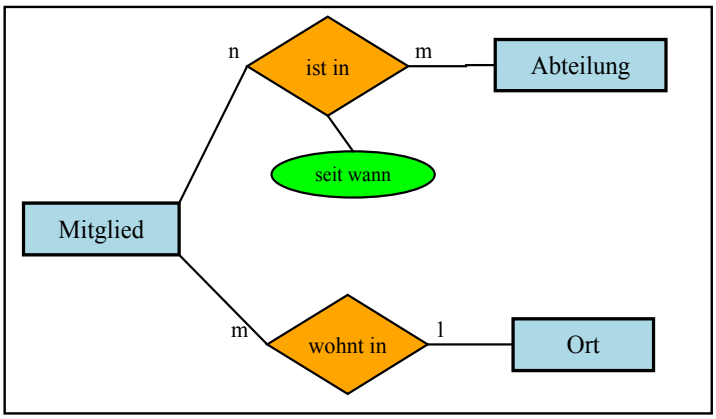
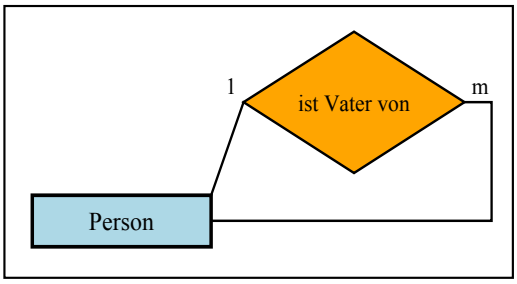
3.3 Entitätstypen und Beziehungstypen: das ER-Diagramm

(entity set, relationship)



Ein **ER-Diagramm** kennt zunächst nur zwei Bausteine: den **Entitätstyp** (*Mitglied, Abteilung, Ort*) und den **Beziehungstyp** (*wohnt in, ist in*). Mit diesen beiden Begriffen versucht man, ein Modell der Wirklichkeit zu erstellen, das als Grundlage für ein Datenbank-Modell dienen kann. Bei Bedarf hängt man noch die Attribute als Ellipsen an. Dieses „**semantische**[1] **Modell**“ ist noch so allgemein verständlich, dass es auch mit Nicht-Datenbankfachleuten diskutiert werden kann (z.B. mit dem Auftraggeber).

Entitäten sind Objekte der Wirklichkeit. Das können sein Personen, Gegenstände, aber auch abstrakte Dinge wie Kontobewegungen oder Aufträge. Entitäten stehen mit anderen in Beziehung. Eine Beziehung hat zwei Richtungen, die man als **Assoziationen** bezeichnet. Die Beziehung zwischen Mitglied Karl Meier und Abteilung Volleyball besteht aus der Assoziation *„Karl Meier ist in Volleyball.“* und der entgegengerichteten Assoziation *„Abt. Volleyball hat Karl Meier als Mitglied.“* Die **Kardinalitäten** einer Beziehung wurden schon weiter oben angesprochen, die wichtigsten Typen sind **1:1**, **1:m** und **n:m**.



Beziehungstypen verbinden nicht immer nur zwei Entitätstypen, es können auch mal drei sein oder nur einer, z.B. „*Person ist Vater von Person*“. Wie bei einer n:m-Beziehung ist es auch hier so, dass man das Modell nicht direkt in die Datenbank übernehmen kann.

Dass ein Entitätstyp **Attribute** hat (*Ort* hat *Name*, *PLZ*), versteht sich von selber. Mach dir klar, dass auch Beziehungstypen Attribute haben können: Die Beziehung „*Karl Meier ist in Volleyball.“* kann das Attribut „*seit wann*“ haben. Das Eintrittsdatum kannst du nicht zu *Abteilung* hinzufügen, denn es passt nicht zu allen Mitgliedern in dieser Abteilung. Es gehört auch nicht zu *Mitglied*, denn ein solches hat vielleicht mehrere Eintrittsdaten in verschiedene Abteilungen. Es gehört eindeutig zur Beziehung zwischen Mitglied und Abteilung: „*Karl Meier ist seit 12.2.1998 in Volleyball.“*

Will man ein Attribut ins ER-Diagramm einzeichnen, so benutzt man eine Ellipse. Wenn man allerdings alle Attribute in das Diagramm übernimmt, wird es leicht unübersichtlich. Dann benutzt man in der Praxis lieber eine eigene Attribut-Liste.

Erst wenn das ER-Diagramm (semantisches[1] Modell) korrekt ist und auch mit dem Auftraggeber abgesprochen ist, sollte man an die Umsetzung ins relationale Datenbankschema (logisches Modell) gehen.

---

[1] Semantik = Wortbedeutung, Bedeutungslehre

- **Füge in die *tbl_Zuordnung M-Abt* das Attribut *seitWann* ein (mit Wertebereich *Datum*).**
- **Gib jedem Datensatz der Tabelle für dieses Attribut einen Wert.**
- **Erstelle Abfragen der Art *qry_Mitglieder in Handball geordnet nach Eintrittsdatum.***
- **Erstelle eine *tbl_Person* und versuche selber herauszufinden, wie folgende Abfragen gelingen: *qry_Kinder von Klaus*, *qry_Vater von Ulla***

Erweiterung unseres Sportverein-Modells:

Jetzt wollen wir in unserer Sportverein-Datenbank weitere Informationen unterbringen: Die Abteilungen (Fußball, Handball ...) haben eine oder mehrere Mannschaften mit Spielern und einem Trainer, etwa die „E-Jugend“ beim Fußball und die „AlteHerren“ beim Volleyball.

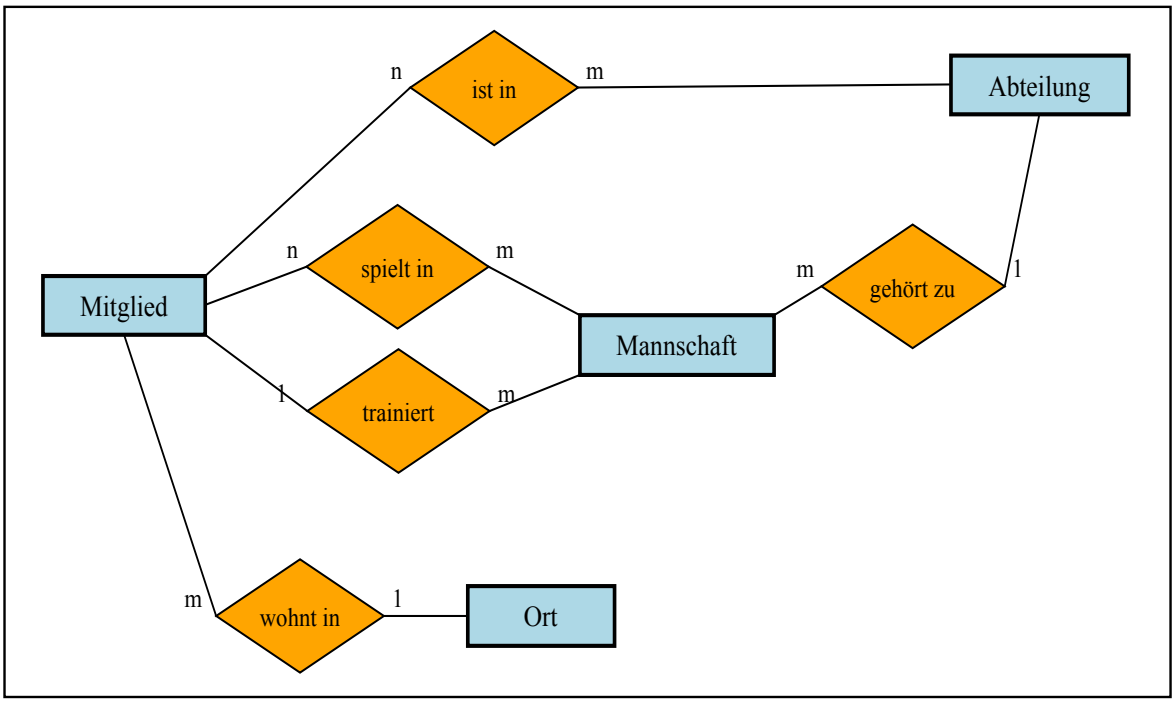
Unser ER-Diagramm erweitert sich um einen **Entitätstyp *Mannschaft*** mit den Attributen *Bez, Trainer, T-Ort, T-Beginn, T-Ende*. Welche **Beziehungstypen** ergeben sich zwischen *Mannschaft* und den bereits vorhandenen Entitätstypen?:

Jede Mannschaft ***gehört zu*** 1 Abteilung, jede Abteilung hat eine oder m Mannschaften. (m:1)

Jedes Mitglied ***spielt in*** einer oder m Mannschaften, jede Mannschaft hat n Spieler. (n:m)

Ein Mitglied ***trainiert*** (evtl.) eine oder m Mannschaften, jede Mannschaft hat 1 Trainer. (m:1)

Du siehst: Auf ganz selbstverständliche Art und Weise lässt sich dieser neue Entitätstyp in unser Diagramm integrieren. Und wenn du das erste Diagramm mit einem Vektorgrafik-Programm gezeichnet hast, lässt es sich sehr schnell umbauen und erweitern.

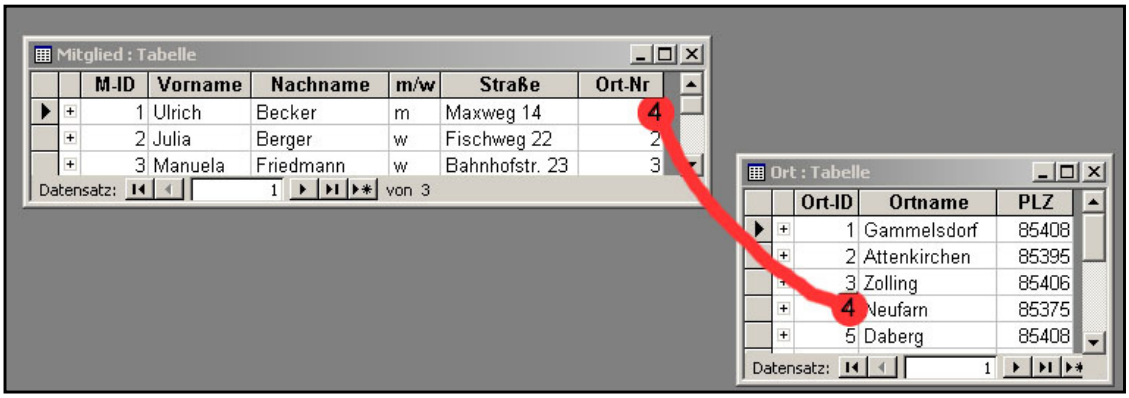


So: Jetzt wollen wir dieses semantische Modell des Sportverein in ein logisches Datenbank-Modell überführen. Dabei gibt es noch viel Grundsätzliches zu lernen.

3.4 Tabellen, Schlüssel: das relationale Datenbankschema

(Schema = Modell)

Eine Datenbank benutzt als Grundelemente **Tabellen** und **Beziehungen zwischen den Tabellen**. In der Datenbank-Sprache heißen Tabellen auch **Relationen**. Die Daten des Sportverein sind in verschiedenen Tabellen gespeichert. Und wenn man sie wieder erhalten oder ändern will, richtet man an die Datenbank eine Abfrage. Nur wenn die verschiedenen Tabellen richtig verknüpft sind und die Daten widerspruchsfrei sind, wird man erfahren können, in welcher Mannschaft Ulrich Becker spielt, wer seine Mitspieler sind oder wer mit ihm aus dem gleichen Ort kommt.



Eine **Tabelle** umfasst sowohl die Klasse (Entitätstyp) als auch alle gespeicherten Objekte (Entitäten, Datensätze) dieses Typs. Die erste Zeile der Tabelle enthält die Attribute als Spaltenüberschriften (=Feldnamen), für die im Entwurfsmodus auch die Wertebereiche (Text, Zahl, Datum, ...) festgelegt wurden. Jede weitere Zeile enthält ein Tupel von Attributwerten (=Datensatz). Jedes Tupel stellt eine Entität dar.

Jede Tabelle braucht einen **Primärschlüssel**. Das ist ein Attribut (oder wenn eines allein nicht ausreicht: eine Kombination von Attributen), wodurch jeder Datensatz eindeutig identifiziert wird. *Nachname* wäre als Primärschlüssel sicher ungeeignet, denn zum Wert 'Meier' gehören irgendwann mehrere Datensätze. Das Datenbanksystem würde sich aber weigern, einen zweiten Meier aufzunehmen. In der Tabelle *Ort* eignet sich die Kombination aus *Ortname* und *PLZ* als Primärschlüssel. Jedes Paar kommt in der Tabelle garantiert nur einmal vor. Der Nachteil: man kann einen solchen kombinierten Primärschlüssel nicht als einfachen Fremdschlüssel in eine andere Tabelle eintragen. Wenn das nötig ist wie bei *Ort*, so benutzt man stattdessen einen **künstlichen Schlüssel** *Ort-ID*, der die Datensätze einfach durchnummeriert und den Wertebereich *Autowert* hat. Dann kann man sagen: Ulrich Becker wohnt im Ort mit der *Ort-ID* 4. Diese Zahl 4 muss im Datensatz des Mitglied Ulrich Becker Platz finden. Deswegen braucht *Mitglied* ein Attribut *Ort-Nr* (**Fremdschlüssel**) mit Wertebereich *Zahl*.

In der Tabelle *Abteilung* unseres Sportverein (sh. nächste Seite) bräuchte man eigentlich keinen künstlichen Schlüssel *Abt-ID,* denn *Sportart* identifiziert jeden Datensatz eindeutig. Und man könnte diesen Primärschlüssel verknüpfen mit dem Fremdschlüssel *Abt* in der Tabelle *Mannschaft* und in der Tabelle *Zuordnung M-Abt*. Dort müsste das Attribut *Abt* vom Wertebereich *Text* sein so wie *Sportart*.

Es gibt auch Tabellen, die mehr als einen **Schlüsselkandidaten** haben. In der Tabelle *Fahrzeuge(Kennz, FahrgestNr, Hersteller)* eines Autoverleihers ist sowohl *Kennz* als auch *FahrgestNr* ein Schlüsselkandidat. Einer von beiden muss als Primärschlüssel festgelegt werden. Für einen Autohändler hingegen eignet sich *Kennz* nicht als Schlüssel: er kann auch Fahrzeuge ohne Kennzeichen an- und verkaufen.

Eine **1:m-Beziehung** erhält man durch Verknüpfen eines Primärschlüssels (auf der 1-Seite) mit einem Fremdschlüssel. Die Tabelle auf der 1-Seite heißt auch Eltern-Tabelle, die andere Kind-Tabelle.

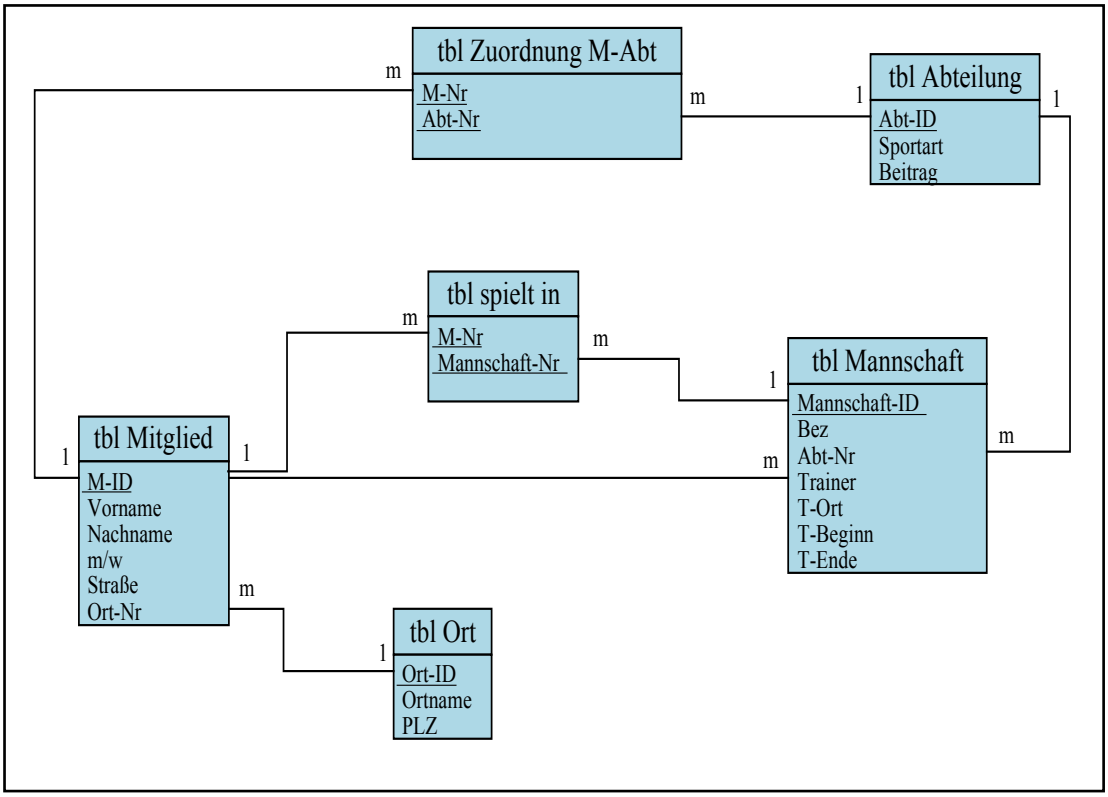
Eine **n:m-Beziehung** wird durch eine Zuordnungstabelle in zwei 1:m-Beziehungen aufgelöst (sh. 3.1). Diese Zuordnungstabelle stellt einen eigenen Entitätstyp dar, der jedoch nicht für sich allein existieren

kann. Man nennt ihn deswegen auch eine „schwache Entität“. Ihr Primärschlüssel ist ein kombinierter Schlüssel aus zwei Fremdschlüsseln.

Eine **1:1-Beziehung** kann man in einer einzigen Tabelle unterbringen, z.B. „*Mitglied hat Tel.nr.*“. Dann wird *TelNr* einfach ein Attribut. Aus Datenschutzgründen benutzt man gelegentlich zwei Tabellen mit unterschiedlichen Zugriffsrechten und verknüpft dann die beiden Primärschlüssel miteinander.

Wie du siehst, ist die Zahl der Tabellen im Datenbank-Modell nicht identisch mit der Zahl der Entitätstypen im ER-Modell. Und auch die Beziehungen kann man nicht eins zu eins übertragen.

Das Datenbank-Modell enthält die Tabellen mit Attributen (aber ohne Datensätze) und die Beziehungen zwischen den Tabellen. Da Mathematiker zu Tabellen auch Relationen sagen, nennt man dieses Modell **das relationale Datenbankschema**.



- Erstelle eine neue Version von ***Sportverein.mdb***, importiere die bisherigen Tabellen und implementiere dieses Modell. Gib mindestens zwei Mannschaften einer Abteilung ein, gib ihnen Spieler und einen Trainer.
- Erstelle Abfragen:
  qry_Alle Mannschaften mit Trainer, geordnet nach Abteilung
  qry_Alle Mannschaften mit Trainer und Zahl der Spieler
  qry_Alle Spieler einer Mannschaft ..., geordnet nach Wohnort

Wer scharf mitgedacht hat, wird sich die Frage gestellt haben, wie sich die **reflexive Beziehung** „*Person ist Vater von Person*“ im relationalen Datenbankschema darstellt. Man kann nämlich nicht eine Tabelle mit sich selber verknüpfen. Die Daten einer Tabelle gehören bereits durch die Tabellenstruktur zusammen. Die Väter sind in tbl_Person bereits enthalten. Erst für eine Abfrage „Alle Kinder von Heinz“ braucht man eine Verknüpfung. Man erstellt dann in der Abfrage von *tbl_Person* eine Kopie *tbl_Person1* und verknüpft die beiden Tabellen entsprechend.

3.5 Normalisierung

Beim Entwurf einer Datenbank müssen einige Regeln eingehalten werden. Die wichtigste Forderung ist die nach **Redundanzfreiheit**:

> Die Tabellen sollen so wenig redundante Daten wie möglich enthalten!

M-ID	Vorname	Nachname	m/w	Straße	PLZ	Ortname	Sportart	Beitrag
1	Ulrich	Becker	m	Maxweg 14	88405	Gammelsdorf	H	55,00 €
2	Ulrich	Becker	m	Maxweg 14	88405	Gammelsdorf	S	80,00 €
3	Julia	Berger	w	Fischweg 22	85395	Attenkirchen	H	55,00 €
4	Manuela	Fiedmann	w	Bahnhofstr. 23	85406	Zolling	L	55,00 €

Wie wir an diesem Beispiel in 3.2 bereits gesehen haben, führt Redundanz, also Mehrfachspeicherung von Daten, unweigerlich zu Widersprüchen. Man braucht bei einer Adressänderung nur einen Datensatz zu übersehen, und schon ist die Datenbank nicht mehr **konsistent** (=widerspruchsfrei). Eine inkonsistente Datenbank kann aber ihren Zweck kaum mehr erfüllen. Bisher haben wir die Daten mehr intuitiv auf mehrere Entitätstypen verteilt und kamen mit dem ER-Diagramm und dem Datenbankschema recht erfolgreich zu einer konsistent angelegten Datenbank. Dies gelingt aber nicht immer.

Dr. Edgar F. Codd hat 1970 ein Verfahren entwickelt, das die Redundanzfreiheit der Daten gewährleistet. Dieses Verfahren heißt **Normalisierung**. Dabei wendet man Regeln in einem dreistufigen Prozess auf die Tabellen an. Diese Stufen heißen **Erste, Zweite** und **Dritte Normalform (NF)**.

> 1NF: Die Attribute einer Tabelle müssen elementar (atomar) sein.

1NF erzwingt oft das **Anlegen zusätzlicher Spalten.** Elementar bedeutet, dass ein Attribut (Spalte) keine zusammengesetzten Werte enthalten darf. Eine einzelne Spalte *Adresse*, die zusammengesetzt wäre aus Straße, Hausnr., PLZ und Ort, widerspräche der 1NF. Obige Tabelle ist bereits in der 1NF, obwohl das Attribut *Straße* zusammengesetzt ist aus Straße und Hausnummer. Das Attribut kann man trotzdem elementar nennen, da man die beiden Teile niemals getrennt voneinander benötigt.

Auch eine Datumsangabe wie ’08.10.1980’ bezeichnet man als elementar, obwohl sie Tag, Monat und Jahr enthält, die man doch gelegentlich getrennt voneinander benötigt. In so einem Fall benutzt man dann spezielle Funktionen, die aus dem Datum den Tag, den Monat oder das Jahr ermitteln. Das Datum gehört trotzdem als Ganzes zusammen, sonst könnte man Personen nicht nach dem Alter ordnen.

Def.:

> **Attribut B hängt von Attribut A funktional ab**, wenn es zu jedem Wert von A genau einen Wert von B gibt. Schreibweise: **A → B**

Z.B. hängt *Wohnort* funktional von *Personalausweis-Nr* ab, oder *Beitrag* von *Sportart*, aber nicht *PLZ* von *Wohnort* (es gibt Orte mit mehreren PLZ). Beachte: Nichtschlüssel-Attribute hängen vom Schlüssel immer funktional ab, sonst wäre der Schlüssel falsch gewählt. Die 2. NF ist deswegen nur von Bedeutung in Tabellen, die einen kombinierten Schlüssel haben:

> 2NF: Die Nichtschlüssel-Attribute müssen vom Schlüssel voll funktional abhängen. (d.h. sie dürfen nicht bereits von einem Teil des Schlüssels abhängen.)

2NF und 3NF erzwingen in der Regel die **Aufteilung der Daten auf mehrere Tabellen.** Was kann in obiger Tabelle der Schlüssel sein? Eigentlich nur die Kombination aus *Vorname, Nachname* und *Sportart*. Der *Beitrag* wird aber bereits von *Sportart* allein bestimmt, der Name ist dafür unnötig. Wohnort und Geschlecht hingegen benötigen nicht die Sportart. Die Tabelle muss aufgeteilt werden.

Die 3. NF soll erzwingen, dass es unter den Nichtschlüssel-Attributen keine funktionalen Abhängigkeiten gibt. Dies ist allerdings nicht in jedem Fall nötig, deswegen braucht man noch einen neuen Begriff:

Def.: Wenn Attribut B von A funktional abhängt und Attribut C von B, so heißt **C transitiv abhängig von A** (falls nicht A von B abhängt). Kurz bedeutet transitiv: $A \rightarrow B \rightarrow C$ (wobei $B \not\rightarrow A$)

3NF: Die Nichtschlüssel-Attribute dürfen nicht transitiv vom Schlüssel abhängen.

Diese Forderung kann in der Praxis nicht immer streng geprüft werden. Man muss nämlich nicht nur die momentanen Werte der Attribute im Auge haben sondern auch die in Zukunft möglichen. In obiger Tabelle verlangt die 3NF die Auslagerung von *PLZ* in eine neue Tabelle, weil es zu *Straße/Ort* genau eine *PLZ* gibt. Deshalb ist *PLZ* transitiv abhängig vom Schlüssel *(Name/Vorname* oder *M-Nr)* über *Straße/Ort*:

M-Nr $\rightarrow$ *Straße/Ort* $\rightarrow$ *PLZ*, wobei *Straße/Ort* $\not\rightarrow$ *M-Nr.*

Oder betrachte folgende Tabelle:

Die Tabelle enthält Redundanz, weil K*Ort* von *KName* funktional abhängt. Genauer:

Auftrag-Nr $\rightarrow$ *KName* $\rightarrow$ *KOrt,*
wobei *KName* $\not\rightarrow$ *AuftragNr*

Deswegen erhalten wir zwei Tabellen:
Bestellung(Auftrag-Nr, Datum, KName) und *Kunde(KName, KOrt)*

Bestellung			
Auftrag-Nr	Datum	KName	KOrt
1	04.05.2003	Bauer	Augsburg
2	04.05.2003	Huber	Kempten
3	12.05.2003	Huber	Kempten
4	13.05.2003	Müller	Augsburg

Ein Beispiel, in dem trotz Abhängigkeit von Nichtschlüssel-Attributen die 3NF nicht verletzt ist, findest du in der Aufgabe mit den Fahrzeug-Tabellen am Ende des Kapitels.

Beachte: Eine Tabelle, die außer dem Schlüssel nur ein einziges Attribut hat, erfüllt immer die Forderung von 3NF.

Ich zeige noch einmal im Einzelnen, wie man ausgehend von den Rohdaten eines Sportverein durch Normalisierung zu den vier Tabellen *Mitglied, Ort, Abteilung* und *Zuordnung M-Abt* kommt:

Sportverein (Rohdaten)				
M-Nr	Name	Ge	Adresse	Sportarten(Beitrag)
1	Heinz Gruber	m	Hauptstraße 7, 86343 Königsbrunn	F(50.-) H(40.-)
2	Udo Meier	m	Bierweg 35, 86199 Augsburg	F(50.-) L(55.-)
3	Frieda Baum	w	Lerchenweg 4, 86161 Augsburg	L(55.-)
4	Ali Berg	m	Kirchgasse 7, 86343 Königsbrunn	F(50.-) V(40.-)

Die 1NF verlangt die Atomisierung der Attribute:

Sportverein (1NF)								
M-Nr	Vorname	Nachname	Ge	Straße	Ort	PLZ	**Sportart**	Beitrag
1	Heinz	Gruber	m	Hauptstraße 7	Königsbrunn	86343	F	50
1	Heinz	Gruber	m	Hauptstraße 7	Königsbrunn	86343	H	40
2	Udo	Meier	m	Bierweg 35	Augsburg	86199	F	50
2	Udo	Meier	m	Bierweg 35	Augsburg	86199	L	55
3	Frieda	Baum	w	Lerchenweg 4	Augsburg	86161	L	55
4	Ali	Berg	m	Kirchgasse 7	Königsbrunn	86343	F	50
4	Ali	Berg	m	Kirchgasse 7	Königsbrunn	86343	V	40

Die Atomisierung bringt zunächst Redundanz, die in den nächsten Schritten beseitigt wird. Die *M-Nr* allein ist für diese Tabelle kein Identifizierungsschlüssel mehr, sondern die Kombination aus *M-Nr* und *Sportart*.

Die 2NF verlangt, dass jedes Nichtschlüssel-Attribut nicht bereits von einem der beiden Schlüssel-Attribute funktional abhängt. *Vorname, Nachname, Ge*, ... (alle personenbezogenen Daten) hängen nicht von *Sportart* ab sondern nur von *M-Nr*. Also müssen diese in eine eigene Tabelle, deren Schlüssel *M-Nr* ist.

Mitglied (2NF)						
M-Nr	Vorname	Nachname	Ge	Straße	Ort	PLZ
1	Heinz	Gruber	m	Hauptstraße 7	Königsbrunn	86343
2	Udo	Meier	m	Bierweg 35	Augsburg	86199
3	Frieda	Baum	w	Lerchenweg 4	Augsburg	86161
4	Ali	Berg	m	Kirchgasse 7	Königsbrunn	86343

Abteilung (1NF)		
M-Nr	**Sportart**	Beitrag
1	F	50
1	H	40
2	F	50
2	L	55
3	L	55
4	F	50
4	V	40

Die zwei Tabellen enthalten exakt die gleiche Information wie die vorige Tabelle. Da *Mitglied* nur ein Schlüsselattribut hat, ist sie automatisch in 2NF.

Allerdings ist in der Tabelle *Abteilung(1NF) Beitrag* nicht abhängig von *M-NR* sondern nur von *Sportart*. *Beitrag* muss in eine eigene Tabelle mit dem Schlüssel *Sportart*. Danach ist *Abteilung* in 3NF, da sie nur noch ein Nichtschlüssel-Attribut hat, erst recht die Zuordnungstabelle *M-Abt*.

Abteilung (3NF)	
Sportart	Beitrag
F	50
H	40
L	55
V	40

M-Abt (3NF)	
M-Nr	**Sportart**
1	F
1	H
2	F
2	L
3	L
4	F
4	V

Jetzt müssen wir noch die Forderung von 3NF auf *Mitglied* anwenden: PLZ hängt von Ort und Straße ab (wobei man von Ort/Straße nicht auf M-Nr schließen kann), PLZ ist also transitiv abhängig von M-Nr. Wir verbannen *PLZ* in eine neue Tabelle. Da *Ortname* nicht eindeutig eine *PLZ* festlegt und umgekehrt auch nicht, hätten wir wieder einen kombinierten Schlüssel. Da wir diesen aber als Fremdschlüssel in die Tabelle Mitglied eintragen müssen, nehmen wir einen künstlichen Schlüssel *Ort-ID* vom Typ „Autowert“.

Mitglied (3NF)					
M-Nr	Vorname	Nachname	Ge	Straße	**Ort-Nr**
1	Heinz	Gruber	m	Hauptstraße 7	1
2	Udo	Meier	m	Bierweg 35	2
3	Frieda	Baum	w	Lerchenweg 4	3
4	Ali	Berg	m	Kirchgasse 7	1

Ort (3NF)		
Ort-ID	Ortname	PLZ
1	Königsbrunn	86343
2	Augsburg	86199
3	Augsburg	86161

M-Abt (3NF)	
M-Nr	**Sportart**
1	F
1	H
2	F
2	L
3	L
4	F
4	V

Abteilung (3NF)	
Sportart	Beitrag
F	50
H	40
L	55
V	40

Diese vier Tabellen enthalten exakt die Information der vorgegebenen Rohdaten. Sie enthalten keine Redundanz mehr, und es ist sichergestellt, dass die Datenbank von ihrer Struktur her konsistent sein wird.

Du siehst, dass man durch Normalisierung ganz ohne Entitäten und ER-Diagramm zur gleichen Datenstruktur gelangt. Auch die Beziehungen der Tabellen sind direkt an den gleichnamigen Attributen zu erkennen. Wenn du eine Datenbank neu entwirfst, bleibt es dir überlassen, ob du mit einem ER-Diagramm beginnst und deine Tabellen anschließend auf die Normalformen prüfst oder ob du mit der Normalisierung der Rohdaten anfängst und daraus das ER-Diagramm erhältst.

Das Ergebnis sollte jedenfalls ein Datenbank-Entwurf sein, dessen Tabellen sich in der dritten Normalform befinden.

Als zweites Beispiel wird im Unterricht das folgende durchgesprochen werden. Aus den Rohdaten soll man allein durch Normalisierung zu einem korrekten Datenbank-Entwurf gelangen.

Unterrichtsverteilung (Rohdaten)		
Lehrer	Kürzel	Klasse, Fach, Stunden
Hafner Bertram (Klassleitung 5c, Raum I53)	Hn	10c, Inf, 2 11c, Ph, 3 11c, Inf, 2 5c, M, 4 ...
Kropf Alexandra (Klassleitung 11c, Raum E60)	Kr	11c, M, 5 ...
Chmielewski Andreas (Klassleitung 10c, Raum I34)	Ch	10c, E, 4 ...
...	...	...

Wenn mit Papier und Bleistift der Entwurf gelungen ist, soll auch ein ER-Diagramm gezeichnet werden. Anschließend kann die Datenbank in Access erstellt werden.

- ► Erstelle anschließend Abfragen: alle Klassen mit Klassenleiter und Klassenzimmer, alle Fachlehrer der 11c mit Kürzel und Fach.
- ► Wo kann der Fachraum (z.B. 11c, Ph in I26) eingebaut werden?

- ► Welcher NF widerspricht die Speicherung von Geburtsdatum und Alter von Personen?

- ► Die Tabelle ***tbl_spielt_in*** des Sportvereins hat einen kombinierten Schlüssel aus ***M-Nr*** und ***Mannschaft-Nr***. Der Verein möchte auch festhalten, seit wann ein Mitglied in einer Mannschaft spielt, wie alt es ist und welche Funktion es in der Mannschaft hat. Teste nebenstehende Tabelle auf die Normalformen.

tbl_spielt in				
M-Nr	**Mannschaft-Nr**	seit wann	Alter	Funktion
1	1	04.05.2001	23	Sturm
3	3	11.07.1999	19	Verteidigung
6	1	12.01.2000	22	Sturm
4	1	08.11.2000	22	Torwart

► Wende auf folgende Tabelle mit Rohdaten wie oben vorgeführt schrittweise die Normalisierung an. Schreibe die Tabellen mit ihren Daten auf! Der Informationsgehalt soll exakt mit dieser Tabelle übereinstimmen.

Bücherausleihe (Rohdaten, bisher auf Karteikarten)			
Name	Adresse	Datum	Bücher
Egon Bauer	Hauptstraße 11, 86343 Königsbrunn	12.5.03 5.6.03	Ro133, Meier, Der Hecht In17/1, Staas, MySQL
Gunda Altmann	Sandweg 35, 86199 Augsburg	4.4.03	Ro205, Meier, Hinter der Mauer
Elke Bast	Holzweg 4, 86161 Augsburg	12.5.03 12.5.03	Ph14, Born, Physik und Philosophie Ma5/3, Degen, Analysis 1
Herbert Soll	Kirchplatz 7, 86343 Königsbrunn		

Erstelle hinterher das zugehörige ER-Diagramm und das Datenbankschema.
Erweitere dieses Modell anschließend so, dass von jedem Buch auch mehrere Exemplare (z.B. In17/1, In17/2, In17/3) verwaltet werden können. Dabei soll das Buch (Autor, Titel, ISBN) nicht redundant gespeichert werden. Jedes Exemplar hat außer der Signatur ein Kaufdatum.

► Wende die Normalisierungsregeln an. Überführe diese Tabelle nacheinander in die erste, zweite und dritte Normalform. Erstelle hinterher ein ER-Diagramm.

Volkshochschule (Rohdaten)						
KursNr	Kurs	Dozent	Tel	Tag	Zeit	Raum
SF217	Französisch 1 intensiv	LeClerc	08212345	Mo Do	19:00 19:00	E12 138
SF218	Französisch 2 intensiv	LeClerc	08212345	Di Di	17:00 18:00	E12 E12
SF219	Französisch 2	Simpson	08213451	Di	19:00	138
MG13	Gitarre Anf.	Rodriguez	08213541	Mo	17:00	139
MG113	Gitarre Anf.	Hafner	08203528	Mo	19:00	139
KA55	Klimt	Coulin	08215544	Mi	17:00	E14

► Nimm deinen Stundenplan und übertrage ihn in eine datenbanktaugliche Tabelle. Welche Attribute bilden den Schlüssel der Tabelle? In welcher NF befindet sie sich?

Wie kannst du noch Beginn und Ende jeder Stunde und den jeweiligen Lehrer mit einbauen? Die drei Normalformen sollen erfüllt sein! Zeichne ein relationales Datenbankschema.

Wie musst du das Schema ändern, wenn in einer Schulstunde zwei verschiedene Fächer eingetragen werden sollen (z.B. Latein/Französisch)?

► Überprüfe die beiden Tabellen auf 3NF. In beiden Tabellen gibt es eine Abhängigkeit von Nichtschlüssel-Attributen. Aber nur in einer liegt eine transitive Abhängigkeit vor, sodass man nur diese Tabelle aufspalten muss.

Fahrzeug		
Kennzeichen	Fabrikat	Hersteller
A-CW27	Golf III	VW
A-BM2332	Sprinter	DaimlerChrysler
A-XY1010	Golf III	VW
A-OH234	Astra	Opel
A-BE303	Golf IV	VW

Fahrzeug		
Kennzeichen	Fahrgest.Nr	Hersteller
A-CW27	W...2196	VW
A-BM2332	W...9270	DaimlerChrysler
A-XY1010	W...8244	VW
A-OH234	W...3062	Opel
A-BE303	W...1198	VW

► Untersuche die Tabelle auf 3NF:

Lieferant			
Lieferant-ID	Name	Telefon	Fax

► Wende auf folgenden Auszug eines Chemikalien-Katalogs die Normalisierung an und erstelle hinterher auch ein ER-Diagramm.

Chemikalien (Auszug aus einem Katalog)					
FN	Formel	MG	Bezeichnung	BRN	Preise
12540	C6H6	78.1147	Benzene, puriss.	969212	5,0ml:48,50; 10ml:83,30
12549	C6H6	78.1147	Benzene, for HPLC	969212	1,0l:48,50; 2,5l:101,50
12550	C6H6	78.1147	Benzene, ACS	969212	500ml:24,70; 1,0l:48,20; 2,5l:103,40
12560	C6H6	78.1147	Benzene, purum	969212	1,0l:23,60; 2,5l:42,50
82700	C5H5N	79.1023	Pyridine, for UVspectroscopy	103233	250ml:92,10; 1,0l:343,40
82701	C5H5N	79.1023	Pyridine, for sequence analysis	103233	250ml:177,90
56380	C5H5NO	95.1017	2-Hydroxypyridine, pract.	105757	100g:62,30; 500g:289,10
56390	C5H5NO	95.1017	3-Hydroxypyridine, purum	105699	25g:17,80: 100g:60,20
56400	C5H5NO	95.1017	4-Hydroxypyridine, techn.	105800	25g:35,60: 100g:130,90
82811	C5H5NO	95.1017	Pyridine-N-oxide, pract.	105257	100g:37,90; 500g:159,00

Interessant ist, dass die Normalisierung auch dann gelingt, wenn man nicht in jedem Fall die Bedeutung der Attribute kennt. Es reicht aus festzustellen, ob es zu jedem Wert von Attribut A genau einen Wert von B gibt oder ob es (jetzt oder später) mehrere Werte von B geben kann.