DATENBANK-DESIGN – EINFÜHRUNG

Inhaltsverzeichnis

0. Einführung

0

Herzlich Willkommen,

im **Datenbank-Design-Kurs** der Bedag Informatik.
Der Kurs dauert einen Tag und hat das Ziel, Ihnen die Grundkenntnisse einer Datenbank und das Wissen zur Erstellung eines Datenbank-Designs zu vermitteln.
Sie werden in diesem Kurs nicht nur viel Neues erfahren; Sie werden auch Gelegenheit haben, praktische Erfahrungen zu sammeln.

Die Kursunterlagen

Die vorliegenden Unterlagen unterstützen und ergänzen die Erklärungen der Kursleiterin oder des Kursleiters. Das Dokument umfasst drei Teile:

- Im **Theorieteil** sind alle wichtigen Informationen zu einem bestimmten Thema (in sogenannten Lerneinheiten) zusammengestellt. Die einzelnen Lerneinheiten sind dabei in sich abgeschlossen.
- Die **Übungen** heben sich farblich von den übrigen Teilen der Unterlagen ab. Sie werden im Verlauf der verschiedenen Lerneinheiten bearbeitet und geben Ihnen die Möglichkeit, das erworbene Wissen gleich anzuwenden und zu üben. So wird das theoretisch Erworbene in die Praxis transferiert.
- Im **Anhang** sind in kurzer, übersichtlicher Form wichtige Informationen zum Thema Datenbank zusammengefasst.

Die Unterlagen sind so aufgebaut, dass Sie nach dem Kurs als Nachschlagewerk dienen können. Schreiben Sie während dem Kurs Ihre Notizen direkt auf die betreffende Seite. So werden Sie die Unterlagen noch besser und persönlicher als Hilfsmittel nutzen können.

Technische Angaben zu den Kursunterlagen

Die Informationen in den Kursunterlagen sind in Lerneinheiten gegliedert. Zu jeder Lerneinheit gibt es mindestens eine Übung, deren Numerierung jeweils mit jener der Lerneinheit übereinstimmt.

Wichtige Hinweise

Der Kursleiter ist sehr dankbar für Kritiken und Anregungen aller Art. Scheuen Sie sich nicht, den Kursleiter zu fragen, auf etwas hinzuweisen oder sich etwas genauer erklären zu lassen. Falls das Tempo der Kursleitung zu schnell oder zu langsam ist, melden Sie dies bitte.

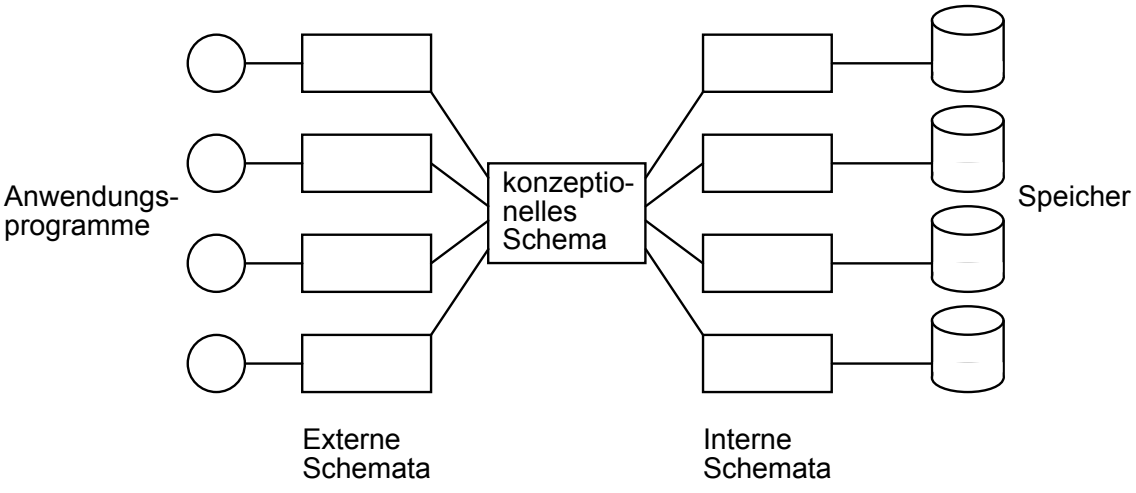
☞ Aus Gründen der sprachlichen Einfachheit verwenden wir in den vorliegenden Kursunterlagen die männliche Form (z.B. Kursteilnehmer, Kursleiter), ohne dabei die Frauen ausschliessen zu wollen. Wir danken für Ihr Verständnis.

1. Einführung in Datenbank-Management-Systeme

1

1.1 Das Dreischichten Modell

Das zentrale Ziel einer Datenbank ist die Integration von Daten und die Datenunabhängigkeit der Anwendungsprogramme. Es setzt voraus, dass sich alle Anwendungsprogramme auf eine einheitliche Beschreibung der Daten einigen. Um diese in der Praxis schwierige Entwurfsaufgabe zu erfüllen, wurde im Jahre 1975 das Dreischichten Modell vorgeschlagen. Die Beschreibung der Daten einer Datenbank erfolgt nach diesem Konzept auf drei verschiedenen Ebenen, die jeweils durch eine andere Sichtweise geprägt sind. Zur Formalisierung einer Sichtweise dienen Schemata. Ein Schema ist eine in einer Datenbeschreibungssprache abgefasste Definition der in einer Datenbank zugelassenen Datenstrukturen.
Für viele Benutzergruppen der Daten (sowohl Programmierer als auch Anwender) genügt ein Ausschnitt eines Gesamtmodells zur Erledigung der mit den Daten verbundenen Aufgaben. Jeder dieser Modellausschnitte muss jedoch aus der logischen Gesamtsicht aller Daten herleitbar sein.



1.2 Die drei Ebenen

1. Logische Gesamtsicht

Auf der **konzeptionellen Ebene** steht die logische Gesamtstruktur der Daten, ihrer Eigenschaften und ihrer Beziehungen untereinander im Vordergrund. Die Beschreibung der konzeptionellen Ebene erfolgt durch das konzeptionelle Schema. Sie berücksichtigt weder die physikalische implementierungsabhängige Organisation der Daten, noch die Wünsche von Anwendungsprogrammen.

2. Physische Sicht

Auf der **internen Ebene** werden die implementierungsabhängigen Eigenschaften der Daten (ihre physikalische Organisation) durch ein internes Schema definiert. Sie beschreibt die Dateiorganisation, die Zugriffsmechanismen und bestimmt im wesentlichen die Leistungsfähigkeit des Datenbanksystems.

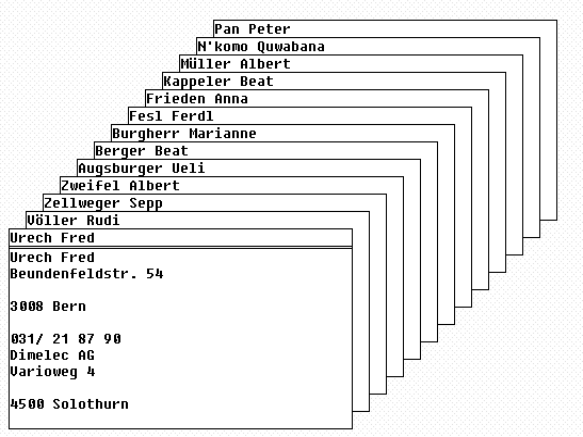
3. Benutzersichten

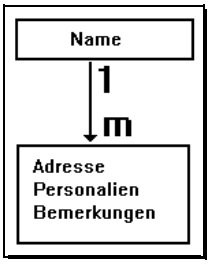
Die Datenbank aus Sicht jedes Anwendungsprogrammes wird auf der **externen Ebene** berücksichtigt und durch externe Schemata beschrieben. Zu jedem Anwendungsprogramm kann es ein eigenes externes Schema geben, das die für das Anwendungsprogramm wichtigen Daten der Datenbank in der Weise definiert, wie sie das Programm als Eingabe erwartet.

Die Trennung der drei Ebenen bezwecken, dass Änderungen in der internen Ebene vorgenommen werden können, ohne dass die konzeptionelle Ebene davon berührt wird. Desgleichen können bestimmte Änderungen an der konzeptionellen Ebene vorgenommen werden, ohne dass bereits existierende Benutzersichten davon berührt werden.

1.3 Das hierarchische Datenmodell

Ein **hierarchisches Datenbankmodell** funktioniert wie der altbekannte Zettelkasten. Sie können nur über den Hauptschlüssel auf die Daten zugreifen, zum Beispiel über den Namen in einer Adresskartei.





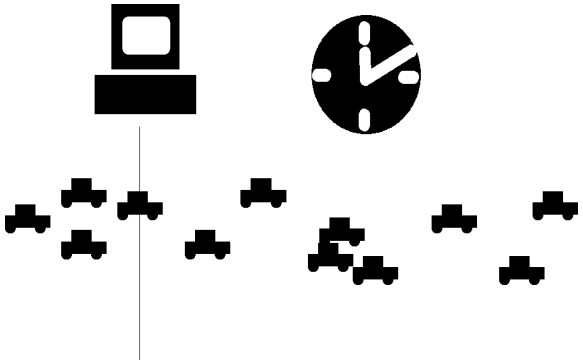
1.4 Das Netzwerkdatenmodell

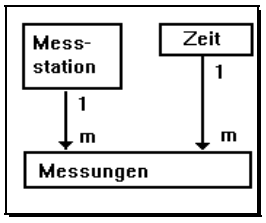
Das Netzwerkmodell wird als Sammelbegriff für alle Datenmodelle benutzt, bei denen die Informationen zwischen Informationsklassen explizit definiert werden müssen. Die einfachste Form eines Netzwerkmodells ist die hierarchische Datenbank (siehe vorhergehendes Kapitel).

Das **Netzwerk** erfasst Beziehungen, die mehrere Hierarchien umfassen. Probleme sind daher vorgegeben, abhängige Daten können nicht mehr eindeutig übergeordneten Elementen zugeordnet werden. Solche Systeme sind sehr komplex und fehleranfällig.

Beispiel eines Netzwerkes.

An einem Messpunkt werden zu verschiedenen Zeiten Messungen vorgenommen.

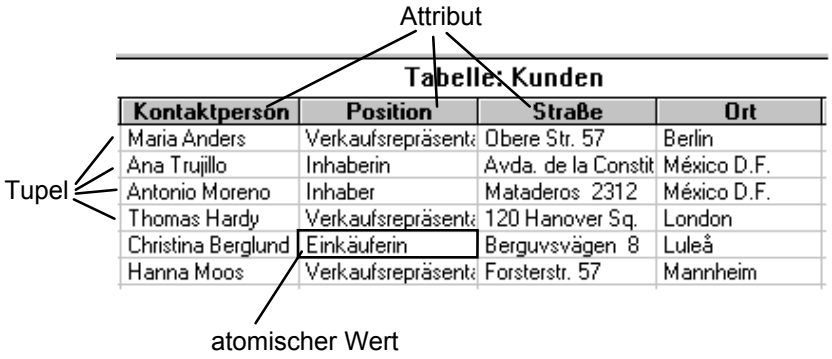




1.5 Das relationale Datenmodell

Das **relationale Modell** wurde 1970 vom Amerikaner E. F. Codd eingeführt. Dieses Modell stellt eine theoretische Grundlage für Datenbanksprachen dar. Die relationale Datenbank besteht aus einigen einfachen Konzepten für das Registrieren von Daten in Datenbanken, sowie aus einer Reihe von Operatoren, mit denen diese Daten bearbeitet werden können:

- Die Daten werden in *Form einer Tabelle*, auch **Relation** genannt, erfasst.
- Durch die Spaltenköpfe (Attribut) wird der *Aufbau der Datenbank* festgelegt.
- In den Spalten stehen die *Eigenschaften* (Attributswerte) *der einzelnen Datensätze* (Tupel).
- Der Schnittpunkt einer Reihe und einer Spalte, das Datenfeld, kann nur *einen Wert* enthalten, den man atomischen Wert nennt.
- In Relationen werden *keine Beziehungen* zwischen den Daten gespeichert (das macht diese Form von Datenbanken so flexibel).

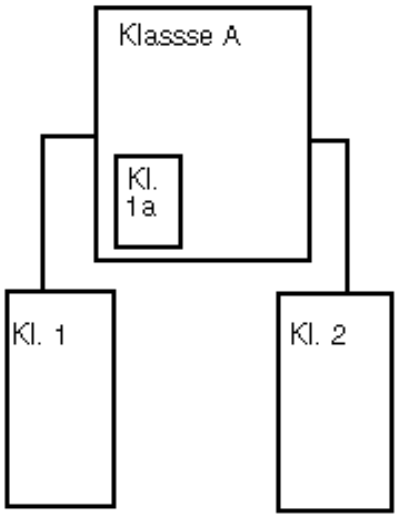


1.6 Das objektorientierte Datenmodell

Die bisher behandelten klassischen Datenmodelle, basieren auf einzelnen Datensätzen und werden deshalb auch satzorientierte Datenmodelle genannt. Neue Ansätze sind durch die objektorientierten Datenmodelle gegeben:

Bei objektorientierten Datenbanken spricht man strenggenommen nicht mehr von Datenmodellen, sondern von Objektmodellen. Der Begriff **Objekt** wird in der Informatik sehr umfassend und allgemein genutzt. Man versteht darunter alle Grössen, die durch einen Bezeichner (Namen) benannt werden können. Das objektorientierte Datenmodell ist eine Weiterentwicklung des Relationenmodells, welche speziell
Bei den objektorientierten Datenmodellen existiert noch keine allgemein anerkannte Definition. Häufig werden bestimmte Eigenschaften aufgestellt, die für sie charakteristisch sein sollen, z.B.:

- Die Typen- bzw. Klassenhierarchie mit entsprechenden Vererbungsmöglichkeiten.
- Zusammengesetzte Objekte, die ausser Attribute (Spalten einer Tabelle) Komponenten besitzen, die selbst wieder Objekte sind.



2. Relationale Grundlagen

2

2.1 Physische und logische Tabellen

Vielen PC-Anwendern, die von der Tabellenkalkulation zur Datenbankwelt kommen, bereitet am Anfang der Unterschied zwischen einer physischen (z.B. eine Excel-Tabelle) und einer logischen Tabelle (z.B. eine Access-Tabelle) einige Verständnisprobleme.

Eine physische Tabelle (z.B. Excel-Tabelle)

C	D	E	F
			26.07.95
	Januar	**Februar**	**März**
Socken	Fr. 12'000.00	Fr. 13'000.00	Fr. 12'870.00
Unterhosen	Fr. 25'370.00	Fr. 24'350.00	Fr. 22'189.00
T-Shirts	Fr. 78'951.00	Fr. 88'125.00	Fr. 101'253.00
Summe:	**Fr. 116'321.00**	**Fr. 125'475.00**	**Fr. 136'312.00**

Ein Tabellenkalkulations-Programm eignet sich sehr gut zur Berechnung von Zahlenmaterial. Zur Berechnung werden die Felder (Zellen) angegeben, welche die gewünschten Daten beinhalten. Übersichthalber ist eine tabellarische Darstellung sinnvoll, aber nicht zwingend.
In verschiedenen Zellen einer Spalte können verschiedene Formate vorhanden sein.

Eine logische Tabelle (z.B. Access-Tabelle)

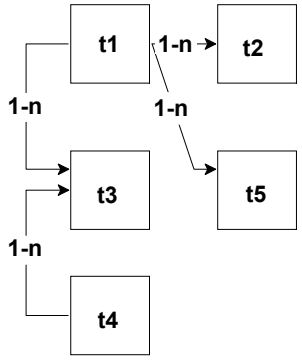
Tabelle: Kunden

Kunden-Code	Firma	Kontaktperson	Position	Straße	Ort
ALFKI	Alfreds Futterkiste	Maria Anders	Verkaufsrepräsenta	Obere Str. 57	Berlin
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Inhaberin	Avda. de la Constit	México D.F.
ANTON	Antonio Moreno Taquería	Antonio Moreno	Inhaber	Mataderos 2312	México D.F.
AROUT	Around the Horn	Thomas Hardy	Verkaufsrepräsenta	120 Hanover Sq.	London
BERGS	Berglunds snabbköp	Christina Berglund	Einkäuferin	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delikatessen	Hanna Moos	Verkaufsrepräsenta	Forsterstr. 57	Mannheim
BLONP	Blondel père et fils	Frédérique Citeaux	Marketingmanager	24, place Kléber	Strasbourg
BOLID	Bólido Comidas preparadas	Martín Sommer	Inhaber	C/ Araquil, 67	Madrid
BONAP	Bon app'	Laurence Lebihan	Inhaber	12, rue des Bouch	Marseille
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Buchhalterin	23 Tsawassen Blv	Tsawassen
BSBEV	B's Beverages	Victoria Ashworth	Verkaufsrepräsenta	Fauntleroy Circus	London
CACTU	Cactus Comidas para llevar	Patricio Simpson	Verkaufsberater	Cerrito 333	Buenos Aires
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketingmanager	Sierras de Granad	México D.F.

Diese Access-Tabelle speichert z.B. verschiedene Informationen über Kunden. Die Firmennamen, Kontaktpersonen etc. sind untereinander in einer eigenen Spalte angeordnet.
Eigenschaften oder Formate sind für alle Einträge in einer Spalte zwingend, egal ob es nun 100 oder 150'000 Felder sind. Alle Einträge in einer Zeile gehören zusammen, sie bilden einen Datensatz.

2.2 Das relationale Modell

E.F. Codd war der führende Kopf in den siebzieger Jahren bei der Entwicklung des bisher erfolgreichsten Datenbanksystems, dem **Relationenmodell oder dem relationalen Modell**.



Das relationale Modell besteht aus der Definition von Objekten, Operationen und Regeln. Die Operatoren definieren eine relationale Algebra, mit der die Objekte bearbeitet werden können.

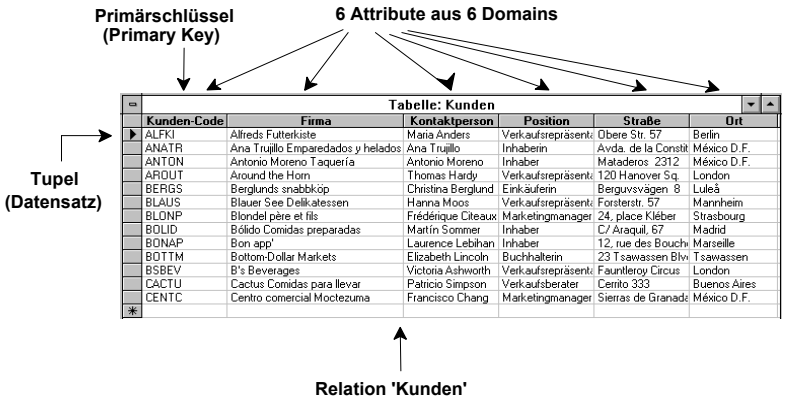


Tabelle: Kunden

Kunden-Code	Firma	Kontaktperson	Position	Straße	Ort
ALFKI	Alfreds Futterkiste	Maria Anders	Verkaufsrepräsentä	Obere Str. 57	Berlin
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Inhaberin	Avda. de la Constit	México D.F.
ANTON	Antonio Moreno Taquería	Antonio Moreno	Inhaber	Mataderos 2312	México D.F.
AROUT	Around the Horn	Thomas Hardy	Verkaufsrepräsentä	120 Hanover Sq.	London
BERGS	Berglunds snabbköp	Christina Berglund	Einkäuferin	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delikatessen	Hanna Moos	Verkaufsrepräsentä	Forsterstr. 57	Mannheim
BLONP	Blondel père et fils	Frédérique Citeaux	Marketingmanager	24, place Kléber	Strasbourg
BOLID	Bólido Comidas preparadas	Martín Sommer	Inhaber	C/ Araquil, 67	Madrid
BONAP	Bon app'	Laurence Lebihan	Inhaber	12, rue des Bouch	Marseille
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Buchhalterin	23 Tsawassen Blv	Tsawassen
BSBEV	B's Beverages	Victoria Ashworth	Verkaufsrepräsentä	Fauntleroy Circus	London
CACTU	Cactus Comidas para llevar	Patricio Simpson	Verkaufsberater	Cerrito 333	Buenos Aires
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketingmanager	Sierras de Granada	México D.F.

Relationale Objekte

Englische Begriffe	*Deutsche Begriffe*
Domain	Wertebereich
Relation	Tabelle
Degree	Ausdehnungsgrad der Tabelle
Attribut	Spalte
Tuple	Datensatz, Rekord
Candidate-Key	eindeutiger Schlüssel
Primary-Key	Haupt- oder Primärschlüssel
Alternate-Key	Zweitschlüssel
Foreign-Key	Fremdschlüssel

Relationale Integritätsregeln

Entity-Integrität
Referentielle Integrität

Relationale Operationen

Englische Begriffe	*Deutsche Begriffe*
Restriction	Zeilenselektion
Projection	Spaltenselektion
Product	Kartesisches Produkt
Union	Vereinigung
Intersection	Schnittmenge
Difference	Differenz
Join	Verbindung
Division	Division

2.3 Relationale Elemente

Domain

Unter Domain wird der Wertebereich eines Attributs verstanden. So besteht die Domain 'Kontaktperson' aus den Werten 'Maria Anders', 'Ana Trujillo', 'Antonio Moreno' und anderen. Bei der Definition eines Attributes (Spalte) wird auf die Domain Bezug genommen.
Domains sind relationenübergreifend definiert. Auf sie kann in jeder Relation Bezug genommen werden.

Kontaktperson
Maria Anders
Ana Trujillo
Antonio Moreno
Thomas Hardy
Christina Berglund
Hanna Moos
Frédérique Citeaux
Martín Sommer
Laurence Lebihan
Elizabeth Lincoln

Relation

Eine Relation (Tabelle) ist eine logisch zusammenhängende Einheit von Informationen. Sie besteht aus einer festen Anzahl von Attributen (Spalten) und einer variablen Anzahl Tupel (Datensätzen).

Eine Relation hat folgende Eigenschaften:

keine doppelten Tupel
: Es gibt zu keinem Zeitpunkt zwei Tupel, deren Attributwerte den gleichen Inhalt haben.

Tupelreihenfolge
: Die Reihenfolge, mit der die Tupel in einer Relation gespeichert sind, ist nicht definiert.

Attributreihenfolge
: Die Reihenfolge der Attribute in einer Relation ist nicht definiert. Es ist also nicht möglich, das n-te Attribut einer Relation anzusprechen. Man kann ein bestimmtes Attribut nur mit seinem Namen ansprechen.

Attributwerte sind atomar

Candidate Key (eindeutiger Schlüssel)

Eine Attributmenge wird Candidate-Key genannt, wenn alle Werte dieser Attributmenge eindeutig sind.

Primary-Key (Haupt- oder Primärschlüssel)

Jede Relation besitzt genau einen Primary-Key (Primärschlüssel), um ein Tupel der Relation eindeutig zu identifizieren. Er wird definiert, indem einer der Candidate-Keys zum Primary-Key erklärt wird. Ein Primary-Key darf keine Nullwerte enthalten.

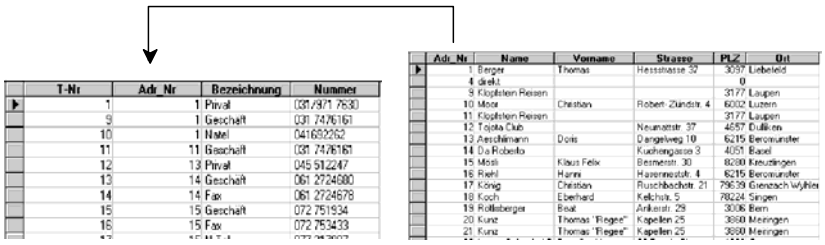
Foreign-Key (Fremdschlüssel)

Die Domain eines Foreign-Keys ist in einer anderen Relation als Primary-Key definiert. Beide Attribute oder Attributmengen (wenn der Key aus mehr als einem Attribut besteht) müssen der gleichen Domain unterliegen.

3. Relationale Integritätsregeln

3

Einen wichtigen Teil der Codd'schen Definitionen sind die Entity- und die referentielle Integrität. Integrität in Bezug auf eine Datenbank bedeutet, dass keine widersprüchlichen Daten in ihr gespeichert sind.



T-Nr	Adr_Nr	Bezeichnung	Nummer
1	1	Privat	031/971 7630
9	1	Geschäft	031 7476161
10	1	Natel	0416622262
11	11	Geschäft	031 7476161
12	13	Privat	045 512247
13	14	Geschäft	061 2724680
14	14	Fax	061 2724678
15	15	Geschäft	072 751934
16	15	Fax	072 753433

Tabelle Telefon

Adr_Nr	Name	Vorname	Strasse	PLZ	Ort
1	Berger	Thomas	Hessstrasse 32	3097	Liebefeld
4	direkt			0	
9	Klopfstein Reisen			3177	Laupen
10	Moor	Chestian	Robert Zündstr. 4	6002	Luzern
11	Klopfstein Reisen			3177	Laupen
12	Tojota Club		Neumattstr. 37	4057	Dulliken
13	Aeschimann	Doris	Dangelweg 10	6215	Beromunster
14	Da Roberto		Kuchengasse 3	4051	Basel
15	Mösli	Klaus Felix	Besmerstr. 30	8280	Kreuzlingen
16	Riehl	Hanni	Hasenmattstr. 4	6215	Beromunster
17	König	Chestian	Ruschibachstr. 21	79639	Grenzach Wyhlen
18	Koch	Eberhard	Kelchstr. 5	78224	Singen
19	Rollsberger	Beat	Ankerstr. 29	3006	Bern
20	Kunz	Thomas "Regee"	Kapellen 25	3860	Meiringen
21	Kunz	Thomas "Regee"	Kapellen 25	3860	Meiringen

Tabelle Adressen

Die Entity-Integrität

Nach der Definition einer relationalen Datenbank muss ein eindeutiger Objektbezeichner existieren, also ein Schlüssel (Key), welcher einen Datensatz in einer Tabelle eindeutig identifiziert. Diese Bedingung nennt man Entity-Integrität.

Ortmann 95

Im obigen Beispiel ist bei beiden Tabellen ein Primary-Key vorhanden:

- In der Tabelle 'Adressen' die 'Adr_Nr'.
- In der Tabelle 'Telefon' die 'T_Nr'.

Die referentielle Integrität

Die referentielle Integrität besagt, dass niemals in einem Fremdschlüssel Daten enthalten sein dürfen, die nicht im zugehörigen Primärschlüssel der anderen Tabelle enthalten sind.

Ortmann 95

In unserem Beispiel bedeutet dies, dass im Foreign-Key 'Adr_Nr' der Tabelle 'Telefon' kein Wert enhalten sein darf, der nicht im Primary-Key 'Adr_Nr' der Tabelle 'Adressen' vorkommt.

Die zwei Integritätsbedingungen sind so zu verstehen, dass zu keiner Zeit Daten in Relationen vorhanden sind, die nicht den Integritätsregeln genügen.
In einer relationalen Datenbank ist somit eine Datenintegrität vorhanden, dass heisst, die Widerspruchsfreiheit und die Korrektheit der Daten ist gegeben.

4. Datenbank-Design

4

Mit der Datenbank sollen Daten über die reale Welt gesammelt werden. Dies ist korrekt und zuverlässig mit einer Datenbank zu erledigen, wenn deren Struktur ein möglichst getreues Modell des zu bearbeitenden Teilausschnitts der realen Welt darstellt.
Sauer 1992

Unter dem Design einer Datenbank versteht man das Aufteilen der Daten in physisch getrennte Relationen.

4.1 Die Normalisierung einer Relation

Unter Normalisierung versteht man das Aufteilen der Daten in Relationen, dass sie am Ende den Normalisierungsregeln entsprechen.
E.F. Codds Definition (1972) unterscheidet drei Normalisierungsregeln oder drei Normalformen. Später wurden zwei weitere Regeln von R. Fagin (1977) eingeführt, die allerdings in der Praxis lediglich eine kleine Rolle spielen. Daher werden an dieser Stelle nur die Coddschen Normalformen aufgeführt.

Es gibt verschiedene Normalisierungsgründe:

- Verständlicheres Datenmodell für Benutzer und Entwickler
- Elimination von Redundanzen
- Verringerung der Notwendigkeit der Umstrukturierung von Relationen bei Einführung neuer Typen von Daten und dadurch Verlängerung der Lebensdauer von Datenbanken
- Das eindeutige Festhalten realitätskonformer Sachverhalte
- Last, but not least: Die Datenbankdesigner werden gezwungen, sich systematisch und intensiv mit den Daten zu beschäftigen.

4.2 Die Ausgangstabelle

Die folgende Tabelle enthält Daten einer kleinen Projektverwaltung. Zu den Mitarbeiterinformationen wie NAME und ABT-NAME gehören auch Projektdaten in die Tabelle. Eine Datenbank soll einfach, schnell und sicher bedient werden können. Diese Eigenschaften werden zwar meistens durch die Oberflächengestaltung erreicht, trotzdem werden einige dieser Eigenschaften durch die Datenbankstruktur bestimmt.

Können in unserem Beispiel die Daten sicher editiert werden, oder wie oft muss der Abteilungsname geändert werden, wenn eine betriebsinterne Umstrukturierung (zum Beispiel eine Änderung der Abteilungsnamen) erfolgt?
Die Normalisierung hilft uns, diese praktischen Bedingungen zu erfüllen.

MNR	NAME	VORNAME	ABT-NAME	PROJ-NR	PROJ-NAME	PROJ-ZEIT
101	MÜLLER	ROLF	LOHN	20,34	PR'95,SPAR	30,89
102	FLÜCKIGER	ANDREAS	LOHN	43	COMPI	120
103	SCHMID	HORST	EDV	20,43	PR'95,COMPI	40,7
104	TEPL	ARNOLD	PERSONAL			
105	BAROLO	ANNA	PR	20,34	PR'95,SPAR	85,2

Eine wichtige Regel besagt:
In einem Feld darf nur ein Wert eingetragen werden.

In unserem Beispiel kann man nicht genau wissen, ob im Datensatz 101 die PROJ-ZEIT 30 für die PROJ-NR 20 gilt, oder für 34. Daraus folgt, dass Einträge eindeutig sein müssen. Pro Feld darf jeweils nur ein Wert eingetragen werden, es darf also *keine innere Struktur* enthalten.

4.3 Die 1. Normalform

Eine Relation befindet sich in 1. Normalform, wenn ihre Attribute nur einfache Attributswerte aufweisen. Zehnder 1989

Wie müssen wir unsere Tabelle korrigieren, damit die 1. Normalform zutrifft?
Die einfachste Möglichkeit wird nachfolgend dargestellt:

MNR	NAME	VORNAME	ABT-NAME	PROJ-NR	PROJ-NAME	PROJ-ZEIT
101	MÜLLER	ROLF	LOHN	34	PR'95	30
101	MÜLLER	ROLF	LOHN	20	SPAR	30
102	FLÜCKIGER	ANDREAS	LOHN	43	COMPI	120
103	SCHMID	HORST	EDV	34	PR'95	40
103	SCHMID	HORST	EDV	43	COMPI	7
104	TEPL	ARNOLD	PERSONAL			
105	BAROLO	ANNA	PR	20	SPAR	85
105	BAROLO	ANNA	PR	34	PR'95	2

Die normalisierte Tabelle enthält nun zwar keine Felder mit innerer Struktur mehr, sie ist in 1. Normalform, aber viele Eintragungen sind doppelt vorhanden.
Wenn Daten mehrfach eingetragen sind, nennt man dies **Redundanz**. Eine Redundanz ist darauf zurückzuführen, dass Informationen zu verschiedenen Objekten in der gleichen Tabelle erfasst wurden.

Wie können wir dieses Problem umgehen? Die einzige Möglichkeit ist die Aufteilung unserer Beispieltabelle in mehrere Tabellen.

4.4 Die 2. Normalform

Tabelle 'Mitarbeiter'

MNR	NAME	VORNAME	ABT-NAME
101	MÜLLER	ROLF	LOHN
102	FLÜCKIGER	ANDREAS	LOHN
103	SCHMID	HORST	Z-CF
104	TEPL	ARNOLD	PERSONAL
105	BAROLO	ANNA	PR

Tabelle 'Projekt'

PROJ-NR	PROJ-NAME
20	PR'95
34	SPAR
43	COMPI

Tabelle 'Mitarbeiterprojekt'

MNR	PROJ-NR	PROJ-ZEIT
101	20	30
101	43	89
102	43	120
103	20	40
103	43	120
105	20	85
105	43	30

Eine Relation ist in 2. Normalform, wenn sie in 1. Normalform ist und jedes nicht zum Identifikationsschlüssel gehörige Attribut voll von diesem abhängig ist. Zehnder 1989

Diese Definition trifft in unserem Fall zu. Die Identifikationsschlüssel oder in der Access-Sprache, die Primärschlüssel, sind in einem grauen Feld (MNR, PROJ-NR).

Der ABT-NAME in der Tabelle 'Mitarbeiter' ist zwar im jetzigen Fall voll vom Identifikationsschlüssel (MNR) abhängig, doch ist dies sinnvoll? Eigentlich nicht, denn der ABT-NAME hat thematisch nicht viel mit der MNR zu tun. Wir teilen die Tabelle 'Mitarbeiter' auf, dies führt uns zur 3. Normalform.

4.5 Die 3. Normalform

Eine Relation befindet sich in 3. Normalform, wenn sie in 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, transitiv (indirekt) von diesem abhängt. Zehnder 1989

Die korrigierte Tabelle 'Mitarbeiter'

MNR	NAME	VORNAME	ABT
101	MÜLLER	ROLF	65
102	FLÜCKIGER	ANDREAS	65
103	SCHMID	HORST	12
104	TEPL	ARNOLD	45
105	BAROLO	ANNA	36

Die neue Tabelle 'Abteilung'

ABT	ABT-NAME
12	EDV
36	PR
45	PERSONAL
65	LOHN

Tabelle 'Projekt' (unverändert)

PROJ-NR	PROJ-NAME
20	PR'95
34	SPAR
43	COMPI

Tabelle 'Mitarbeiterprojekt' (unverändert)

MNR	PROJ-NR	PROJ-ZEIT
101	20	30
101	43	89
102	43	120
103	20	40
103	43	120
105	20	85
105	43	30

Die 3. Normalform ist mit dieser Aufteilung erreicht worden, die Identifikationsschlüssel (oder Primärschlüssel) sind in einem grauschraffierten Feld. Alle Attribute, die nicht zum Primärschlüssel gehören, sind direkt von ihnen abhängig.

5. Das Entity-Relationship-Modell

5

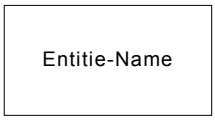
Die Methode der Normalisierung wird auch Bottom-Up genannt, da man vom elementarsten Level beginnt (den Attributen) und mit fertigen Relationen endet. In der Praxis ist diese Vorgehensweise allerdings nur dann anwendbar, wenn die zu strukturierenden Attribute und deren Zusammenhänge vorab sehr genau bekannt sind.

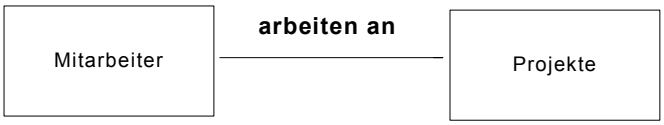
Wenn eine Datenbank sehr viele unterschiedliche Attribute beinhaltet, ist es ratsam, diese zunächst zu Hauptgruppen wie Projektdaten, Personendaten usw., also zu sogenannten Entities, zusammenzufassen. In einem weiteren Schritt werden dann deren Beziehungen (Relationship) zueinander festgelegt. Somit ist das Entity-Relationship-Modell eine Top-Down-Methode mit den folgenden zwei Schritten:

- Das Festlegen von Entities und deren Beziehung zueinander.
- Das Definieren von Attributen und Relationen für die Entities und Relationships in der Art, dass am Ende normalisierte Relationen entstehen.

5.1 Entity-Relationship-Diagramme (ER-Diagramme)

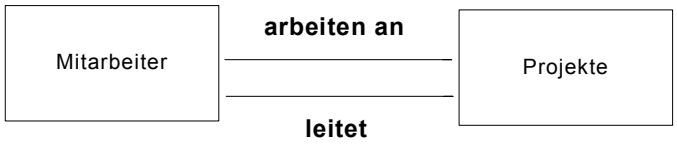
Ein Entitie wird durch nebenstehendes Symbol dargestellt, die Beziehung zwischen den Entities durch eine Linie, die eine Beziehungsbezeichnung enthält, z.B.:





Die Bezeichnung «arbeiten an» wurde aus der Sicht des Entities 'Mitarbeiter' gewählt, aus der Sicht der Entity 'Projekte' könnte man die Bezeichnung «werden bearbeitet» wählen.

Falls in unserem Beispiel ein Projekt einen Projektleiter besitzt, der auch ein Mitarbeiter ist, so existiert noch eine zweite Beziehung zwischen 'Mitarbeiter' und 'Projekte':



5.2 Degree eines Relationship (Grad einer Beziehung)

Man unterscheidet drei Grade von Entity-Beziehungen:

1 :1 Eins zu eins Beziehung
1 : N Eins zu viele Beziehung
N :M Viele zu viele Beziehung

Die Beziehungsgrade haben folgende Bedeutung:

1:1 Beziehung

Genau ein Mitarbeiter leitet genau ein Projekt. Daraus folgt, jedes Projekt wird von genau einem Mitarbeiter geleitet. Es muss jedoch nicht jeder Mitarbeiter ein Projekt leiten und nicht jedes Projekt muss einen Projektleiter besitzen.

1:N Beziehung:

Ein Mitarbeiter kann kein, ein oder mehrere Projekte leiten. Jedes Projekt muss von keinem oder genau einem Mitarbeiter geleitet werden.

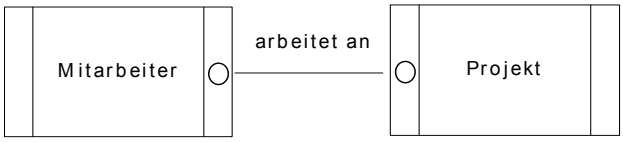
M:N Beziehung:

Ein Mitarbeiter kann an keinem, einem oder mehreren Projekten arbeiten. In einem Projekt können kein, ein oder mehrere Mitarbeiter beteiligt sein.

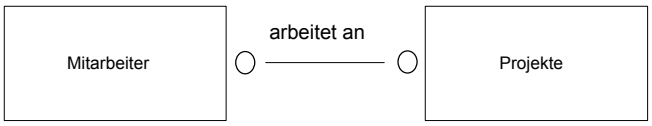
5.3 Mitgliedsklassen

Die Entity-Beziehung «M Mitarbeiter arbeiten an N Projekte» sagt nichts darüber aus, ob es Mitarbeiter gibt, die an keinem Projekt arbeiten. Oder ob es Projekte gibt, an denen kein Mitarbeiter arbeitet. Solche Aussagen werden mit Hilfe des Begriffs «obligatorische Mitgliedschaft» definiert. Obligatorische Mitgliedschaften werden pro Beziehung und Entity festgelegt.

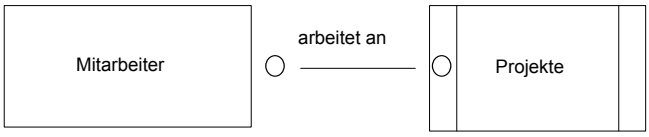
Jeder Mitarbeiter muss mindestens an einem Projekt arbeiten. In jedem Projekt arbeitet mindestens ein Mitarbeiter.



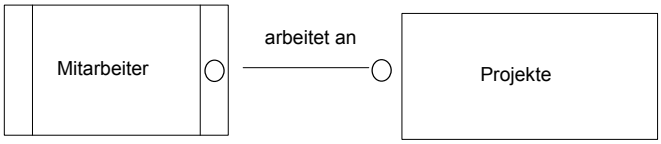
Ein Mitarbeiter muss nicht an einem Projekt arbeiten. Es gibt Projekte an denen kein Angestellter arbeitet.



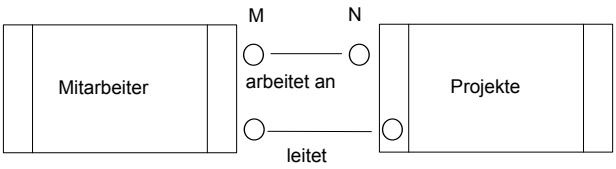
Ein Mitarbeiter muss nicht an einem Projekt arbeiten. An jedem Projekt arbeitet mindestens ein Mitarbeiter.



Jeder Mitarbeiter muss mindestens an einem Projekt arbeiten. Es gibt Projekte an denen kein Mitarbeiter beteiligt ist.



Ein Beispiel für eine komplexere Beziehungsstruktur: Ein Mitarbeiter leitet kein, ein oder mehrere Projekte und ein Projekt muss von genau einem Mitarbeiter geleitet werden.



6. Datenabfragesprachen

6

6.1 Grundlage

Was ist eine Abfragesprache?

relationale Algebra ist prozedural

Die relationale Algebra ist prozedural, das heisst wie ein gewünschtes Ergebnis erreicht wird, muss der Anwender selber formulieren. In der Praxis kann dies zu einigen Schwierigkeiten führen, da der Programmierer auch über interne Abläufe viel verstehen muss, um die Befehle optimal zu formulieren. Dies ist einer der Gründe dafür, wehalb relationale Abfragesprachen vermehrt nicht-prozedural sind.

neue relationale Abfragesprachen sind nicht-prozedural

Nicht-prozedurale Abfragesprachen sind Programmiersprachen, die nur ausdrücken, welches Ergebnis gewünscht wird, und nicht wie der Rechner zu diesem Ergebnis kommen soll.

Diese Art von Sprachen werden zu den Programmiersprachen der vierten Generation (4GL) gezählt. Die derzeit verbreiteste Sprache für relationale Datenbanken ist **Structured Query Language (SQL)**.

Geschichte von SQL (Structured Query Language)

Anfang der 70er Jahre wurde von den IBM Forschungslaboratorien in San Jose ein Projekt begonnen, das die Praktizierbarkeit der relationalen Theorien untersuchte. Innerhalb dieses Projektes wurde eine Sprache namens SQUARE definiert. Es erwies sich in der Syntax als zu mathematisch.

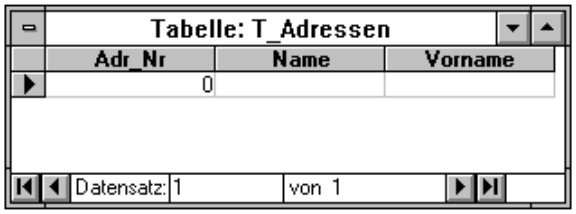
Von den IBM-Mitarbeitern Boyce und Chamberlain wurde die Sprache SEQUEL entwickelt, die später in SQL umbenannt wurde. Dieses Projekt wurde weiterverfolgt und es kamen die Datenbanksysteme DB2 und SQL/DS auf den Markt. Seit dieser Zeit werden praktisch von allen Datenbank-Herstellern SQL-Schnittstellen zu ihren relationalen und nichtrelationalen Datenbanksystemen entwickelt.

6.2 Ein SQL-Beispiel

Tabelle erstellen

Mit folgenden SQL-Anweisungen wird die untenstehende Tabelle definiert:

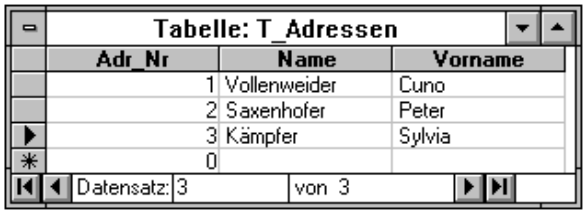
```
CREATE TABLE T_Adressen
   (Adr_Nr  SMALLINT,
    Name    CHAR(15),
    Vorname CHAR(15))
```



Daten einfügen

Mit dem INSERT-Befehl ist es möglich, Daten in eine Tabelle einzufügen, z.B. ergeben folgende Anweisungen untenstehende Tabelle:

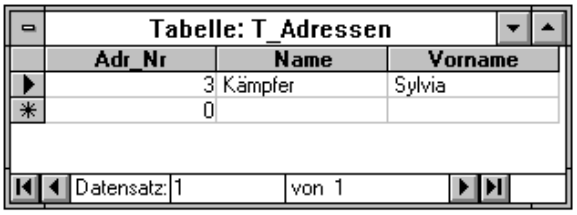
```
INSERT INTO T_Adressen
   VALUES (1,'Vollenweider', 'Cuno')
INSERT INTO T_Adressen
   VALUES (2,'Saxenhofer'', 'Peter')
INSERT INTO T_Adressen
   VALUES (3,'Kämpfer', 'Sylvia')
```



Daten auswählen

Mit dem SELECT-Befehl werden Daten aus einer oder mehreren Tabellen ausgewählt:

```
SELECT Adr_Nr, Name, Vorname
FROM T_Adressen
WHERE Adr_Nr = 3
```



6.3 SQL-Befehlübersicht

In der Praxis unterschiedet man bei den relationalen Datenbanken zwei Gruppen von Datenbankanweisungen:

DDL Data Definition Language

Dazu zählen alle Anweisungen, mit denen die logische Struktur der Tabelle verändert werden. Zum Beispiel:

`CREATE TABLE`	neue Basistabelle erstellen
`CREATE VIEW`	definieren einer logischen Tabelle
`GRANT`	Benutzerberechtigungen vergeben

DML Data Manipulation Language

Dazu zählen alle Anweisungen, die dazu dienen, die Daten zu verarbeiten:

`SELECT`	Daten aus der Datenbank lesen
`DELETE`	Zeilen einer Tabelle löschen
`UPDATE`	Zeilen einer Tabelle ändern
`INSERT`	Zeilen in einer Tabelle zufügen

7. Die Datenbanksoftware-Auswahl

7

Mittlerweile befinden sich etwa 150 verschiedene relationale DB-Systeme auf dem Markt. Welches ist nun aber das beste bzw. das geeignetste System?

7.1 Allgemeine Vorgehensweise

1. Anforderungskatalog

Zunächst wird in Abhängigkeit vom zukünftigen Datenbank-Umfeld ein Anforderungskatalog erstellt, der die erwarteten Anforderungen an das zukünftige DB-Management-System (DBMS) enthält. Dieser wird in folgende Abschnitte untergegliedert:

- K.O.-Kriterien
- Strategische Kriterien
- Technische Kriterien

2. Gewichtung der Kriterien

Hier werden die einzelnen Kriterien und Hauptpunkte in Abhängigkeit des zukünftigen Einsatzgebietes gewichtet. Kriterien, die als wichtig erachtet werden, bekommen eine hohe Gewichtungspunktzahl (z.B. Wertungs-Skala von 1 bis 10).

3. Erfüllungsgrad (EG)

Hier wird der Erfüllungsgrad der in Frage kommenden DB-Managementsysteme bezüglich der Kriterien untersucht (z.B. von 0 bis 10).

4. Punktzahl = Gewichtung x Erfüllungsgrad

Hier werden pro Kriterium die vergebenen Punkte mit der Gewichtung multipliziert. Das DB-Managementsystem mit der höchsten Gesamtpunktzahl ist dann für dieses Einsatzgebiet am besten geeignet.

		DB1		DB2		DB2	
Kriterien	**Gewichtung**	**EG**	**Punkte**	**EG**	**Punkte**	**EG**	**Punkte**
1. Flexibilität der Anw.-Entwicklung	4	3	**12**	8	**32**	6	**24**
2. Integration der SW-Produkte	6	4	**24**	7	**42**	4	**24**
3. Netzwerkfähigkeit	8	8	**64**	3	**24**	8	**64**
4. Datensicherheit	7	3	**21**	5	**35**	9	**63**
	Total		**121**		**133**		**175**

Beispiel einer kleinen Entscheidungsmatrix

7.2 Entscheidungskriterien

K.O.-Kriterien

Hier werden die unabdingbaren Anforderungen an das zukünftige DB-Managementsystem (DBMS) aufgeführt:

- bestehende Hardware, die das DBMS unterstützen muss
- gewünschtes Betriebssystem, die das DBMS unterstützen muss
- finanzieller Aufwand
- die bestehende Verbreitung des DBMS

Strategische Kriterien

Unter diesem Punkt werden allgemeine Angaben, die nicht mit einer exakten Zahl ausgedrückt werden können, aufgeführt:

- Derzeitiger Reifegrad der Produkte und die Produktpalette im geplanten Hard- und Softwareumfeld
- Langfristige Perspektiven der Produkte
- Grad der Abhängigkeit vom DBMS-Hersteller
- Externes Personalangebot und vorhandenes Know-How
- Unterstützung von Standards
- Welches ist die aktuelle Version?
- Zugrundeliegende Programmiersprache
- Anzahl der Versionswechsel und der zu erwartende Aufwand pro Versionswechsel
- Laufzeitversionen

Technische Kriterien

Hauptanforderungen an relationale Datenbankmanagement-Systeme sind:

- Flexibiltät der Anwendungsentwicklung und -wartung
- Netzwerkfähigkeit
- Datensicherheit
- Datenschutz
- Betriebsführung
- Herstellerunterstützung
- Endbenutzertools