3. Datenbanken

Als Grundlage für diesen Unterrichtsentwurf diente das Datenbank-Skript von Matzke/Lehmann. Als Datenbank-Programm wird MS-Access benutzt.

Im ersten Abschnitt möchte ich dich an die Denkweise in Datenbanken heranführen. Wir werden eine erste Mini-Datenbank erstellen und versuchen, aus ihr Informationen abzufragen.

3.1 Einführendes Beispiel

Wenn man von Datenverwaltung spricht, dann handelt es sich dabei immer um Daten, die in Tabellen gespeichert werden: Stundenplan, Telefonbuch, Punktetabellen bei den Bundesjugendspielen, Bestelllisten, Lagerlisten, Bauteillisten, Kontoauszüge, Kataloge, Auktionsartikel ...

Für einen kleinen Sportverein wollen wir in einer Datenbank die Namen der Mitglieder und ihre Sportarten verwalten:

Nachname	Vorname	Sportart
Binner	Sandra	Handball
Huber	Stefan	Fussball, Handball
Zuber	Tobias	Fussball, Handball
Mühsam	Christian	Fussball
Wagner	Frank	Handball, Basketball
Huber	Veronika	Basketball
...		

Hat man so eine Tabelle, dann kann man ihr leicht Daten entnehmen wie:

Welche Sportarten betreibt Huber?

Welche Personen sind beim Fussball?

Welche Personen sind beim Fussball und beim Handball?

Wir legen diese Tabelle nun in Access an und versuchen anschließend, die drei Fragen mit Datenbankmitteln zu beantworten.

- **Erzeuge eine neue Datenbank mit dem Namen *Verein1.mdb***
- **Erstelle die Tabelle in Entwurfs-Ansicht.**
- **Schließe die Tabelle und gib ihr dabei den Namen *tblPerson*.**
- **Öffne die Tabelle in Datenblatt-Ansicht und fülle sie mit obigen Werten.**

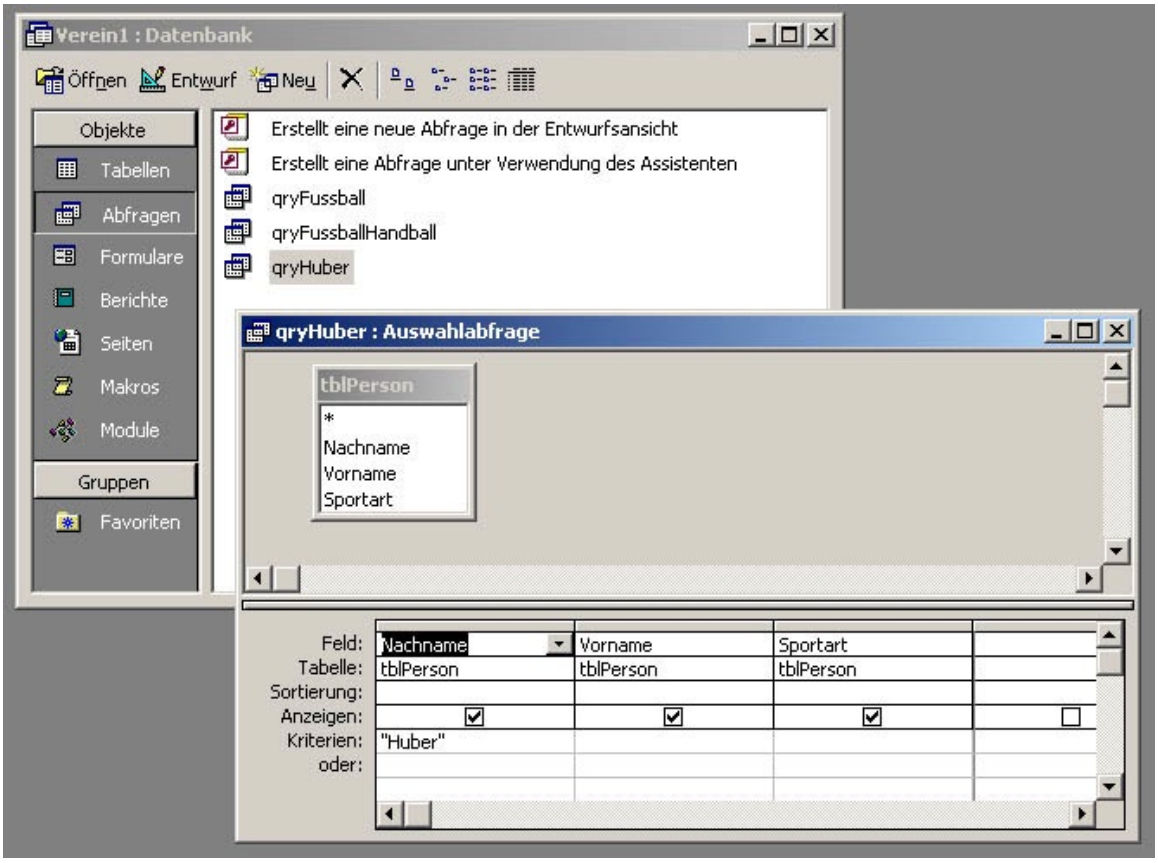
Entwurfs-Ansicht

Datenblatt-Ansicht

In Datenbanken gibt es außer Tabellen noch Abfragen, Formulare und Berichte. Damit man den Überblick nicht verliert, benutzt man für diese vier verschiedenen Datenbank-Objekte folgende Vorsilben:

Tabelle (table):	tbl
Abfrage (query):	qry
Formular (form):	frm
Bericht (report):	rpt

Aus diesem Grund haben wir unsere Tabelle *tblPerson* genannt. Jetzt wollen wir an unsere Mini-Datenbank die obigen Fragen stellen. Als erstes soll uns die Datenbank die Sportarten von Huber nennen. Wir erstellen eine **Abfrage** in Entwurfsansicht (der wir nachher beim Speichern den Namen *qryHuber* geben). Wir fügen zuerst die Tabelle, die wir abfragen wollen, hinzu. Wie wir die Abfrage formulieren, siehst du in der Abbildung, die Datenblatt-Ansicht liefert die beiden Datensätze mit Nachname 'Huber', und dies wiederum in Form einer Tabelle. Diese Tabelle unterscheidet sich aber von *tblPerson*: sie wird nicht gespeichert sondern in dem Augenblick erzeugt, in dem wir die Abfrage *qryHuber* in der Datenblatt-Ansicht öffnen.



Entwurfs-Ansicht

Datenblatt-Ansicht

- **Erstelle auf die gleiche Weise eine Abfrage *qryFussball* und eine Abfrage *qryFussHandball*. Überrascht dich das Ergebnis? Diskutiere deine Beobachtung mit deinen Nachbarn.**

Offenbar haben wir die *tblPerson* noch nicht korrekt angelegt. Wahrscheinlich würdest du jetzt gern eine weitere Spalte *Sportart2* anhängen. Dies löst das Problem aber nicht grundsätzlich! Denn es könnte jemand ja auch 3 oder 4 Sportarten betreiben. Gibt es einen anderen Ausweg, wenn wir in die Spalte *Sportart* nicht mehr als eine Sportart eintragen wollen?

- **Ändere deine Tabelle so ab, wie du sie hier siehst. Erstelle anschließend nochmal diese drei Abfragen von oben und vergleiche die Ergebnisse.**

tblPerson : Tabelle

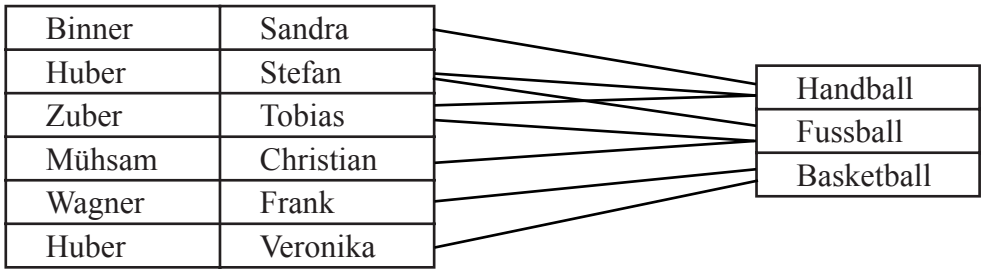
Nachname	Vorname	Sportart
Binner	Sandra	Handball
Huber	Stefan	Fussball
Huber	Stefan	Handball
Zuber	Tobias	Fussball
Zuber	Tobias	Handball
Mühsam	Christian	Fussball
Wagner	Frank	Handball
Wagner	Frank	Basketball
Huber	Veronika	Basketball

Datensatz: 10 von 10

Eine erste Forderung an eine konsistente (=widerspruchsfreie) Datenbank haben wir damit erfüllt: Die Werte der Attribute müssen '**atomar**' sein, sie dürfen nur eine einzige Information enthalten. Allerdings handeln wir uns damit das nächste Problem ein: wir müssen viele Mitglieder mehrfach speichern. Diese Mehrfachspeicherung nennt man Redundanz und muss unbedingt vermieden werden. Die Tabellen müssen **redundanzfrei** sein.

Welcher Ausweg bleibt?

Die einzige denkbare Möglichkeit besteht darin, die Person von der Sportart zu trennen und die Beziehung ***Person betreibt Sportart*** anders herzustellen als dadurch, dass man sie in eine Zeile schreibt.



Die beiden Tabellen sind jetzt sehr einfach geworden. Sie sind atomar und redundanzfrei. Aber diese Linien lassen sich so nicht in die Datenbank eingeben. Deswegen gibt man den Datensätzen in beiden Tabellen eine eindeutige Nummer. Diese nennt man meist ID und bezeichnet dieses Attribut als **Primärschlüssel**. Für jede Linie trägt man ein Nummernpaar in eine **Zuordnungstabelle** ein.

PersonID	Nachname	Vorname
1	Binner	Sandra
2	Huber	Stefan
3	Zuber	Tobias
4	Mühsam	Christian
5	Wagner	Frank
6	Huber	Veronika

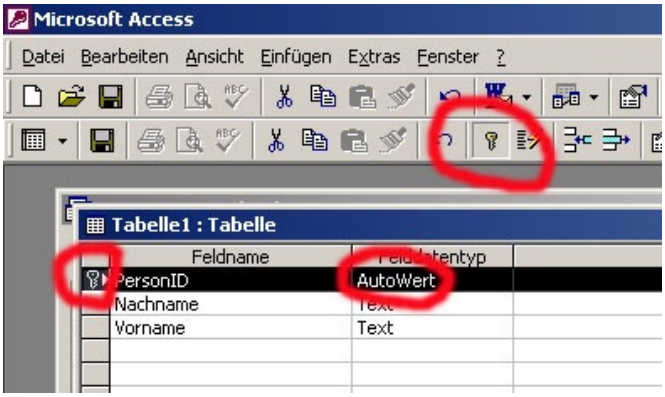
1	1
2	2
2	1

SportartID	Sportart
1	Handball
2	Fussball
3	Basketball

- **Vervollständige die Zahlenpaare in der Zuordnungstabelle.**

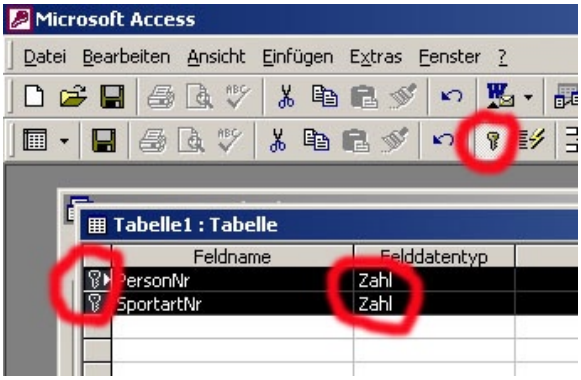
- **Lösche deine *tblPerson* (oder öffne eine ganz neue Datenbank) und erstelle die *tblPerson* neu.**

Das Primärschlüssel-Attribut muss vom Typ 'Autowert' sein, das bedeutet, Access verwaltet die Nummern selber. Nach dem Markieren der Zeile *PersonID* kann man dieses Attribut als Primärschlüssel festlegen.

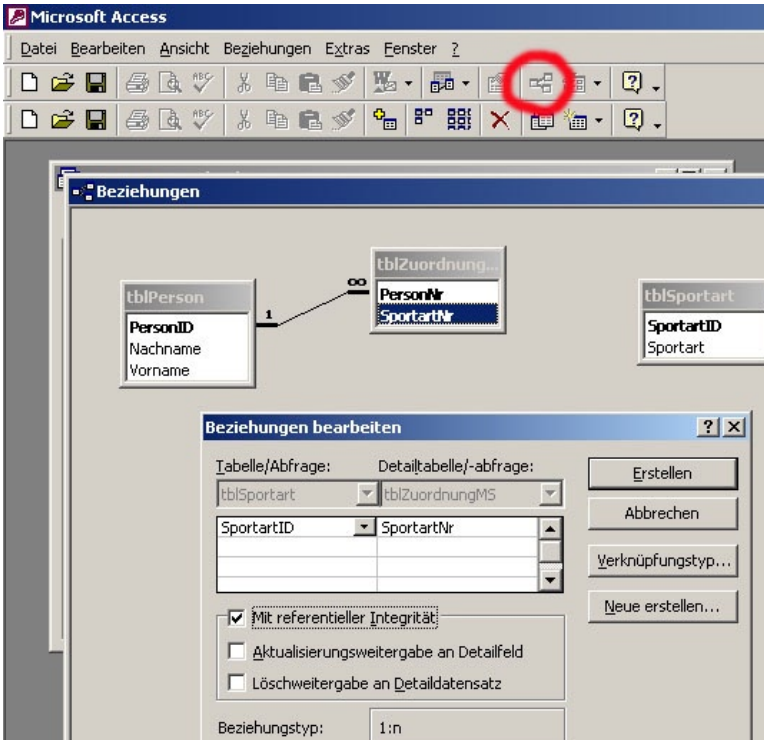


- **Gib anschließend die Namen in das Datenblatt ein.**
- **Erstelle auf die gleiche Weise die *tblSportart*.**
- **Erstelle nun die *tblZuordnungPS*. *Beachte dazu folgende Hinweise!***

Die Zuordnungstabelle braucht keinen Autowert-Schlüssel. Solche Nummern sind ja nur dazu da, dass man sie in eine andere Tabelle eintragen kann. Trotzdem muss die Datenbank jeden Datensatz der Tabelle eindeutig identifizieren können. Jede Tabelle braucht also einen Primärschlüssel. Da jedes Zahlenpaar der *tblZuordnungPS* bestimmt nur einmal vorkommt, kann man beide Attribute gemeinsam als **kombinierten Primärschlüssel** festlegen. Die beiden Attribute müssen vom Typ Zahl sein, weil dort ja die Person-Nummer und die Sportart-Nummer eingetragen werden. Jedes dieser beiden Attribute für sich nennt man einen **Fremdschlüssel**, das bedeutet: Die Werte in diesen Spalten enthalten Primärschlüssel-Werte einer anderen Tabelle. Damit wir Primär- und Fremdschlüssel bereits am Namen unterscheiden können, benenne ich Primärschlüssel immer mit 'ID' und Fremdschlüssel immer mit 'Nr'.



Wenn die drei Tabellen erstellt und mit Daten gefüllt sind, muss man der Datenbank mitteilen, wie die Tabellen miteinander in **Beziehung** stehen.



- **Schließe die Tabellen und öffne das Beziehungen-Fenster. Füge alle drei Tabellen hinzu. Verbinde mit der Maus *PersonID* und *PersonNr.* Markiere 'Referentielle Integrität' (damit wird eine Prüfung eingeschaltet) und erstelle die Beziehung. Das gleiche machst du mit der Beziehung zwischen *SportartID* und *SportartNr*.**

Jetzt haben wir eine korrekt angelegte Datenbank, die alle weiteren Aufgaben meistern kann. Vielleicht stört es dich, dass wir noch sehr unkomfortabel die Werte und Nummern in die Tabellen eintippen müssen. Dies verbessern wir später, wenn wir Formulare erstellen.

Aufgaben:

Erstelle eine Abfrage, die folgendes ermittelt:

a) alle Namen und Sportarten zum Nachnamen 'Huber'

b) die Namen aller Fußballer

c) alle Personen mit Fußball oder (auch) Handball

d) alle Personen mit Fußball und Handball

e) die Anzahl der Fußballer

f) zu jeder Sportart die Zahl der Personen

g) die Zahl der Sportarten von Frank Wagner

h) Erstelle eine Abfrage, die die Sportarten alphabetisch geordnet auflistet und jeweils dazu in alphabetischer Reihenfolge die Personen.

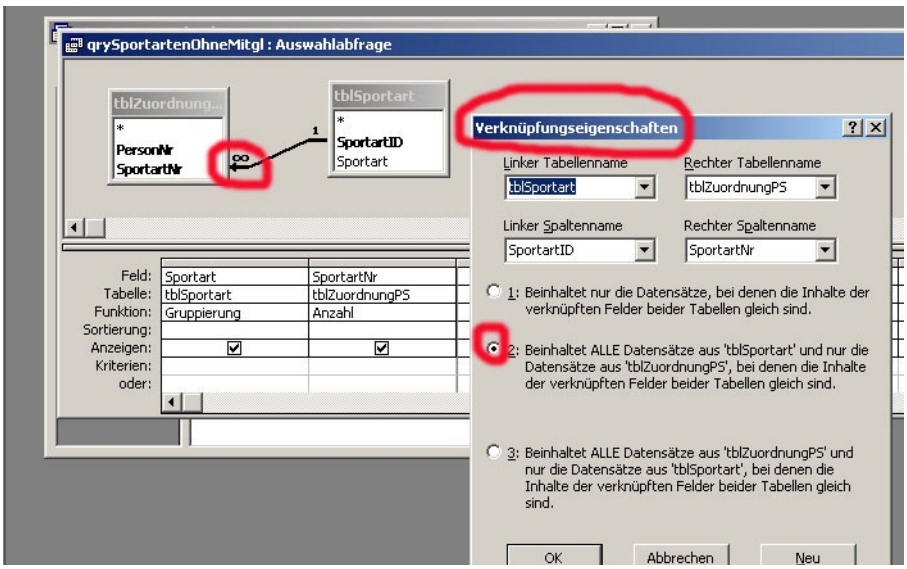
i) Der Sportverein bietet neuerdings auch Tennis an, hat aber noch keine gemeldeten Personen für Tennis. Gib in *tblSportart* Tennis ein. Erstelle eine Abfrage, die angibt, welche Sportart noch von keiner Person betrieben wird.

Du merkst schnell, dass es nicht einfach ist, die Abfragen richtig anzulegen. Und das, obwohl wir bisher nur 3 kleine Tabellen benutzen. Hier noch ein paar Tipps:

zu e): Anzahl der Fussballer: Nimm die zwei Attribute *Sportart* und *Nachname* in die Abfrage auf. Das Kriterium bei *Sportart* lautet „Fussball". Wenn du jetzt nicht alle Namen aufgelistet haben willst, die zu Fussball gehören, sondern ihre Anzahl, verwende in der Zeile 'Funktionen' die Funktion ***Anzahl***. (Die Zeile Funktionen schaltet man in der Symbolleiste mit ∑ ein oder aus.)

zu h): Benutze zum alphabetischen Ordnen der Datensätze in der Zeile 'Sortieren' die Einträge *aufsteigend* oder *absteigend*. Verwendet man dies in mehr als einer Spalte, so wird zuerst nach der linken Spalte sortiert.

zu i): Wenn du so vorgehst wie in e), dann wird Tennis nicht aufgelistet, denn es gibt ja keine Person, die Tennis spielt. Datensätze, die nicht vorkommen, werden auch nicht gezählt. Für dieses Problem gibt es die Möglichkeit, die Verknüpfungseigenschaften im Abfragefenster zu verändern. Gehe so vor: Benutze nur die Tabellen *tblSportart* und *tblZuordnungPS*. Benutze die Attribute *Sportart* und *SportartNr*, denn wir wir wollen nachsehen, ob die *SportartID* von Tennis in *SportartNr* eingetragen ist. Mit der rechten Maustaste auf die Verknüpfung erhältst du das Fenster 'Verknüpfungseigenschaften'. Hier kannst du 'alle Datensätze aus tblSportart und ...' wählen. So erhältst du nicht nur die Datensätze, die in beiden Tabellen vorkommen, sondern auch Tennis.



3.2 Ein erster grundlegender Überblick

Neue Fachbegriffe:

Ein **Objekt** in der Datenbank heißt **Entität** und ist identisch mit einem **Datensatz** in einer Tabelle (z.B. ' 3 Julia Berger w Fischweg 22 ...', kurz: das Objekt mit der *M-ID* 3).

Die **Klasse** heißt **Entitätstyp.** Sie gibt die Struktur für die Objekte vor. In diesem Beispiel hat sie die **Attribute** *M-ID, Vorname, Nachname, ...*

Eine **Entitätsmenge** umfasst alle gespeicherten Objekte (Entitäten) gleichen Typs, z.B. alle Mitglieder des Sportvereins; diese befinden sich alle in einer Tabelle.

Eine **Tabelle** enthält sowohl die Klasse mit den Attributen (=Feldnamen=Spaltenüberschriften) als auch alle Objekte der Klasse mit ihren Attributwerten (=Datensatz=Zeile=Tupel). Eine Tabelle nennt man auch **Relation**.

Klasse

Objekt

Rohtabelle : Tabelle

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 €

Ausgehend von dieser jetzt etwas ausführlicheren Tabelle mit Rohdaten von Vereins-Mitgliedern soll versucht werden, den Sportverein mit einer großen Zahl von Mitgliedern zu verwalten. In der ersten Zeile der Tabelle stehen die **Feldnamen (Attribute)** der zu speichernden Daten. Jede Zeile darunter stellt einen **Datensatz** dar.

Wie wir im Einführungsbeispiel gesehen haben, stößt man sehr schnell auf unüberwindbare Probleme, wenn man die Daten so wie abgebildet in eine Datenbanktabelle eingibt: Betreibt ein Mitglied mehr als eine Sportart, werden die gleichen Personendaten mehrfach gespeichert. Soll eine Adresse geändert werden, so muss man das in mehreren Datensätzen tun. Ebenso gehören zu einer Sportart mehrere Mitglieder, in deren Datensatz jedesmal der Mitgliedsbeitrag gespeichert wird. Auch die Postleitzahl von Gammelsdorf wird mehrfach gespeichert. Diese Mehrfachspeicherung von Daten nennt man **Redundanz**. Will man Daten ändern, einfügen oder löschen, so treten unweigerlich Fehler auf. Die Tabelle enthält sehr schnell Widersprüche, die Datenbank ist dann **inkonsistent** (widersprüchlich) und deswegen unbrauchbar.

Bei der Erstellung einer Datenbank musst du besonders darauf achten, Redundanz zu vermeiden. Deswegen müssen wir auch das Vereins-Modell und seine Struktur neu durchdenken. Wie das zwischen Mitglied und Abteilung funktioniert, haben wir im vorigen Kapitel schon gelernt. **Wir teilen die vorhandenen Datenfelder auf mehrere Klassen (also mehrere Tabellen) auf**. Dabei kommen zusammengehörende Daten jeweils in eine Tabelle:

Abteilung (Sportart, Beitrag)

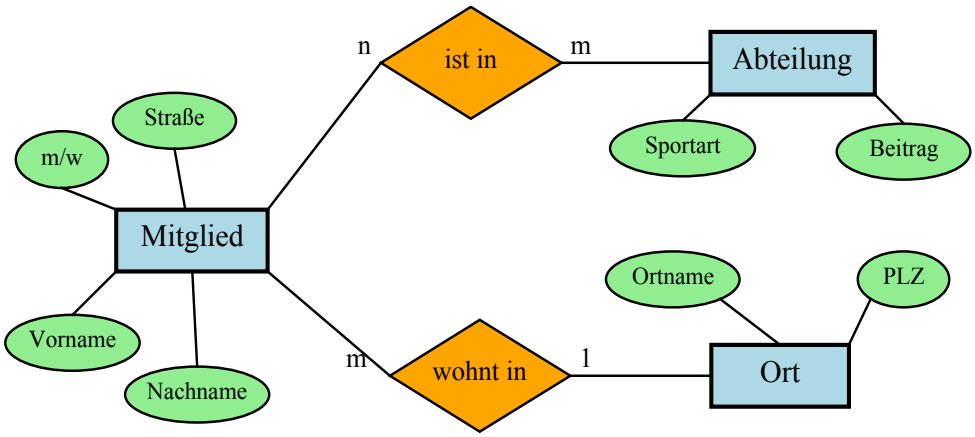
Ort (Ortname, PLZ)

Mitglied (Vorname, Nachname, m/w, Straße)

- **Warum gehört *Straße* nicht in die Tabelle *Ort*?**
- **In welche Tabelle würde ein Attribut *TelNr* gehören?**
- **In welche Tabelle würde ein Attribut *Trainingsbeginn* gehören?**

Das ER-Diagramm (**E**ntity-**R**elation-Diagramm)

Die Bezeihungen zwischen den Entitätstypen stellt man grafisch dar. Dieses Klassen-Diagramm ist ein abstraktes Modell der Realität (sh. Grafik S.1). Es enthält die **Klassen** (=Entitätstypen) mit ihren Attributen und die **Beziehungen** zwischen den Klassen:



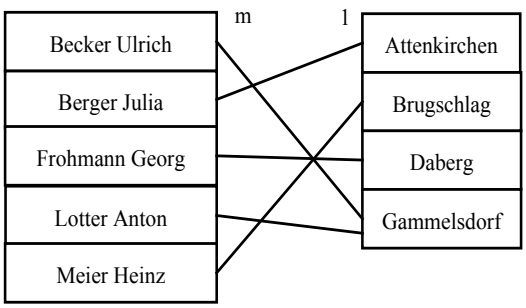
Das Diagramm zeigt auch die **Kardinalitäten** der Beziehungen an:

- Jedes Mitglied wohnt in einem (**1**) Ort.
- Jeder Ort hat mehrere (**m**) Mitglieder.
- Jedes Mitglied ist evtl. in mehreren (**m**) Abteilungen.
- Jede Abteilung hat mehrere (**n**) Mitglieder.

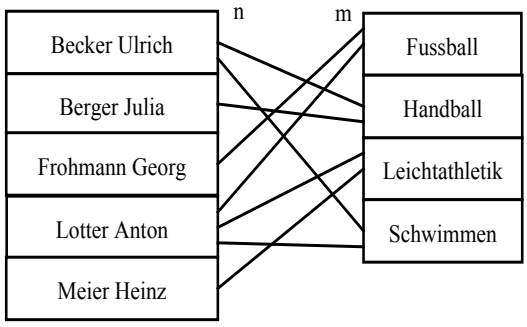
Ort-Mitglied ist eine **1:m-Beziehung**, *Mitglied-Abteilung* eine **n:m-Beziehung**.

Die Symbole eines ER-Diagramms sind seit einigen Jahren genormt: Rechteck für eine Klasse, Raute für eine Beziehung und Ellipse für ein Attribut.

Unterschied zwischen einer 1:m-Beziehung und einer n:m-Beziehung:



m:1-Beziehung: Jedes Mitglied wohnt in **genau einem** Ort, aber in jedem Ort können **m**ehrere Mitglieder wohnen.
(1:m kommt von „one to many“)



n:m-Beziehung: Jedes Mitglied kann mehrere Sportarten betreiben, jede Sportart wird von mehreren Mitgliedern betrieben.

Die 1:m-Beziehung lässt sich leicht mit einem Fremdschlüssel *OrtNr* in *tblMitglied* realisieren:

M-ID	Nachname	Vorname	...	OrtNr
1	Becker	Ulrich		1
2	Berger	Julia		4
3	Frohmann	Georg		3
4	...			

m 1

OrtID	Ortname	PLZ
1	Attenkirchen	85395
2	Brugschlag	...
3	Daberg	
4	Gammelsdorf	
5	...	

Jedes Mitglied wohnt **1** Ort.
Jeder Ort beherbergt **m** Mitglieder.

Die n:m-Beziehung muss durch eine Zuordnungstabelle aufgelöst werden:

Da Datenbanksysteme prinzipiell keine n:m-Beziehungen darstellen können, muss eine n:m-Beziehung mit Hilfe einer Zuordnungstabelle in zwei 1:m-Beziehungen aufgelöst werden.

M-ID	Nachname	Vorname	...
1	Becker	Ulrich	
2	Berger	Julia	
3	Frohmann	Georg	
4	...		

1 m

M-Nr	Abt-Nr
1	2
1	1
2	2
3	4

m 1

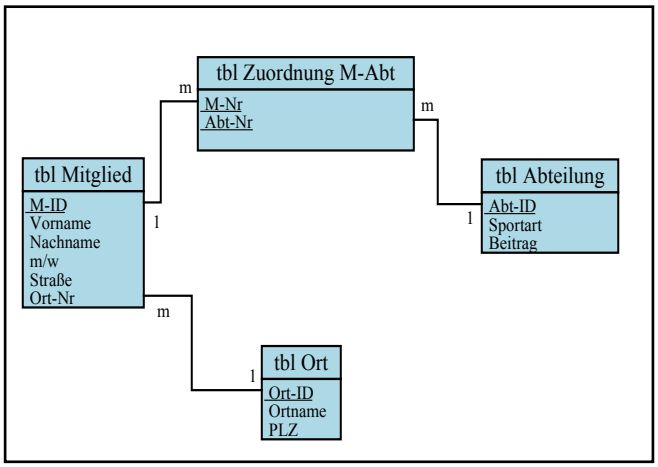
AbtID	Sportart	Beitrag
1	S	80
2	H	55
3	L	55
4	F	55
...		

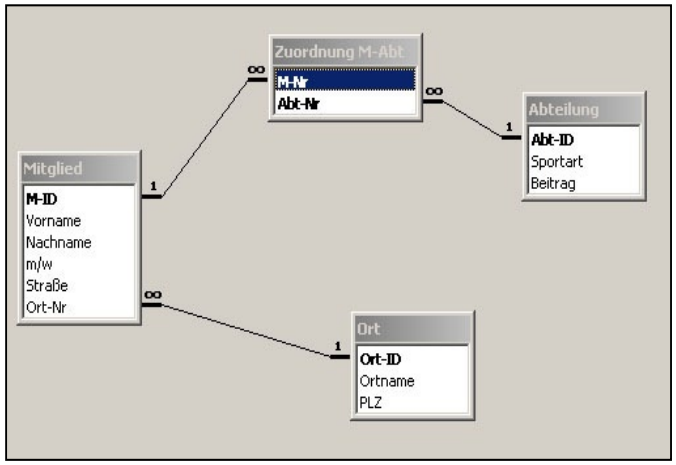
Die Zuordnungstabelle enthält als Datensätze die Zahlenpaare (M-Nr , Abt-Nr).

Jedes Mitglied hat **m** Datensätze in der Zuordnungstabelle.
Jeder Datensatz der Zuordnungstabelle gehört zu **1** Mitglied.

Jede Abteilung hat **m** Datensätze in der Zuordnungstabelle.
Jeder Datensatz in der Zuordnungstabelle gehort zu **1** Abteilung.

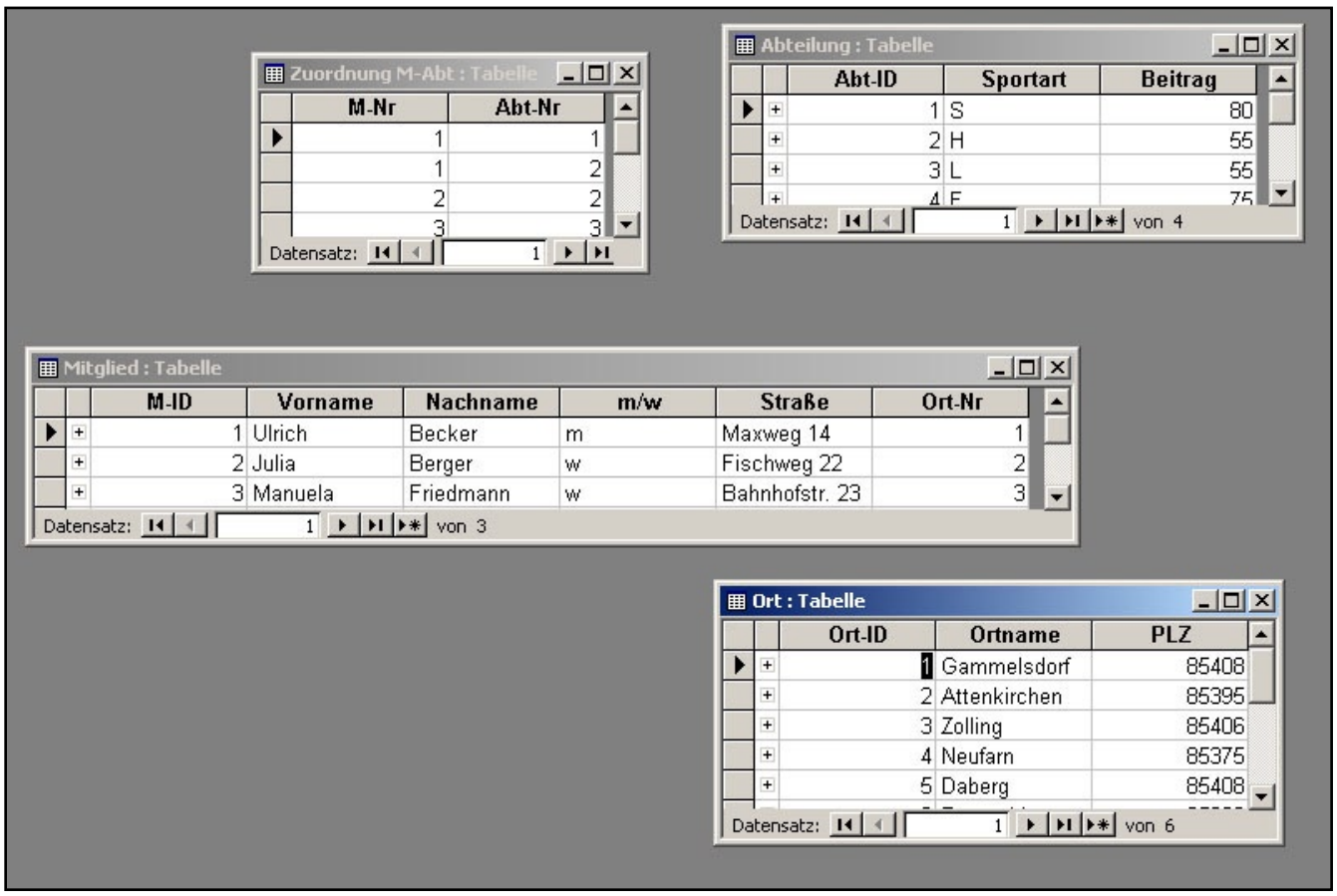
Das relationale-Datenbankschema ist das Klassendiagramm. Es ist für die direkte Umsetzung in einem Datenbank-Programm wie Access geeignet. Aus Sicht der Modellierung stellt es das reale Modell dar (sh. Grafik S.1) .





Dieses Diagramm wird so wie rechts im Access-Fenster *Beziehungen* angezeigt, wenn du vorher die vier Tabellen erstellt und sie korrekt verknüpft hast. Access benutzt anstelle des Buchstaben m das ∞-Zeichen. Unterstrichen bzw. mit Fettschrift wird der jeweilige Primärschlüssel einer Tabelle gekennzeichnet.

Die zugehörigen Tabellen:



Zuordnung M-Abt : Tabelle

M-Nr	Abt-Nr
1	1
1	2
2	2
3	3

Datensatz: 1

Abteilung : Tabelle

Abt-ID	Sportart	Beitrag
1	S	80
2	H	55
3	L	55
4	F	75

Datensatz: 1 von 4

Mitglied : Tabelle

M-ID	Vorname	Nachname	m/w	Straße	Ort-Nr
1	Ulrich	Becker	m	Maxweg 14	1
2	Julia	Berger	w	Fischweg 22	2
3	Manuela	Friedmann	w	Bahnhofstr. 23	3

Datensatz: 1 von 3

Ort : Tabelle

Ort-ID	Ortname	PLZ
1	Gammelsdorf	85408
2	Attenkirchen	85395
3	Zolling	85406
4	Neufarn	85375
5	Daberg	85408

Datensatz: 1 von 6

Regeln für die Erstellung der Datenbank:

- Jeder Entitätstyp *Mitglied, Ort, Abteilung* bekommt eine eigene Tabelle.
- Jede Tabelle braucht einen **Primärschlüssel**, das ist ein Attribut (oder eine Attribut-Kombination), das jeden Datensatz eindeutig identifiziert. Weil *Name* nicht unbedingt eindeutig ist, benutzt man meist ein künstliches Attribut (*M-ID, Ort-ID, ...*), das dann vom Typ *Autowert* sein muss (das ist eine Zahl, die Access selber verwaltet).
- Um die Tabellen *Mitglied* und *Ort* verknüpfen zu können, bekommt die Tabelle *Mitglied* ein Attribut *Ort-Nr* (**Fremdschlüssel**) vom Typ *Zahl*. Dann kann man im Beziehungen-Fenster *Ort.Ort-ID* mit *Mitglied.Ort-Nr* verknüpfen. Im Feld *Ort-Nr* der *Mitglied*-Tabelle stehen dann nicht die Ortsnamen, sondern nur die Nummern der Orte.
- Die n:m-Beziehung Mitglied-Abteilung wird durch eine eigene Tabelle ***Zuordnung M-Abt*** aufgelöst. Diese Tabelle enthält (zunächst) nur die Attribute *M-Nr* und *Abt-Nr*. Weil jedes Zahlenpaar *M-Nr/Abt-Nr* in der Tabelle nur einmal vorkommt, identifiziert es den Datensatz eindeutig, und man benutzt beide Attribute zusammen als **kombinierten Primärschlüssel**.
- Eine 1:1-Beziehung (z.B. Person-Tel.nr.) gehört normaler Weise in eine einzige Tabelle.

Ein wenig Geschichte:

Das **relationale Datenbankmodell** wurde 1970 von Dr. Edgar F. Codd (IBM) als rein mathematisches Modell entwickelt. Zehn Jahre später wurde die erste relationale Datenbank zum ersten Mal implementiert von der Firma **ORACLE**. Relationale Datenbanken haben fast alle anderen Datenbankmodelle verdrängt, weil sie den großen Vorteil haben, dass man mit ihnen eine Datenbankstruktur verändern kann, ohne dass man die Anwendungen verändern muss, die diese Datenbank benutzen. Die Firma Borland hat ihre moderne objektorientierte, datenbankfähige Programmiersprache **Delphi** genannt, weil man bekanntlich in Delphi das Orakel befragte. Allerdings brauchst du, wenn du mit Delphi Datenbank-Anwendungen programmieren willst, die Professional-Version.

Schrittweise erarbeiten:

- **Tabellen *Mitglied* und *Ort* erstellen mit Primärschlüssel und Feldnamen, Datentypen**
- **Fremdschlüssel, Beziehungen definieren (dazu auch Kap. 3.4 lesen!!)**
- **Tabellen *Abteilung* und *Zuordnung M-Abt*, kombinierter Schlüssel**
- **jetzt erst Daten eingeben in die Tabellen**
- **Den Schülern eine Sportverein.mdb mit vielen Datensätzen geben und damit**
- **Einfache Auswahl-Abfragen erstellen (Sortierung, Kriterien, Parameterabfrage)**
  Abfragen immer mit qry ... bezeichnen!
- **Untersuche die Wirkung, wenn in einer Abfrage die Beziehung zw. 2 Tabellen fehlt.**
- **Untersuche die Wirkung, wenn eine unnötige Tabelle in einer Abfrage ist.**
- **Abfrage: Alle Mitglieder mit Sportarten, verbesserte Darstellung in Kreuztabelle**
- **Abfrage: Sportarten mit Anzahl der Mitglieder**
- **Abfrage: Alle Mitglieder mit Alter (berechnetes Feld)**

Termin-Aufgabe:

Eine Musikschule will ihren Unterricht mit einer kleinen Datenbank verwalten. Sie hat mehrere Musiklehrer (Name, Vorname, Instrument), von denen jeder nur ein Instrument unterrichtet, und viele Schüler (Name, Vorname, Adresse mit PLZ). Die Schüler werden in Einzelunterricht jeweils von einem Lehrer in seinem Instrument unterrichtet. Ein Schüler kann auch mehr als ein Instrument lernen, dann hat er für jedes Instrument einen eigenen Lehrer. Zur Zeit hat die Musikschule Lehrer für Gitarre, Klavier, Schlagzeug, Trompete und Posaune.

1. Erstelle ein **ER-Diagramm** mit den Entitätstypen *Lehrer*, *Schüler* und *Ort* und ein **relationales Datenbankschema**. (Zeichne beide mit einem Vektorgrafik-Programm am PC !)
2. Erstelle die Datenbank „Musikschule.mdb“ mit Access und verknüpfe die Tabellen, wie es die Diagramme vorgeben.
3. Gib in die Tabellen die Datensätze für mindestens die obigen 5 Lehrer und 20 Schüler ein. Achte darauf, dass es auch Schüler gibt, die mehr als ein Instrument lernen.
4. Erstelle folgende Abfragen:

   a) Alle Lehrer mit Instrumenten, alphabetisch sortiert nach den Instrumenten.

   b) Alle Schüler alphabetisch sortiert, mit Instrument und Lehrer.

   c) Alle Schüler, die von einem bestimmten [Lehrer] unterrichtet werden (Parameter-Abfrage).

   d) Alle Schüler, die Gitarre lernen.

   e) Alle Schüler, die ein bestimmtes [Instrument] lernen (Parameter-Abfrage).

   f) Alle Schüler, die in Königsbrunn wohnen.

   g) Alle Schüler aus Königsbrunn, die Gitarre oder Klavier lernen.
5. Es soll zusätzlich Tag und Uhrzeit des jeweiligen Unterrichts gespeichert werden. Da diese Attribute weder allein zu Schüler noch allein zu Lehrer gehören, sollte man sie in der Zuordnungstabelle Schüler-Lehrer unterbringen. Erstelle dann Abfragen der Art:

   h) Alle Unterrichtszeiten der Woche für einen [Lehrer].

   i) Alle Schüler mit Instrument, Lehrer und Unterrichtszeit.

Viel Spaß! Hn

schwierige Aufgabe:

In der gleichen Musikschule soll ein Lehrer auch mehrere Instrumente unterrichten können. Außerdem soll der Monatsbetrag für den Unterricht gespeichert werden, z.B. kostet Klavierunterricht monatlich 80€ und Gitarreunterricht 70€. Eine Unterrichtsstunde wird jetzt nicht mehr allein durch die Angabe des Lehrers und Schülers bestimmt, sondern benötigt auch die Angabe des Instruments.

Die Struktur dieser Datenbank wird deutlich anders aussehen müssen als die bisherige. Fertige ein ER-Diagramm und realisiere diese Datenbank.

Serienbrief mit *MS-Word* auf der Grundlage einer *Access*-Abfrage

(Dies liegt etwas abseits vom eigentlichen Thema, wird aber in der Praxis häufig gebraucht.)

Aufgabenstellung:

An alle Mitglieder des Sportverein, die nächstes Jahr volljährig werden (die also im Jahr ... geboren sind), soll ein Brief verfasst werden.

Lösung:

1. Erstelle zuerst in der Datenbank *Sportverein.mdb* eine Abfrage *qry18*, die erstens alle Mitglieder mit dem Geburtsjahr ... liefert und zweitens alle Datenfelder enthält, die für den Brief von Bedeutung sind: *Name, Vorname, Geschlecht, Straße, PLZ, Ortname*.

2. Erstelle in *MS-Word* einen Serienbrief, der die erforderlichen Daten aus der Abfrage *qry18* von *Sportverein.mdb* bezieht. *Word* hat zur Erstellung eines Serienbriefs einen Assistenten, der über *Extras* aufgerufen wird. Je nach Version von *MS-Office* sind die Einzelschritte etwas unterschiedlich. Generell musst du die folgenden Schritte durchlaufen:

   1) ein **Hauptdokument erstellen:** Dieses soll nachher den Brieftext und Platzhalter für <<Name>>, <<Vorname>>, <<Herr/Frau>> ... enthalten.

   2) eine **Datenquelle importieren:** Wähle *Datenquelle öffnen*, suche deine Datenbank *Sportverein.mdb* und wähle darin *qry18*.

   3) das **Hauptdokument bearbeiten:** Dabei schreibst du den Brief und fügst an den entsprechenden Stellen **Bedingungsfelder** ein (z.B. Wenn *Geschlecht* gleich *w* dann *Frau* sonst *Herr*) und **Seriendruckfelder** (Word bietet dir alle Feldnamen von *qry18* zur Auswahl an.)

   4) die **Daten mit dem Dokument zusammenführen:** Wähle *Ausführen/Seriendruck in neues Dokument/Zusammenführen* (in OfficeXP heißt es ein bißchen anders). Dabei wird jeder Datensatz der Abfrage *qry18* mit dem Hauptdokument zusammengeführt. Es wird ein neues Dokument erstellt, das der Reihe nach alle gewünschten Briefe enthält und gedruckt werden kann.

Zwei fertige Seiten des Serienbriefs:

SV Hintertupfing

An Herrn
Michael Fiedmann

Bahnhofplatz 23
85406 Zolling

Lieber Michael,

nachdem du im komm
darüber informieren, .

SV Hintertupfing

An Frau
Maria Brucker

Winkelstraße 233
85399 Halbergmoos

Liebe Maria,

Bedingungsfeld

Seriendruckfeld

nachdem du im kommenden Jahr volljährig wirst, wollen wir dich rechtzeitig darüber informieren, ...

schwierigere Aufgabe:

Erstelle einen Serienbrief an alle weiblichen Mitglieder des Sportvereins und unterscheide in der Anrede, ob das Mitglied volljährig ist oder nicht, in der Art:

volljährig: „Sehr geehrte Frau Fiedmann,“

nicht vollj.: „Liebe Inge,“

Das Problem: Im Text des Wenn-Dann-Bedingungsfelds müssen die Serienfelder <<Nachname>> bzw. <<Vorname>> untergebracht werden. Dies gelingt nur im Quelltext des Bedingungsfelds. Du erreichst diesen mit der rechten Maustaste / *Feldfunktionen ein*.