aggregationshierarchien* vorsah, kann eine geschachtelte Aggregation im erweiterten Entity-Relationship Modell durch Umwandlung von Beziehungstypen zu Entitätstypen bewirkt werden.

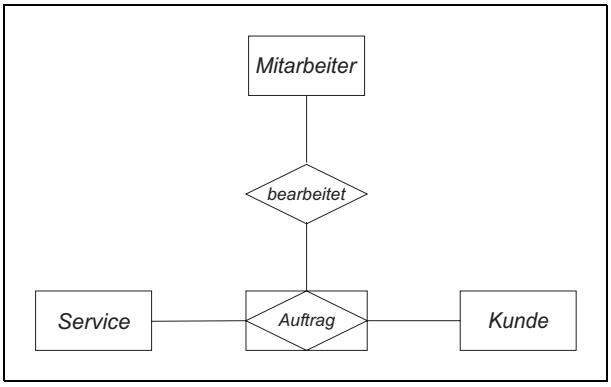

Abbildung 3.9: Beispiel der Aggregation

Ausgangspunkt der Erweiterung ist dabei, dass sich ein Beziehungstyp zwischen Entitäten als elementarer Konstruktor für aggregierte Objekte benutzen lässt, sofern man sie als ein aus den beteiligten Entitäten zusammengesetztes Datenobjekt auffasst. In diesem Sinne liegt es nahe, auch Beziehungstypen zuzulassen, die andere Beziehungstypen zueinander in Beziehung setzen. Da dies im Modell verboten ist, führt man im erweiterten Modell ein spezielles Symbol – die von einem Rechteck umschlossene Raute ein, die sowohl den Beziehungstyp als auch den Entitätstyp gleichermaßen symbolisiert.

In der Abbildung 3.9 wird ein Typ *Auftrag* dargestellt, der sich aus der Aggregation von *Kunden* und *Service* ergibt. Dieser neue Typ kann Eigenschaften besitzen, die sowohl dem Kunden als auch dem Service zuzuordnen wären. Ein typisches Beispiel für diesen Fall wäre das Datum an dem der Kunde einen Service beauftragt. Die neue Eigenschaft *Auftragsdatum* kann nun problemlos an den Typen *Auftrag* vergeben werden und bezieht sich auf Kunden und Service gleichermaßen.

3.2.5 Abbildung auf das relationale Datenmodell

Die Abbildung von ER-Modellen auf das relationale Datenmodell kann grundsätzlich vorgenommen werden, indem Entitätstypen und Beziehungstypen auf eigenständige Relationen (Tabellen) abgebildet werden.

Die Überführung eines Entitätstyps in eine eigenständige Relation ist bei vorgegebenen Attributen und Primärschlüssel leicht möglich. Das Modell aus Abbildung 3.7 würde umgesetzt in Tabellenform wie folgt aussehen:

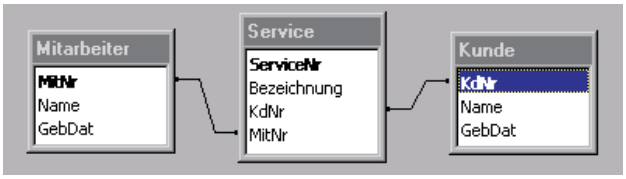

Abbildung 3.10: Die Tabellen Mitarbeiter, Service und Kunde

Die Spalten *MitNr*, *ServiceNr* und *KundenNr* bilden jeweils den *Primärschlüssel* in den drei Tabellen. Um eine Beziehung zwischen den Tabellen aufbauen zu können, müssen darüber hinaus Fremdschlüssel definiert werden. In der Tabelle *Mitarbeiter* ist die Spalte *ServiceNr* der Fremdschlüssel, der in Verbindung mit dem Primärschlüssel *ServiceNr* aus der Tabelle *Service* eine Verknüpfung zwischen den beiden Tabellen ermöglicht. Relativ leicht können so *1:1*- und *1:n*-Beziehungen erstellt werden. Eine *n:m*-Beziehung, wie sie zwischen den Objekten *Service* und *Kunde* vorliegt, ist, wie dies in Abbildung 3.10 geschehen ist, nicht realisierbar. Dazu müsste man mit einer Hilfstabelle arbeiten oder man bedient sich des Modells aus Abbildung 3.9, bei dem ein Beziehungstyp zum Entitätstyp aggregierte. Bei diesem Beispiel könnte die Tabellenstruktur ähnlich aussehen. Es wird eine weitere Tabelle *Auftrag* angelegt, so wie es in Abbildung 3.11 dargestellt ist.

Bei dieser Tabellenstruktur liegt zwischen den Tabellen *Kunde*, *Service*, *Mitarbeiter* und der neuen (Hilfs-) Tabelle jeweils eine *1:n*-Beziehung vor, wenn wir voraussetzen, dass jeweils nur ein Service beauftragt wird.

Bei der Überführung von Beziehungstypen (einschließlich Aggregationen) werden im Allgemeinen alle Eigenschaften des Beziehungstypen, sowie die Primärschlüssel der beteiligten Entitätstypen in eine eigenständige Relation überführt. Wenn Relationship-Typen eine *1:1*- oder *1:n*-Beziehung realisieren, kann aber auch darauf verzichtet werden und stattdessen der Relationship-Typ mit der Relation eines beteiligten Entity Typen adäquat verschmolzen werden. Der Primärschlüssel, der sich ergebenden Relation setzt sich dann auch aus den Primärschlüsseln der beteiligten Relationen zusammen und kann (falls dies das Datenbanksystem erlaubt) zusätzlich als Fremdschlüssel deklariert werden, um die referenzielle Integrität von Relationships zu modellieren.

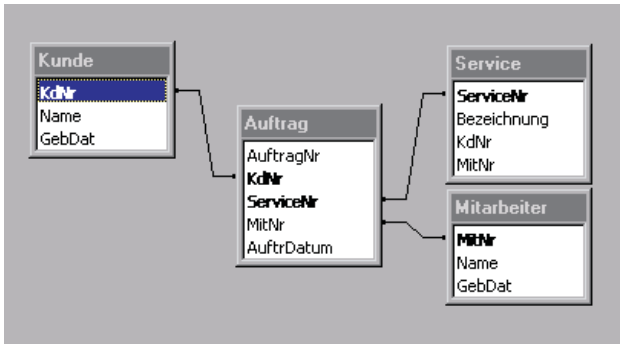

Abbildung 3.11: Abbildung der Entität Auftrag in eine Relation

Ebenfalls nicht direkt vorgesehen ist das Konzept der Generalisierung. Bei Generalisierungsbeziehungen werden daher im Allgemeinen nur Untertypen, die nicht wiederum Obertypen sind, in Relationen überführt. Diese Untertypen übernehmen die Attribute von ihren Obertypen.

Die Generalisierung lässt sich aber durch Projektion mehrerer Relationen auf gemeinsame Attribute, gefolgt von einer Vereinigung, operativ ausdrücken, was im Rahmen der Definition von externen Schemata durch Operatoren der relationalen Algebra zur Realisierung von »virtuellen« Generalisierungsrelationen ausgenutzt werden kann.

3.3 Optimieren des Datenbankentwurfs durch Anwendung der Normalformen

Vielfältige Untersuchungen zum relationalen Modell haben eine eigene Datenbanktheorie hervorgebracht, die sich unter anderem mit den Abhängigkeiten innerhalb der Tabellen beschäftigt. Diese Untersuchungen tragen dazu bei, Redundanzen und Anomalien aufzudecken oder helfen diese zu vermeiden.

Nach dem Entwurf einer relationalen Datenbank kann mit der Normalisierung Datenredundanz stufenweise reduziert werden. Beim Prozess der Normalisierung wird ein bereits bestehendes Schema mit Regeln, die aus der Relationenalgebra ableitbar sind, optimiert. Die Tabellenstruktur wird dermaßen angepasst, dass die eingangs erwähnten Qualitätsmerkmale auf das Gesamtschema zutreffen.

Jede Tabelle sollte so aufgebaut sein, dass das Hinzufügen oder Löschen einer Zeile (bzw. eines Eintrages in ihr) nicht Informationen verändert oder löscht, die anderwärts noch benötigt werden.

Treten solche Probleme auf, dann spricht man von *Modifikations-Anomalien*, d.h. es kommt zu unerwünschten Folgen, wenn Daten in einer Tabelle hinzugefügt, geändert

oder gelöscht werden. Solche Anomalien treten dann auf, wenn eine Tabelle Einträge enthält, die sinnvollerweise auf zwei oder mehr Tabellen verteilt werden sollten. Das Aufspalten einer Tabelle in zwei oder mehrere Tabellen, um solche Anomalien zu beseitigen, nennt man *Normalisierung* (Normalization).

Zwischen 1970 und 1981 betrieb man Forschung, wie man Modifikations-Anomalien vermeiden kann. Je nach Quelle der Modifikations-Anomalie, die man auschließt, unterscheidet man seither verschiedene Normalformen:

- Erste Normalform (first normal form) (1NF),
- Zweite Normalform (second normal form) (2NF),
- Dritte Normalform (third normal form) (3NF).

Es gelten zunächst einige Grundbedingungen, die wie folgt zusammengefasst werden können:

- Tabellen sind zweidimensional mit Reihen und Spalten,
- jede Reihe enthält Daten, die zu einem Objekt oder einem Teil eines Objektes gehören,
- jede Spalte muss einen (in der Tabelle) einmaligen Namen tragen,
- keine zwei Reihen dürfen identisch sein,
- die Reihenfolge der Spalten und Reihen ist bedeutungslos.

3.3.1 Erste Normalform

Regel: *Eine Tabelle ist in der ersten Normalform, wenn sie den Grundbedingungen entspricht und die Einträge der Tabellenspalten einwertig (atomar) sind.*

Tabelle: Mitarbeiter

MitarbNr	Adresse
1001	Schmidt, Jochen, Wienerweg 3, Hamburg
1002	Meyer, Heidi, Amselstieg 12, Berlin
1003	Ehrhardt, Klaus, Kölnerweg 11, Berlin
⋮	⋮

Abbildung 3.12: Die Tabelle Mitarbeiter

Abbildung 3.12 zeigt uns eine Tabelle, die nicht dieser Regel entspricht. Die Spalte *Adresse* hat keine einwertigen Einträge. Mehrere Informationen, wie Name, Straße und Ort sind in einer Spalte zusammengefasst. Die nächste Abbildung zeigt uns, wie ein Entwurfsschema einer Tabellenstruktur in der ersten Normalform aussehen müsste.

Tabelle: Mitarbeiter

MitarbNr	AbtNr	Int_Ext	AdrNr
1001	1	I	221
1002	1	E	222
1003	2	I	223
⋮			

Tabelle: Adresse

AdrNr	Name	Vorname	Straße	Hnr	Ort
221	Schmidt	Jochen	Wiener Weg	3	Hamburg
222	Meyer	Heidi	Amselstieg	12	Berlin
223	Ehrhardt	Klaus	Kölnerweg	11	Berlin
⋮					

Abbildung 3.13: Tabellen Mitarbeiter und Adresse in der ersten Normalform

Die Adressinformationen sind in eine separaten Tabelle eingegangen. In beiden Tabellen gibt es nun ausschließlich Spalten, die einwertige Einträge vorweisen.

3.3.2 Zweite Normalform

Regel: *Eine Tabelle genügt der zweiten Normalform, wenn sie der ersten Normalform entspricht und Nicht-Schlüsselattribute voll funktional abhängig vom Schlüssel sind, und nicht nur von einer Teilmenge des Schlüssels.*

In der zweiten Normalform hängen alle Spalten (Attribute), die nicht zum Primärschlüssel gehören, vom Primärschlüssel als Ganzem ab. Alle Tabellen in der ersten Normalform, die einen einfachen (aus nur einer Spalte bestehenden) Primärschlüssel haben, sind so automatisch auch in der zweiten Normalform. Ist aber der Primärschlüssel zusammengesetzt, ist für die zweite Normalform zu prüfen, ob es Attribute gibt, die in Wirklichkeit nur von einer oder einem Teil der Spalten des Primärschlüssels abhängen. Ist dies der Fall, ist mit diesem Teil des Primärschlüssels eine neue Tabelle zu bilden und die ursprüngliche Tabelle in zwei Tabellen aufzuteilen.

Wir sehen uns die Abbildung 3.14 an. In der Tabelle *Mitarbeiter* bilden die Spalten *MitarbNr* und *AbtNr* einen zusammengesetzten primären Schlüssel. Die Spalte *Abteilung* ist nicht von beiden Schlüsselspalten, sondern nur von der *AbtNr* funktional abhängig. Damit ist sie nur von einer Teilmenge des Primärschlüssels abhängig und verstößt gegen die Regel der zweiten Normalform. Gibt es keinen zusammengesetzten Primärenschlüssel, dann ist es auch nicht erforderlich, die zweite Normalform anzuwenden. Im Beispiel aus Abbildung 3.14 wäre es jedoch erforderlich, erneut eine Aufteilung der Daten vorzunehmen. Zu diesem Zweck werden die Informationen über die Abteilung inklusive der Abteilungsnummer in einer neuen Tabelle *Abteilung* übertragen. Das

Ergebnis sehen wir in der Abbildung 3.15. Die Tabelle *Mitarbeiter* hat noch die Spalte *AbtNr*, die hier den Fremdschlüssel darstellt. Über diesen Fremdschlüssel kann eine Verknüpfung mit der Tabelle *Abteilung* erfolgen. Die Tabelle *Abteilung* kann ihrerseits weitere Attribute über den hier angegebenen *Namen* hinaus beinhalten.

Tabelle: Mitarbeiter

MitarbNr	AbtNr	Abteilung	Int_Ext
1001	1	Service	I
1002	1	Service	E
1003	2	Buchhaltung	I
⋮			

Abbildung 3.14: Tabelle Mitarbeiter

Tabelle: Abteilung **Tabelle: Mitarbeiter**

AbtNr	Name
1	Service
2	Buchhaltung
3	Marketing

MitarbNr	AbtNr	Int_Ext	AdrNr
1001	1	I	221
1002	1	E	222
1003	2	I	223
1004	3	E	224
1005	3	E	225

Abbildung 3.15: Die Tabellen Abteilung und Mitarbeiter in der zweiten Normalform

3.3.3 Dritte Normalform

Regel: *Eine Tabelle genügt der dritten Normalform, wenn sie der zweiten Normalform entspricht und jedes Nicht-Schlüsselattribut nicht-transitiv ist, d.h. es hängt (nur) direkt vom Primärschlüssel ab.*

Eine Tabelle in der dritten Normalform enthält keine *transitiven Abhängigkeiten*, d.h. Abhängigkeiten der Form, dass ein Attribut *A1* von einem anderen Attribut *A2* abhängt, welches wieder von einem anderen Attribut *A3* abhängt. Solche Abhängigkeiten können bewirken, dass das Löschen einer Reihe mit einem Eintrag von *A3* über *A2* zu einem Informationsverlust für *A1* führt.

Die Tabelle *Mitarbeiter* soll auch für diese Normalform als Beispiel dienen. Diesmal wird ein Teil der Adresse direkt dem Mitarbeiter zugewiesen. Die Adressnummer und die Ortsangabe sind Bestandteil dieser Tabelle. Weitere Angaben wie Straße, Hausnummer und PLZ sind aus Gründen der Übersichtlichkeit weggelassen. Die Spalte mit der Ortsangabe ist in diesem Fall nicht direkt vom Primärschlüssel (*MitarbNr*), sondern von der Spalte *AdrNr* abhängig. Es ist möglich, dass ein Mitarbeiter mehrere Wohnsitze hat. In der Tabellenstruktur aus Abbildung 3.16, wäre es nur auf Kosten von Redundanz möglich, dem Mitarbeiter mehrere Adressen zuzuordnen. So müsste für jeden Wohnsitz ein Datensatz in der Tabelle vorhanden sein, der dann jedesmal die Attribute *AbtNr* sowie *Int_Ext* mitschleppen würde, die nur einmal pro Mitarbeiter erforderlich sind.

Tabelle: Universal

MitarbNr	Name	Ort	Int_Ext	AbtNr	Abteilung
1001	Schmidt	Hamburg	I	1	Service
1002	Meyer	Berlin	E	1	Buchhaltung
1003	Ehrhardt	Berlin	I	2	Marketing
1003	Ehrhardt	Hamburg	I	2	Marketing

Abbildung 3.16: Universaltabelle

Die Lösung ist auch hier wieder eine Aufteilung in mehrere Tabellen. In Bild 3.13 sind bereits die Tabellen *Mitarbeiter* und *Adresse* verknüpft abgebildet. Diese Tabellenstruktur würde den Bedingungen der dritten Normalform genügen. Man spricht dann auch davon, dass die Tabelle(n) sich in der dritten Normalform befinden.

3.3.4 Weitergehende Normalformen

Über die bisher dargestellten Normalformen hinaus, die sich mit der Eliminierung funktionaler Abhängigkeiten befassen, existieren weitergehende Normalformen (vierte und fünfte Normalform) zum Ausschluss mehrwertiger Abhängigkeiten und sogenannter *Join Dependencies*, die aber in der Praxis kaum eine Rolle spielen. Sie werden deshalb an dieser Stelle nicht weiter behandelt. Selbst die ersten drei Normalformen können u.U. umgangen werden, wenn es aufgrund von Leistungskriterien sinnvoll ist. So können beispielsweise redundante Daten gezielt in eine Tabelle erzeugt werden, um über diese Zwischenwerte zu erhalten.

3.4 Physischer Entwurf

Bei einem physischen Entwurf geht es darum, das bereits erarbeitete logische Konzept an die gegebenen physischen Voraussetzungen anzupassen. Speicherstrukturen sowie Zugriffsmechanismen sind aufgrund der Datenbankarchitektur zu entwerfen. Wichtiger Faktor ist neben der geforderten Funktionalität der Versuch die Zugriffszeiten zu minimieren. Folgende Überlegungen sollten deshalb zu einem physischen Entwurf führen, der schließlich in der Testphase optimiert und in der Produktivphase »getunet« werden kann:

- Definition des internen Schemas,
- Verwendung der Dateiformate,
- Block- und Seitenzuweisung auf Platte,
- Indexstrukturen,
- Verteilung von Datenobjekten auf mehrere Platten (Clustering),
- Grundlagen des physischen Datenbankentwurfs,
- Fragmentierung in verteilten Datenbanken,
- Datenbank-Benchmarks,
- Festlegung von Speicherungsstrukturen und Zugriffsmechanismen,
- logische Speicherorganisation.

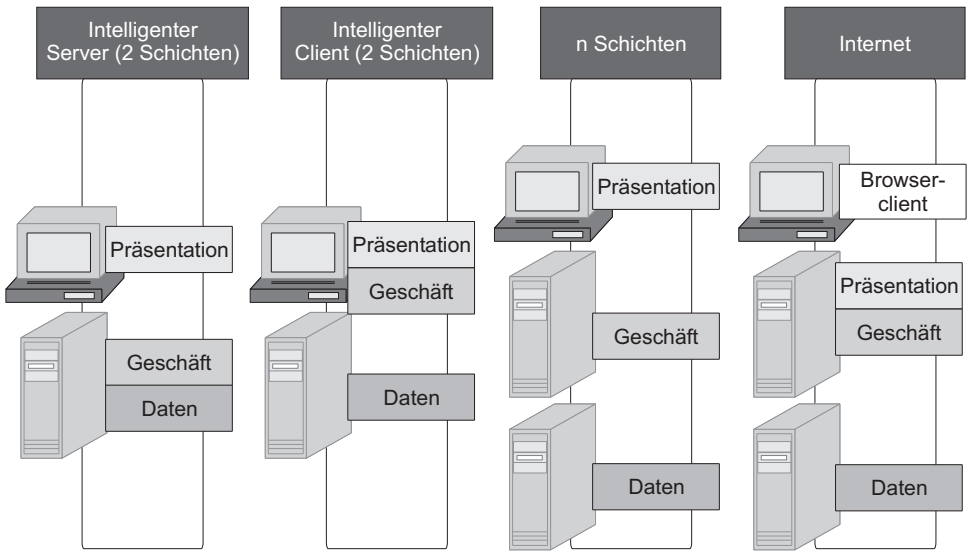

Abbildung 3.17: Systementwürfe für die Datenbankanwendung

Die Kenntnis der Geschäftsabläufe und die gegebenen Hardwarevoraussetzungen ergeben die nötigen Informationen für die Datenbankkonzepte. Die Abbildung 3.17 zeigt vier Entwürfe für eine Datenbankanwendung.

Beim »Intelligenten Server« trägt der Server die größte Last an Verarbeitungsprozessen. Die Geschäftslogik ist zum Großteil in der Datenbank implementiert. Haben die Client-Rechner nicht ausreichende Ressourcen, dann eignet sich dieser Entwurf für das System.

Bei der Lösung mit dem »Intelligenten Client« erfolgt ein Großteil der Verarbeitung auf den Client-Rechnern. Dadurch wird der Server entlastet, aber gleichzeitig steigt die Belastung des Netzwerkes. Transaktionen können so unter Umständen längere Ausführungszeiten haben.

Eine komplexere Lösung stellt der Entwurf mit n Schichten dar. Datenbank und Anwendungslogik sind hierbei auf getrennten Servern untergebracht. Bei komplexen Anwendungen wird somit die Last verteilt, was bei kleineren Anwendungen jedoch zu Leistungsverschlechterungen führen könnte.

Sollten die Informationen über einen Browser im Internet oder Intranet abgerufen werden, dann bietet sich die vierte Lösung mit einem Web-Server an.

3.5 Datenbankdiagramme bringen Struktur in den Entwurf

Komplexe Sachverhalte, die der Datenbankprogrammierer in der Praxis vorfindet, erfordern es, dass er die einzelnen Schritte seiner Arbeit dokumentiert. Besonders für Mitarbeiter, die nicht direkt an der Entwicklung beteiligt waren, ist es oftmals überhaupt nicht überschaubar, welche Tabelle nun welche Daten beinhaltet und welches Objekt aus der Realität abgebildet wird. Manchmal ist es einfacher, ein System neu zu implementieren, anstatt sich in ein schlecht dokumentiertes System einzuarbeiten. Einige wenige Diagramme sagen oftmals mehr als viele Worte. Zumindest ein Diagramm, welches die Abhängigkeiten der einzelnen Tabellen aufzeigt, sollte als Minimalbeschreibung vorhanden sein. Ein solches Diagramm liefert uns, wie bereits gesehen, der Entwurfprozess nach dem ER-Modell. Für objektorientierte Systeme sind das *Generische Semantische Modell* (GSM) oder das *Semantische Datenmodell* (SDM) Beschreibungsmöglichkeiten mit grafischer Unterstützung. Aber nicht nur für die Lesbarkeit nach der Entwicklung sind Diagramme erforderlich. Das Phasenmodell der Softwareentwicklung in Kapitel 2 verlangt nach der Entwurfsphase eine Spezifikation. Üblicherweise sind Diagramme Bestandteile der Spezifikation bzw. sie bilden die Grundlage für die Spezifikation. An dieser Stelle ist es also erforderlich, unseren konzeptionellen Entwurf, den wir bisher ansatzweise vollzogen haben, zu vervollständigen, um aus ihm eine Spezifikation für die Tabellenstruktur abzuleiten.

Das erste Teildiagramm für das Dienstleistungsunternehmen haben wir bereits in Abbildung 3.6 vorliegen. Hier wird die Beziehung zwischen Mitarbeiter, Service und Kunde beschrieben. In Abbildung 3.8 sind wir bereits einen Schritt weiter und haben durch die Aggregation aus der Beauftragung eine aggregierte Entität, nämlich den Auftrag erhalten. Ein weiteres Teildiagramm zeigt die folgende Abbildung.

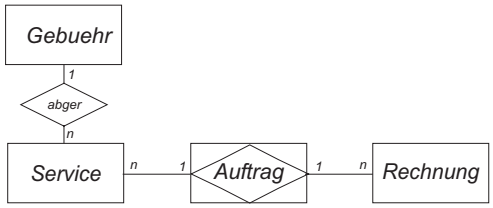

Abbildung 3.18: Darstellung der Abhängigkeiten von Service, Rechnung und Auftrag

Jeder beauftragte Service wird in Rechnung gestellt. Dabei wird jeder Service nach eine Gebührenordnung abgerechnet. Jede Rechnung wird genau einem Auftrag zugewiesen. Andrerseits können für jeden Auftrag mehrere Rechnungen erstellt werden. Daher haben wir zwischen Auftrag und Rechnung eine *1:n*-Beziehung. Die Dienstleistungen (Service) haben eine eindeutig abzurechnende Gebühr. Diese Gebühr kann allerdings auch als Abrechnungsgrundlage für andere Leistungen herangezogen werden. Daraus ergibt sich eine *1:n*-Beziehung auch für die Entitäten *Gebühr* und *Service*. Die *Leistungen* sollen jeweils genau einem *Auftrag* unterstellt sein. Auch hieraus ergibt sich eine *1:n*-Beziehung bei Service und Auftrag. Die Attribute sind in diesem Teildiagramm, wie auch in den anderen, nicht berücksichtigt.

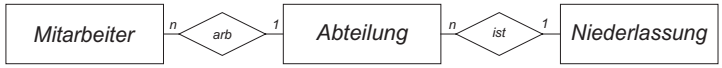

Abbildung 3.19: Abhängigkeiten zwischen Mitarbeiter, Abteilung und Niederlassung

Die Abbildung 3.19 bringt eine Übersicht der Abhängigkeiten zwischen *Mitarbeiter*, *Abteilung* und *Niederlassung*. Jeder Mitarbeiter arbeitet in ausschließlich einer Abteilung und diese wiederum wird eindeutig einer Niederlassung zugeordnet.

In dem Teildiagramm in Bild 3.20 wird dem Kunden eine Adresse und Bankverbindung zugewiesen. Die Adresse wird erweitert durch Angaben der Telekommunikation in der Entität *Telekom* sowie Angaben zum Land durch die Entität *Land*.

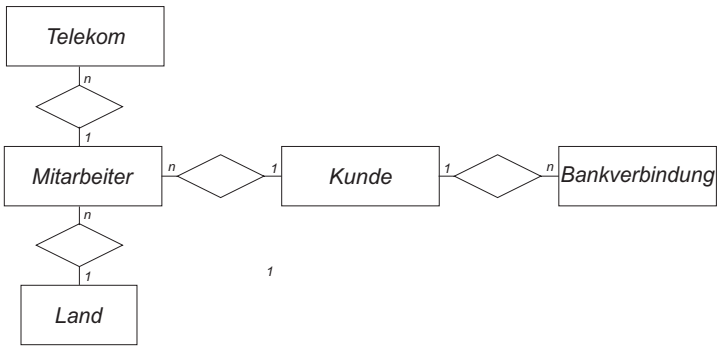

Abbildung 3.20: Teildiagramm mit Bankverbindung

Bei strukturell aufwändigen Entwürfen ist es üblich, das DB-Diagramm in mehrere Teildiagramme aufzuteilen. Für eine Gesamtübersicht ist es dann ratsam, auf Details zu verzichten. So kann in einem Gesamtdiagramm, wie in Abbildung 3.21, auf Attribute und Relationships verzichtet werden.

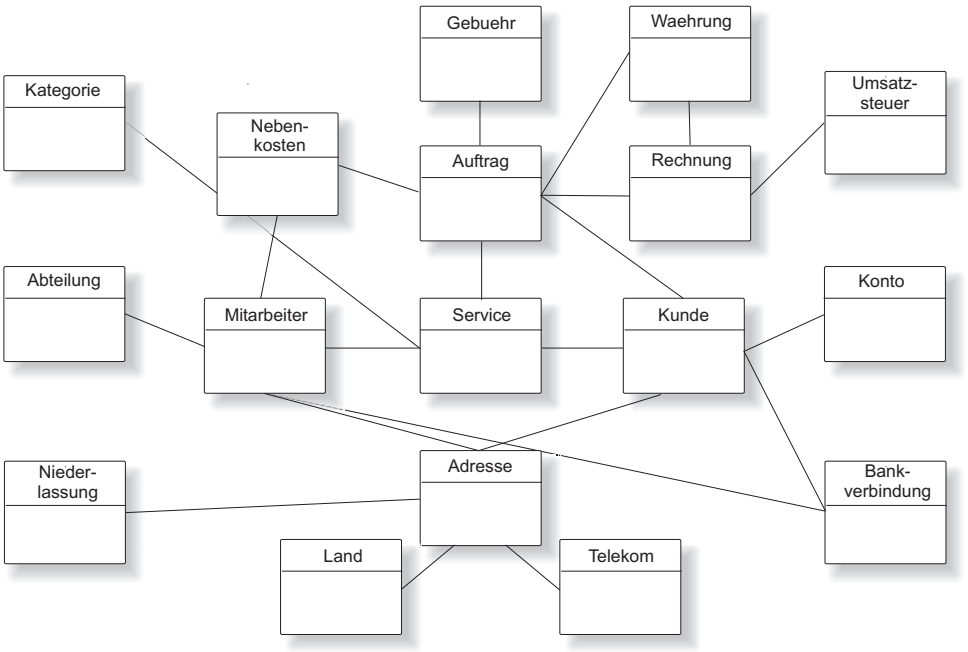

Abbildung 3.21: Gesamtübersicht für das Dienstleistungsunternehmen

In der Abbildung 3.21 sind die Teildiagramme zusammengefasst und um einige untergeordnete Entitäten erweitert. Die Entität *Konto* beispielsweise beinhaltet das Gutha-

ben bzw. den zu zahlenden Betrag des Kunden an das Unternehmen. Dies hat nichts mit dem Konto bei der entsprechenden Bank zu tun. Die Angaben zum Bankkonto gehören zur Entität *Bankverbindung*. Für die Erstellung der Rechnung ist die Information über Umsatzsteuer und auch Währung notwendig, deshalb sind hierfür zwei weitere Entitäten vorgesehen. Diese Angaben sind auch abhängig von den Vereinbarungen im Auftrag (deshalb besteht auch eine Abhängigkeit zwischen Auftrag und Währung).

Jeder Service gehört zu einer Kategorie, die sich auch in der Gebührenverordnung widerspiegelt, aus einem vom Unternehmen angebotenen Katalog an Leistungen. Also muss auch dieser Umstand mit einer weiteren Entität berücksichtigt werden. Dem Unternehmen entstehen nicht nur Kosten, die sich direkt aus der Arbeitszeit für die beauftragte Leistung ergeben, sondern Reisekosten und Materialkosten müssen auch an den Kunden weitergereicht werden, deshalb wird die Entität *Nebenkosten* ebenso in das Konzept aufgenommen.

3.6 Werkzeuge für den Entwurf

Für den Entwurf oder auch für das Erstellen von Diagrammen und Grafiken gibt es von verschiedenen Anbietern Werkzeuge, welche die Arbeit erleichtern.

Auf der Begleit-CD finden Sie das Programm ERM. Dies ist ein Werkzeug, mit dem Sie Ihre ER-Modelle erstellen können.

3.6.1 MS Access

Datenbankdiagramme können auch in MS Access nach dem Entwurf erstellt werden. Wenn Sie auf EXTRAS/BEZIEHUNGEN... klicken, dann wird das Fenster BEZIEHUNGEN geöffnet (siehe Abbildung 3.22).

Dort haben Sie durch Klicken auf die Schaltfläche TABELLE ANZEIGEN, die Möglichkeit, die zuvor entworfenen Tabellen im Diagramm einzufügen. Wählen Sie eine oder auch mehrere Tabellen aus und klicken Sie dann auf die Befehlsschaltfläche HINZUFÜGEN, um die Tabellen einzufügen. (siehe Abbildung 3.23)

Schließlich kommen Sie durch Positionieren der einzelnen Tabellen und Erzeugung der Beziehungen zu einem Ergebnis, welches in Abbildung 3.23 als Teilstruktur genügen soll.

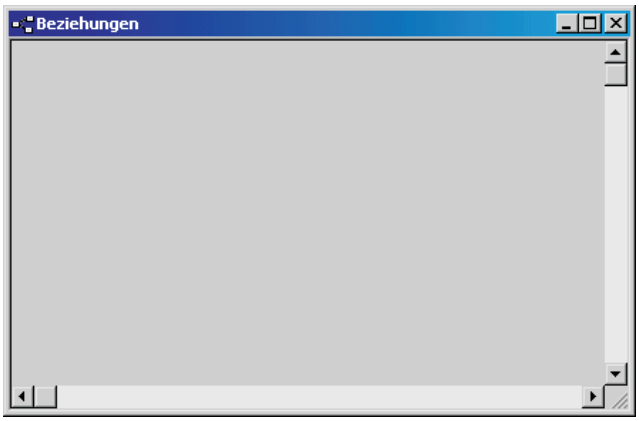

Abbildung 3.22: Das Fenster »Beziehungen« in MS Access

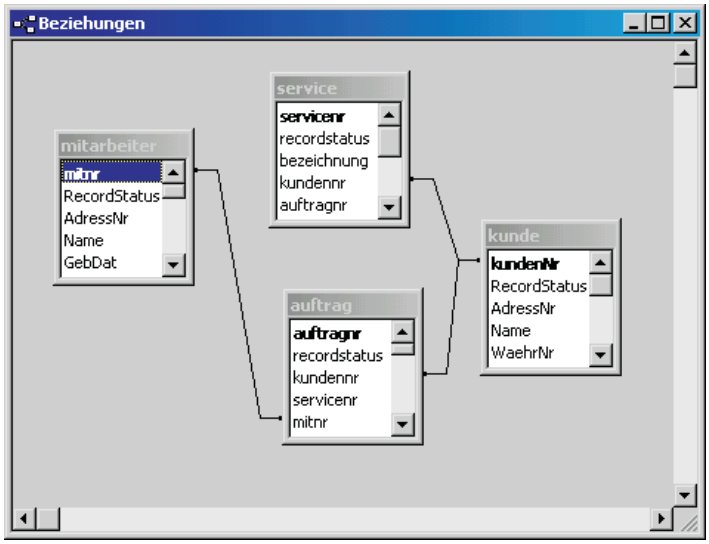

Abbildung 3.23: Beziehungen zwischen Access-Tabellen

3.6.2 MS Visio

Microsoft Visio ist ein nützliches Werkzeug für die Diagrammerstellung. Mit Visio können Sie die vielfältigen, täglich anfallenden Aufgaben klar umreißen und dokumentieren, anderen Ihre Ideen mitteilen und Informationen gemeinsam nutzen. Mit Visio stellen Sie diese Informationen in einer Form dar, die leicht zu erfassen ist und auch im Gedächtnis bleibt. In Visio Standard erstellte Diagramme ermöglichen wertvolle Einblicke in bestehende Prozesse, Projekte und Informationen zu Mitarbeitern

und helfen einzelnen Mitarbeitern sowie Teams auf diese Weise bei einer effizienteren Zusammenarbeit. Und wenn Sie Visio-Diagramme in Ihre Office-Dokumente integrieren, werden Ihre Mitteilungen noch prägnanter, die wichtigsten Punkte können noch besser hervorgehoben werden, und kulturelle Unterschiede und technische Hindernisse können leichter überwunden werden. Abbildung 3.24 zeigt zwei Diagramme, die mit Visio erstellt wurden.

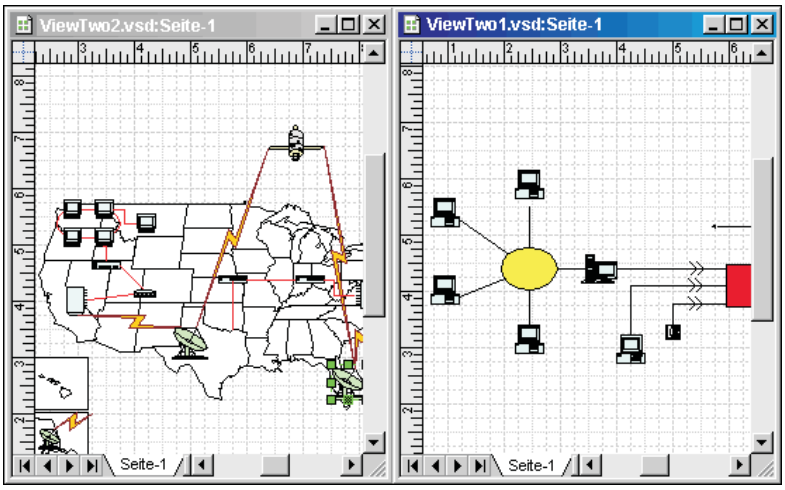

Abbildung 3.24: Diagramme in Visio

3.7 Vom Entwurf zur Spezifikation

Nach Fertigstellung des Entwurfes werden die Ergebnisse in die Erstellung der Spezifikation eingebracht. Die Spezifikation wird auch häufig als Feinentwurf bezeichnet. Komponenten für Rahmenwerke, die Berücksichtigung von Standards und die Unabhängigkeit gegenüber proprietären Umgebungen erfordern eine genaue Spezifikation der zu erstellenden Software. Aus dieser abstrakten Spezifikation wird eine Implementierung für die jeweilige Umgebung (Sprache, Rahmenwerke, etc.) erstellt. Dieser Schritt gehört zu den fehlerträchtigsten Tätigkeiten beim Erstellen von Software. Das liegt daran, dass bei der Umsetzung des Entwurfs nicht nur das Wissen um die zu verwendende Umgebung, sondern auch das Fachwissen aus dem Problembereich vorhanden sein muss. Sollen dabei sogar Konzepte wie z.B. Entwurfsmuster oder Optimierungen eingesetzt werden, steigen die Anforderungen an den Implementierungsschritt nochmals erheblich.

Es wurden Techniken evaluiert und ein System entworfen, das eine automatische Umsetzung abstrakter Spezifikationen ermöglicht. Mit Hilfe eines Graphenregelsys-

tems (GRS) wird die Software-Spezifikation, die als UML-Diagramm (Unified Modeling Language) formuliert werden kann, auf die Implementierungsebene abgebildet. Dazu wird eine Regelbasis verwendet, die die Abbildung bestimmter Konzepte beschreibt. Diese Regelbasis kann ebenfalls als UML-Graph beschrieben werden, was die Anpassung und Erweiterung der Regeln vereinfacht. Ein sich daraus ergebendes Problem ist die Synchronisation der abstrakten Spezifikation und der dazugehörigen Implementierung. Die Erfahrung zeigt, dass Änderungen an der Software nicht nur auf der Ebene der Spezifikation, sondern auch direkt in der Implementierung erfolgen. Um sicherzustellen, dass die Spezifikation immer zur aktuellen Implementierung passt, wird derzeit ein Verfahren entwickelt, das Änderungen an der Implementierung auf der abstrakten Ebene mitführt.

3.8 Zusammenfassung

Bei der Entwicklung einer Datenbank wird das Ziel angestrebt, einen Ausschnitt der realen Welt abzubilden. In diesem kleinen Weltausschnitt existieren eine Vielzahl von Objekten, wie Personen, Unternehmungen, Produkte oder Dienstleistungen. Diese stehen in unterschiedlichen Beziehungen zueinander.

Im ersten Schritt werden die Objekte der realen Welt, die für die Aufgabenstellung relevant sind, mit ihren Beziehungen untereinander in abstrakter Weise beschrieben, d.h. modelliert. Um diesen Vorgang visuell zu unterstützen, werden Datenmodelle mit Hilfe des ER-Modells erstellt.

Damit die abstrakten Objekte und Beziehungen noch deutlicher werden, wird dieser Modellansatz durch ein ER-Diagramm unterstützt.

Im zweiten Schritt wird aus der Datenmodellierung das logische Datenbankmodell erstellt. Man spricht hier auch von der sogenannten konzeptionellen Phase. Auf welches Datenbankmodell man dabei abzielt, liegt daran, welches Datenbankverwaltungssystem später zum Einsatz kommen soll.

Bei der Konzeptionierung des logischen Datenbankmodells gibt es schon eine Ausrichtung auf das Datenbankverwaltungssystem, das man später verwenden will.

Im dritten Schritt wird das logische Datenbankmodell dann in der Datenbeschreibungssprache des Zielsystems umgesetzt. Hierbei werden Tabellen angelegt, wobei durch die Felddefinition die Datenbeschreibung erfolgt.

Nach dem Entwurf einer relationalen Datenbank kann man mit der Normalisierung Datenredundanz stufenweise reduzieren. Beim Prozess der Normalisierung wird ein bereits bestehendes Schema durch Anwendung einiger Regeln optimiert.

Alle Normalformen sind von großer theoretischer Bedeutung. In der Praxis beschränkt man sich jedoch auf die ersten drei.

Interessant wird die Kombination aus semantischem Datenmodell und logischem Datenbankmodell, wenn bei der Modellierung mit dem Entity-Relationship Modell eine saubere Definition eines Entitäten-Beziehungsmodells erfolgt. Werden hierbei nämlich die Abbildungsregeln beachtet, dann sind die Normalformen jederzeit automatisch erfüllt.

Dieser Modellierungsansatz gewährleistet uns redundanzfreie, gegen Anomalien gesicherte Datenbankmodelle. Die Implementierung, d.h. die Realisierung solcher Modelle lässt sich so leichter in eine konkrete Datenbank umsetzen.