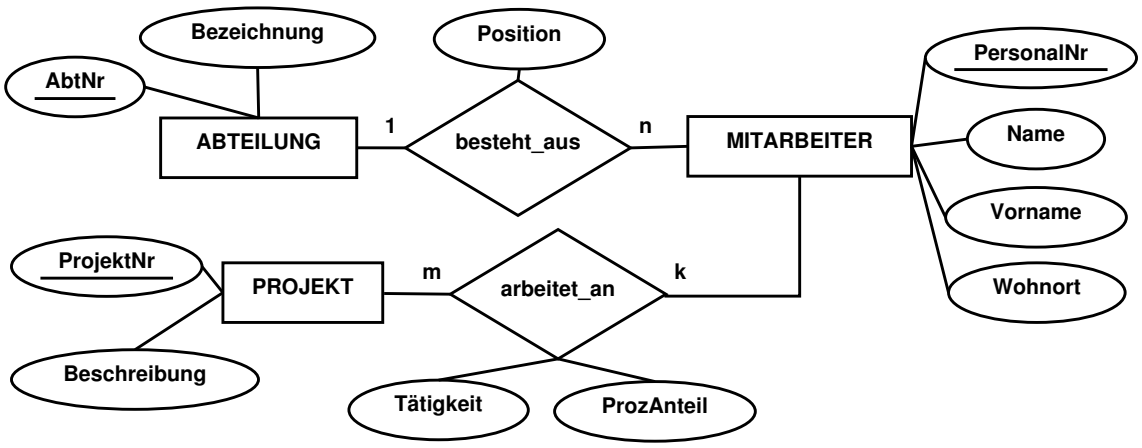
6. Vom Bauplan zur Datenbank

Ein Bauplan alleine ist nicht viel wert. In diesem Kapitel möchte ich Ihnen nun zeigen, wie Sie Ihren mühsam erstellten theoretischen Datenbankentwurf in das Relationale Modell überführen können. Das Ergebnis ist dann endlich direkt in Microsoft Access umsetzbar. Es wird also eine reine technische Beschreibung von Tabellen und ihrer Struktur, dem Datenbank–Schema sein.

6.1. Man nehme: Ein Beispiel

Dieses Buch ist gleichzeitig die Erfolgsgeschichte eines kleinen, leider nur in meiner Fantasie existierenden, Bücherladens BUCH4YOU und seinem Geschäftsführer, einem Herrn Gernegroß. Dieser Geschäftsführer hat beschlossen, sein Unternehmen zu modernisieren und was gibt es schon moderneres als seine Daten mit einem Datenbankmanagementsystem zu pflegen? Kurzerhand wird der studierende Neffe von Herrn Gernegroß beauftragt, einen theoretischen Datenbankentwurf für den Bücherladen zu erstellen. Betrachten Sie also das folgende ER–Diagramm für die Datenbank von BUCH4YOU, das der Neffe als Ergebnis lieferte:



Einem Kochrezept gleich werden wir jetzt Schritt für Schritt die Entitäten und Beziehungen in Tabellen überführen. Was denken Sie, wie viele Tabellen am Ende dabei heraus kommen werden? Wenn Ihr Ergebnis drei lautet: Sie haben gut gedacht, lediglich eine Kleinigkeit übersehen. Sollten Sie auf vier oder fünf gekommen sein, schämen Sie sich, sofern Sie vor Ihrer Antwort nachgesehen hatten. Wo wir gerade dabei sind, werfen Sie doch noch mal einen Blick auf das ER–Diagramm: Was wird wohl in dem Attribut *ProzAnteil* gespeichert?

6.2. Zuerst die Entitäten ...

Man wird es bereits vermuten: Aus jeder Entität wird im Relationalen Modell eine eigene Tabelle.

Tipp Alle Tabellen- und Spaltennamen sollten dabei identisch gewählt werden wie im ER–Diagramm. Das ER–Diagramm ist für den Menschen leichter lesbar als eine Liste von Tabellennamen und enthält mehr semantische Informationen, die zwar für die technische Umsetzung auf den ersten Blick nicht so wichtig sind, wohl aber, wenn man mit einem Auftraggeber über die Datenmodellierung sprechen will oder gedankliche Fehler bei der technischen Umsetzung suchen muss. Auf diese Weise kann ein Anwendungsentwickler bei seiner Arbeit sinnvoll mit dem theoretischen Datenbankentwurf arbeiten und muss sich nicht jedes Mal durch Tabellenlisten quälen. Werden Änderungen im Relationalen Modell vorgenommen, sollten diese dafür dann aber auch auf jeden Fall in das ER–Diagramm mit zurück übernommen werden. Theoretischer Datenbankentwurf und Relationales Modell müssen sich immer auf die gleiche Ist–Situation beziehen, um so den vollen Nutzen aus beiden Modellen ziehen zu können.

Aber jetzt zur Umsetzung — Man nehme:

- Jede Entität wird eine eigene Tabelle.
- Jedes Attribut der Entität wird eine eigene Spalte der Tabelle.
- Der Primärschlüssel der Entität wird Primärschlüssel der Tabelle.
- Sofern nicht bei der ER–Modellierung geschehen, müssen jetzt für jede Spalte geeignete Datentypen vergeben werden.
- Eventuell müssen weitere Einstellungen für die Spalten vorgenommen werden. Bei einer Postleitzahl kann es sinnvoll sein, im Eingabeformat festzulegen, dass sie nur aus exakt fünf Ziffern bestehen darf, sofern nur deutsche Postleitzahlen gespeichert werden sollen.

Wir haben für Buch4You folgende Entitäten zu betrachten:

- Entität ABTEILUNG → Tabelle ABTEILUNG:

Schlüssel	Spaltenname	Datentyp
✓	AbtNr	AutoWert
	AbteilungsName	Text(30)

gefüllt mit den folgenden Daten, die uns Herr Gernegroß gab:

AbtNr	AbteilungsName
1	Personal
2	Einkauf
3	Verkauf

- Entität PROJEKT → Tabelle PROJEKT:

Schlüssel	Spaltenname	Datentyp
✓	ProjektNr	AutoWert
	Beschreibung	Text(50)

mit den Daten:

ProjektNr	Beschreibung
1	Kundenumfrage
2	Verkaufsmesse
3	Konkurrenzanalyse

➤ Entität MITARBEITER → Tabelle MITARBEITER:

Schlüssel	Spaltenname	Datentyp
✓	PersonalNr	AutoWert
	Name	Text(30)
	Vorname	Text(30)
	Wohnort	Text(30)

hier ebenfalls mit einem Einblick in die Daten:

PersonalNr	Name	Vorname	Wohnort
1	Lorenz	Sophia	Mausbach
2	Hohl	Tatjana	Mausloch
3	Willschrei	Theodor	Katzbergen
4	Richter	Hans	Katzenhausen
5	Wiesenland	Brunhilde	Hundsberg

Hierbei soll *Text(50)* für den Microsoft Access Datentyp Text mit einer maximalen Länge von 50 Zeichen stehen. Die Festlegung dieser Anzahl Zeichen erfolgt hier bei der Übertragung in das Relationale Modell. Zu viele Zeichen wären eine Speicherplatzverschwendung, zu wenige könnten bedeuten, dass die Daten nicht ohne Abkürzungen aufgenommen werden können. Zwar kann die Anzahl der Zeichen auch noch später angepasst werden, sinnvoller ist es gleich eine vernünftige Größe zu wählen. Eine Möglichkeit ist, die Anzahl der Zeichen im längsten Dateneintrag der Beispieldaten einfach mal zwei zu nehmen. Gut, wenn man dann Beispieldaten hat. Allgemein ist es immer ein guter Rat, sich so viele Beispieldaten geben zu lassen, wie nur möglich, um die Problemwelt dahinter analysieren und Zusamenhänge verstehen zu können. Damit sind nicht nur elektronisch erfasste Daten gemeint, Papier–Listen etc. können ebenfalls sehr hilfreich sein.

Tipp

6.3. ... dann die Relationships

Die Projekte, Abteilungen und Mitarbeiter können wir jetzt erfassen. Was noch fehlt, sind die Beziehungen zwischen dern Entitäten, den Relationships. Wäre unsere Datenbank nur auf dem Papier gemalt, würden wir schnell auf die Idee kommen,

durch einen Pfeil von *Personalabteilung* zu *Lorenz, Sophia* kennzeichnen, dass sie in dieser Abteilung arbeitet. Die Information, welche Tätigkeit sie dort ausübt, würde man vielleicht an dem Pfeil zusätzlich vermerken.

Relationale Datenbankmanagementsysteme haben leider ihre liebe Not mit dem Malen von Pfeilen. Wie in Kapitel 3 gesagt, können sie Daten ausschließlich in Tabellen speichern. Die Betonung liegt dabei auf ausschließlich. Aber ein Pfeil verbindet doch nur zwei Punkte. Diese Punkte sind wiederum Datensätze. Datensätze sind eindeutig gekennzeichnet durch ... — richtig: den Schlüssel! Statt einen Pfeil in unseren Computer zu malen, könnten wir uns doch stattdessen einfach Anfangs- und Endpunkt, repräsentiert durch die Schlüsselwerte der entsprechenden Datensätze merken. Dieses Merken wird dann in Form eines Datensatzes in einer Tabelle erledigt. Zusätzliche Informationen, wie die Tätigkeit in einer Abteilung schreiben wir dann zu diesem Datensatz dazu. Das ist die Grundidee des Speicherns von Beziehungen bei Relationalen Datenbankmanagementsystemen.

6.3.1. Kardinalitäten

Am Ende des Kapitels 5 hatte ich damit gedroht, dass das Relationale Modell keine many–to–many Beziehungen kennt. Auf dem Papier können wir beliebig von einem Punkt zu einem anderen Verbindungen ziehen, auch mehrere, die den gleichen Start- und Endpunkt besitzen. Im Relationalen Modell werden wir an dieser Stelle Schwierigkeiten mit der Schlüsselbedingung bekommen. Daher müssen wir die Umsetzung nach den Kardinalitäten der Bezihungen getrennt betrachten.

6.3.2. one–to–one und one–to–many Beziehungen

Hier haben wir angenehmer Weise zwei Möglichkeiten zur Auswahl.

Erste Möglichkeit

Man nehme — eine neue Tabelle, wie bei der Grundidee skizziert. Die Tabelle besitzt als Spalten alle Attribute der Primärschlüssel beider beteiligten Entitäten. Die Datentypen müssen dabei identisch sein, damit in der Spalte der neuen Tabelle die Schlüsselwerte der jeweiligen Spalte im Schlüssel der Entität abgespeichert werden können. Die einzige Ausnahme hierbei bildet der Datentyp *AutoWert*. Es ist wenig sinnvoll, wenn in der neuen Tabelle die Werte einfach von eins beginnend wieder hochzählen. Stattdessen soll ja der bereits vergebene, automatisch generierte Wert aus der Tabelle der Entität hier eingetragen werden, dessen Datensatz hier in Beziehung steht. Es soll ja kein neuer eindeutiger Wert erzeugt werden. Die Lösung ist einfach, tragen Sie als Datentyp in den neuen Spalten *LongInteger* ein. Weitere beschreibende Attribute am Relationship werden als zusätzliche Spalten der neuen Tabelle hinzugefügt. Jedes weitere Attribut aus den Tabellen von ABTEILUNG oder MITARBEITER außer den Schlüsseln wäre pure Redundanz.

Bleibt noch der Primärschlüssel der neuen Tabelle festzulegen. Bei one–to–one ist es egal, genau einer der beiden Ausgangsschlüssel wird zum Schlüssel der neuen Tabelle bestimmt. Bei one–to–many wird der Schlüssel der to–many Entität der Primärschlüssel der neuen Tabelle. Man kann es sich aber auch über die inhaltliche

Bedeutung veranschaulichen: Bei BOOK4YOU steht ABTEILUNG in einer one–to–many Beziehung zu der Entität MITARBEITER. Die neue Tabelle beschreibt wie eine Zuordnungstabelle welcher Mitarbeiter in welcher Abteilung arbeitet. Durch die one–to–many Beziehung ist offensichtlich, dass gleichzeitig ein Mitarbeiter nur einer Abteilung zugeordnet sein kann, während eine Abteilung aus mehreren Mitarbeitern bestehen wird. Folgerichtig ist der Schlüssel dieser Tabelle die Spalte *PersonalNr*, wodurch sichergestellt ist, dass jeder Mitarbeiter nur einmal einer Abteilung zugeordnet werden kann. Wäre die hingegen *AbtNr* als Schlüssel festgelegt worden, hätten wir die sehr spannende Situation, dass in jeder Abteilung maximal ein Mitarbeiter sitzen darf, dafür ein Mitarbeiter mehrere Abteilungen gleichzeitig besetzen kann. Dies mag ein Gedankenmodell für eine ein–Mann–Firma sein, nicht für die Realität und schon gar nicht für BOOK4YOU. Die letzte Möglichkeit wäre, beide Schlüssel zusammen den neuen Primärschlüssel bilden zu lassen. Damit darf dann jede Kombination in den Spalten nur ein Mal auftreten, in unserem Fall müsste dann nur die *PersonalNr–AbtNr* Kombination eindeutig sein. Damit hätten wir zwar wieder mehrere Mitarbeiter in einer Abteilung sitzen, aber ein Mitarbeiter dürfte sehr wohl auch anderen Abteilungen gleichzeitig zugeordnet werden, allerdings jeder nur ein Mal — ebenfalls nicht ganz so, wie ursprünglich beabsichtigt. Es bleibt dabei, der einzig sinnvolle Primärschlüssel ist *PersonalNr*.

➤ Relationship besteht_aus → Tabelle BESTEHT_AUS:

SCHLÜSSEL	Spaltenname	Datentyp
✓	PersonalNr	LongInteger
	AbtNr	LongInteger
	Position	Text(30)

die so aussehen könnte:

PersonalNr	AbtNr	Position
1	1	Leiterin
2	2	Leiterin
3	2	Mitarbeiter
4	3	Leiter
5	3	Mitarbeiterin

Hier stolpern wir über ein zusätzliches Problem. In der Spalte *Position* sind in Wirklichkeit zwei Daten hinterlegt. Während die wirkliche Position wie die Leitung sicherlich etwas ist, dass durch das Relationship zwischen ABTEILUNG und MITARBEITER charakterisiert wird, hat sich hier noch klammheimlich das Geschlecht des Mitarbeiters mit hinein geschlichen. Das Geschlecht ist aber ein Attribut, welches allein zum Mitarbeiter gehört und damit auch in die Entität MITARBEITER. Nicht ohne Grund sind in unserem Sprachgebrauch Amtsbezeichnungen geschlechtsneutral. Des weiteren besteht hier Inkonsistenzgefahr. Nehmen wir an, die Mitarbeiterin mit der Personalnummer 2, Frau Hohl und der Mitarbeiter mit der Personalnummer 4, Herr Richter tauschen ihre Aufgabegebiete. Da beide jeweils die Leitung in ihrer Abteilung haben, ändert sich an der Position nichts. In der Tabelle BESTEHT_AUS

müssten wir nur bei den entsprechenden Abteilungsnummern die neuen Werte eintragen. Leider haben wir damit auf eine sehr preiswerte Art und Weise den beiden Mitarbeitern gleichzeitig noch eine Geschlechtsumwandlung zukommen lassen.

Tipp Prüfen Sie ihre Attribute immer darauf, ob nicht versteckt mehr als ein Datum in einer Spalte gespeichert wird. In der Begriffswelt des Relationalen Modells bedeutet

Begriff dies, dass in der obigen Tabelle das Attribut *Position* nicht *atomar*[1] ist. Sofern Herr Gernegroß es benötigt, werden wir die Entität und die Tabelle MITARBEITER um eine Spalte Geschlecht ergänzen und verfassen die Tabelle von eben besser so:

PersonalNr	AbtNr	Position
1	1	Leiter
2	2	Leiter
3	2	Mitarbeiter
4	3	Leiter
5	3	Mitarbeiter

Zweite Möglichkeit

Überlegen wir einmal weiter: Durch die im letzten Abschnitt neu erstellte Tabelle haben wir die Beziehung gespeichert. Dadurch, dass *PersonalNr* Primärschlüssel dieser Tabelle ist, liegt die Vermutung nahe, dass wir an sich aber nur weitere Informationen gespeichert haben, die über eben diesen Schlüssel genau so gut mit dem Mitarbeiter assoziiert werden können. Diese Überlegung liefert uns die zweite Möglichkeit für die Speicherung der Beziehungsdaten.

Die Tabelle der to–many Entität des Relationsships wird erweitert um den Primärschlüssel der Entität der to–one Seite. Bei einer one–to–one Beziehung ist es egal, welche Seite erweitert wird. Der Primärschlüssel der erweiterten Tabelle bleibt dabei aber unverändert. Lediglich weitere Attribute, die zusätzlich am Relationship hängen, erweitern ebenfalls das Schema der Tabelle, genau wie im letzten Abschnitt auch geschehen.

Die zweite Möglichkeit bedingt also, dass wir das Schema der Tabelle MITARBEITER noch einmal anpassen müssen.

Schlüssel	Spaltenname	Datentyp
✓	PersonalNr	AutoWert
	Name	Text(30)
	Vorname	Text(30)
	Wohnort	Text(30)
	AbtNr	LongInteger
	Position	Text(30)

Gefüllt mit den gleichen Daten wie im letzten Abschnitt:

[1] Für die Mathematik–Fetischisten unter uns: In *Position* ist also kein einzelner Wert, sondern eine kleine Liste, genauer: ein Wertepaar, ein Zweier–Tupel („Leiter“, „weiblich“) gespeichert.

PersonalNr	Name	Vorname	Wohnort	AbtNr	Position
1	Lorenz	Sophia	Mausbach	1	Leiter
2	Hohl	Tatjana	Mausloch	2	Leiter
3	Willschrei	Theodor	Katzbergen	2	Mitarbeiter
4	Richter	Hans	Katzenhausen	3	Leiter
5	Wiesenland	Brunhilde	Hundsberg	3	Mitarbeiter

Die Notwendigkeit einer zusätzlichen Tabelle entfällt bei dieser Umsetzung. Ein Mitarbeiter, der (noch) zu keiner Abteilung zugeordnet ist, erzeugt bei dieser Lösung zwei null–Werte in den Spalten *AbtNr* und *Position*, während er bei der ersten Möglichkeit der Umsetzung gar keinen Datensatz in der Tabelle BESTEHT_AUS erzeugt hätte.

Erweitern Sie niemals bei einer one–to–many Beziehung die Tabelle der to–one Entität! Auf diese Weise entstehen sehr schnell Redundanzen und Verletzungen des Schlüsselkriteriums. Um es uns zu veranschaulichen, werden wir die Tabelle ABTEILUNG falsch erweitern:

Schlüssel	Spaltenname	Datentyp
✓	AbtNr	AutoWert
	AbteilungsName	Text(30)
	PersonalNr	LongInteger
	Position	Text(30)

Um die Situation bei Book4You zu speichern, müssten wir offensichtlich manche Abteilungsdatensätze mehrfach anlegen, sofern auch mehrere Mitarbeiter in der betreffenden Abteilung arbeiten:

AbtNr	AbteilungsName	PersonalNr	Position
1	Personal	1	Leiter
2	Einkauf	2	Leiter
2	***Einkauf***	3	Mitarbeiter
3	Verkauf	4	Leiter
3	***Verkauf***	5	Mitarbeiter

Niemand könnte verhindern, dass im dritten Datensatz der *AbteilungsName* von *Einkauf* zu *Reklamation* verändert wird. Wie heißt dann die Abteilung mit der Nummer *2* wirklich? Das Datenbankmanagementsystem würde aber in der letzten Tabelle das Einfügen der Datensätze drei und fünf schon von vornherein verweigern, da der Schlüsselwert für *AbtNr* bereits einmal vergeben wurde.

Absolut verboten und mega–böse ist die Idee, zur Umgehung der Verletzung des Schlüsselkriteriums in *AbtNr* für Book4You das Schema der Tabelle ABTEILUNG um zwei weitere Spalten *PersonalNr2* und *Position2* für den zweiten Mitarbeiter zu ergänzen. Was ist dann bei drei Mitarbeitern in einer Abteilung — oder bei 124 Mitarbeitern? Neben einem sehr unübersichtlichem Schema ist das Ergebnis einer solchen Vorgehensweise, dass Abfragen extrem kompliziert werden: Ist ein gesuchter Mitarbeiter unter *PersonalNr, PersonalNr2* oder *PersonalNr124* gespeichert? Eine Abfrage müsste jede dieser Möglichkeiten berücksichtigen.

Wann wähle ich welche Umsetzung?

Die Möglichkeit mit der neuen Tabelle bedeutet, dass eine zusätzliche Tabelle angelegt wird. Umso mehr Tabellen eine Datenbank enthält, desto wahrscheinlicher verlangsamt dieser Umstand gleichzeitig Abfragen auf den beiden beteiligten Entitäten. Bemerken werden Sie dies allerdings erst bei vielen Datensätzen. Dafür hat sie den Vorteil, dass wirklich nur Datensatz angelegt wird, wenn auch wirklich eine Beziehung besteht. Bei der zweiten Möglichkeit der Erweiterung der to–many Entität muss in dem Fall, dass keine Beziehung besteht der Datensatz mit null–Werten belegt werden. Welches ist also die bessere Möglichkeit eine one–to–one oder one–to–many Beziehung umzusetzen?

Es lässt sich nicht pauschal sagen, welche Möglichkeit die optimale ist. Stellen Sie sich die Entitäten BUCH und AUSLEIHER vor. Jedes Buch kann nur ein Mal ausgeliehen sein, aber jeder Ausleiher kann mehrere Bücher entleihen. In einer Stadtbibliothek werden viele Bücher gleichzeitig ausgeliehen sein, eine zusätzliche Tabelle daher sehr groß und damit Abfragen ausbremsend. Die gleiche Situation aber betrachtet vor dem Hintergrund der Bibliothek einer Universitätsklinik mit einem hohen Präsenzbestand und nur wenigen ausleihbaren Lehrbüchern, lässt sofort die Möglichkeit der neuen Tabelle interessanter werden, da diese Tabelle nur klein wäre und die Erweiterung der BUCH Tabelle nur unnötig viele null–Werte bringen würde bei den Präsenzexemplaren, die nie entliehen werden dürfen.

Im Zweifel ziehen Sie trotzdem immer die Methode der Erweiterung der to–many Entität vor. In der Datenbank–Literatur wird Ihnen die erste Möglichkeit von vielen Autoren sogar verschwiegen. Der Ansatz, auf möglichst wenige Tabellen im Gesamtschema zu kommen, hat sich in der Praxis am besten bewährt.

6.3.3. many–to–many Beziehungen

Bei den many–to–many Beziehungen haben wir nur die Möglichkeit, eine neue Tabelle anzulegen, um das Relationship zu speichern.

Die neue Tabelle setzt sich zusammen aus den beiden Primärschlüsseln der beiden beteiligten Entitäten, die zusammen den Schlüssel der neuen Tabelle bilden. Weitere Attribute, die ggf. noch an dem Relationship hängen werden ebenfalls mit in diese neue Tabelle aufgenommen.

Vorher ist aber auch ein guter Zeitpunkt über meine letzte Frage am Ende von Abschnitt 6.1 noch einmal zu sprechen: Was ist bei ProzAnteil gespeichert? Es ist ein zusätzliches Attribut der Beziehung zwischen MITARBEITER und PROJEKT. Es wird also beschreiben, wie viel Prozent ein bestimmter Mitarbeiter an einem bestimmten Projekt mitarbeitet. Gut, aber was sind hierbei 100 %? Es könnten sowohl die 100 % = die gesamte Arbeitszeit eines Mitarbeiters als auch 100 % = die gesamte Arbeitszeit an einem Projekt gemeint sein. Das ER–Diagramm allein gibt hierüber keine Auskunft!

➤ Relationship arbeitet_an → Tabelle ARBEITET_AN:

Schlüssel	Spaltenname	Datentyp
✓	PersonalNr	LongInteger
✓	ProjektNr	LongInteger
	Tätigkeit	Text(30)
	ProzAnteil	Integer

Bei Book4You sind die Projekte zur Zeit so geplant:

PersonalNr	ProjektNr	Tätigkeit	ProzAnteil
2	1	Leiter	50
3	1	Sachbearbeiter	50
4	1	Sachbearbeiter	50
4	2	Leiter	25
5	2	Präsentationsvorbereitung	100
4	3	Leiter	25
2	3	Sachbearbeiter	50
3	3	Sachbearbeiter	50

Erst jetzt können wir mit etwas mehr Überzeugung vermuten, dass 100 % bei *ProzAnteil* für die 100 % Arbeitszeit des Mitarbeiters stehen, da wir anhand der Beispieldaten sehen können, dass Mitarbeiter in der Summe immer zu genau 100 % ausgelastet sind[2]. Die Projekte wären in der Summe mit 150 % und zwei Mal mit 125 % geplant, was definitiv ein Fall für den Betriebsrat wäre. Mit diesem Beiepiel möchte ich Ihnen noch einmal in Erinnerung rufen, wie wichtig Beispieldaten und eine zusätzliche verbale Beschreibung zu einem ER–Diagramm sind. Die Beispieldaten lieferten uns zusammen mit ein wenig gesundem Menschenverstand eine Vermutung und die zusätzliche Beschreibung im ER–Diagramm hätte uns bei der nächsten Besprechung mit Herrn Gernegroß daran erinnert, ihn darauf hin anzusprechen, um so die letzte Sicherheit über die Bedeutung von *ProzAnteil* zu erhalten.

Durch die technische Umsetzung des many–to–many Relationships wird gleichzeitig deutlich, dass zwar jede Entität aus der einen Entitätsmenge mit beliebig vielen anderen aus der anderen Entitästmenge in Beziehung stehen kann (und umgekehrt). Jedoch können niemals die gleichen zwei Entitäten mehr als ein mal miteinander in Beziehung stehen, dies ist selbst bei einer many–to–many Beziehung nicht möglich.

Jeder Versuch, das many–to–many Relationship als Tabellen–Erweiterung umzusetzen, würde in der Verletzung des Schlüsselkriteriums enden! Wenn wir die Tabelle PROJEKT erweitern, würde dies inhaltlich bedeuten, dass nur noch ein Mitarbeiter an einem Projekt gleichzeitig arbeiten könnte, da ansonsten der entsprechende Schlüsselwert in *ProjektNr* mehrmals auftauchen müsste. Umgekehrt ist auch keine Lösung Mitarbeiter zu erweitern, da dadurch jeder Mitarbeiter gleichzeitig nur ein Projekt bearbeiten könnte. Dies wäre für Herrn Gernegroß dann doch etwas zu wenig Multitasking in seiner Belegschaft.

[2]Seien Sie ehrlich: Haben didaktische Beispiele nicht etwas magisches, wenn immer alles passt?

6.4. Notation des Relationalen Modells

Jedes Mal die Tabellen selbst hinzuzeichnen, ist zwar übersichtlich aber viel zu aufwändig. Analog zur textuellen Notation für das ER–Modell gibt es eine textuelle Notation für das Relationale Modell. Leider sehen sich beide Darstellungen verteufelt ähnlich, obwohl sie unterschiedliches beschreiben. Im Relationalen Modell werden ausschließlich die Tabellen angegeben. Dadurch ist ein Unterschied zwischen der Herkunft einer Tabelle aus einer Entität oder einem Relationship des ER–Modells ohne die Kenntnis des inhaltliches Hintergrundes der Datenbank nur sehr schwer erkennbar. Wir vereinbaren die folgenden Konventionen:

- Tabellennamen werden in Großbuchstaben angegeben.
- Nach dem Tabellennamen folgt eine geklammerte Liste der Spalten.
- Spaltennamen beginnen mit einem Großbuchstaben, werden dann mit Kleinbuchstaben fortgesetzt. Besteht der Spaltenname aus mehreren Worten, so wird der erste Buchstabe jedes Wortes groß geschrieben und die Worte direkt aneinander gehängt.
- Nach jedem Spaltennamen kann optional der Datentyp für dieses Attribut angegeben werden. Zwischen dem Spaltennamen und dem Datentyp steht in diesem Fall ein Doppelpunkt als Trennzeichen.
- Der Schlüssel sollte in der Attributliste möglichst weit vorne stehen. Alle Attribute des Primärschlüssels werden mit ihren Datentypen unterstrichen.

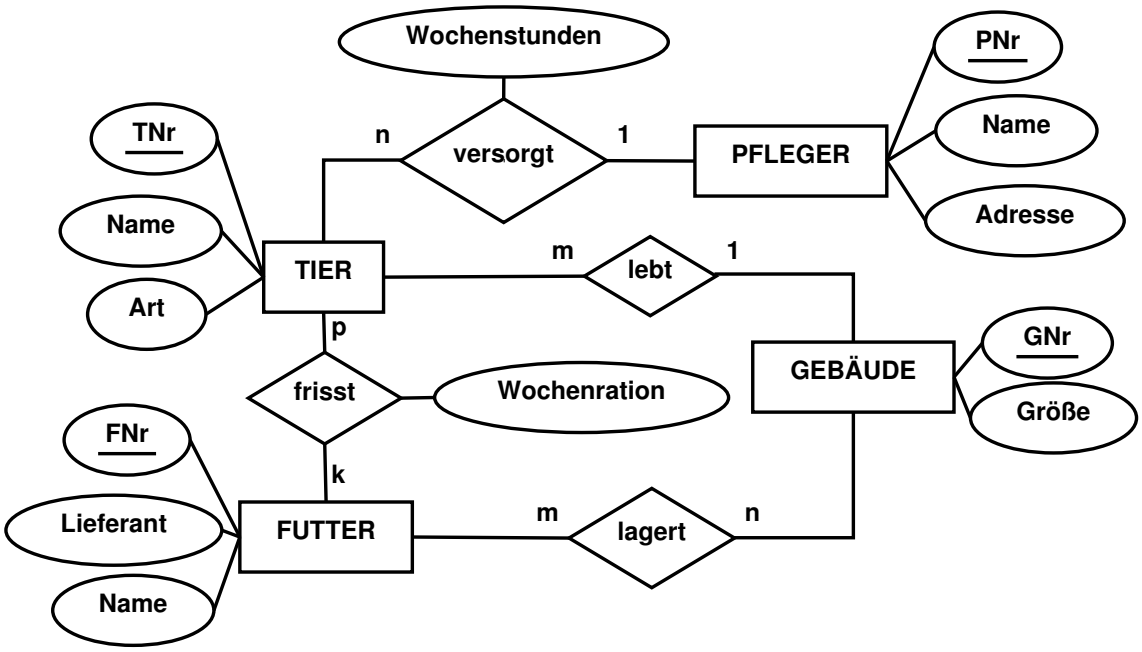
Für Book4You würde das gesamte Relationale Modell unter der Annahme, dass wir die one–to–many Beziehung durch Erweiterung der Tabelle MITARBEITER umgesetzt haben, sich so darstellen:

```
ABTEILUNG(AbtNr:AutoWert, AbteilungsName:Text(30))

PROJEKT(ProjektNr:AutoWert, Beschreibung:Text(50))

MITARBEITER(PersonalNr:AutoWert, Name:Text(30), Vorname:Text(30),
   Wohnort:Text(30), AbtNr:LongInteger, Position:Text(30))

ARBEITET_AN(ProjektNr:LongInteger, PersonalNr:LongInteger,
   Tätigkeit:Text(30), ProzAnteil:Integer)
```

Am Ende von Abschnitt 6.1 hatte ich ebenfalls die Frage gestellt, wie viele Tabellen wir nach der Umsetzung des ER–Diagramms von Book4You erhalten werden. Spätestens jetzt wissen wir, dass es niemals weniger als die Anzahl der Entitäten plus der Anzahl der many–to–many Beziehungen werden können und nicht mehr als die Anzahl der Entitäten plus der Anzahl aller Beziehungen zusammen. Die Antworten vier (bei Erweiterung der Tabelle MITARBEITER) oder fünf Tabellen (bei einer zusätzlichen Tabelle für die one–to–many Beziehung zwischen ABTEILUNG und MITARBEITER stattdessen) wären also goldrichtig gewesen.

6.5. Jetzt sind Sie wieder dran!

Der hiesige Zoo Tigga's Tausend–Morgen–Wald hat seine Datenbank mit dem im folgenden angegebenen ER–Diagramm geplant. Überführen Sie dieses Diagramm in das Relationale Modell:



Verwenden Sie dabei so wenige Tabellen wie möglich und geben Sie für jede Spalte einen geeigneten Microsoft Access Datentypen an. Die Attribute *Wochenration, Wochenstunden* und *Größe* dürfen Sie hierbei als ganzzahlige Datentypen ansetzen. Die Lösung zu dieser Aufgabe finden Sie im Kapitel B. Lösung

6.6. Zum Nachdenken

Betrachten Sie den theoretischen Datenankentwurf vom Beginn des Kapitels und seine Umsetzung in das relationale Modell noch einmal ganz in Ruhe.

- Lassen sich die Tabellen einer Datenbank zurück überführen in ein ER–Modell?
- Woran erkennt man eine many–to–many Beziehung im Relationalen Modell?