1 Einführung

1.1 Beispiel einer Datenbank, Teil I

Die Verwaltung der Mitglieder eines Sportvereins soll mit Hilfe einer Datenbank gemanagt werden.
Tabellenstruktur mit Feldnamen:

M-Nr	Vorname	Nachname	m/w	Straße	PLZ	Ort	Sportart	Beitrag
1	Ulrich	Becker	m	Maxweg 14	85408	Gammelsdorf	S	80,00 €
2	Ulrich	Becker	m	Maxweg 41	85408	Gammelsdorf	H	55,00 €
3	Julia	Berger	w	Fischweg 22	85395	Attenkirchen	H	55,00 €
4	Manuela	Fiedmann	w	Bahnhofstr. 23	85406	Zolling	L	55,00 €
5	Otto	Fischer	m	Karlweg 12	85375	Neufarn	F	75,00 €
6	Georg	Frohmann	m	Meierweg 99	85408	Daberg	F	75,00 €
7	Hans	Huber	m	Postweg 12	85368	Brugschlag	S	80,00 €
8	Hans	Huber	m	Postweg 12	85368	Brugschlag	F	75,00 €

Dabei treten bei der Verwaltung der **Vereinsmitglieder in einer Tabelle Probleme** auf:

a) Wenn ein Mitglied in mehreren Abteilungen aktiv ist, muss der gesamte Datensatz des Mitgliedes (vollständiger Name und Adresse) erneut eingegeben werden. Dadurch ergibt sich bei der automatischen Erstellung der M-Nr das Problem, dass bei Mitgliedschaften in mehreren Abteilungen ein Mitglied verschiedene M-Nrn erhält → **Inkonsistenz.** Außerdem ist dies eine **Fehlerquelle** sowohl bei der erneuten Eingabe als auch bei Änderungen (vgl. M-Nr 1, *Straße* mit M-Nr 2, *Straße*).

Beim Ändern von Personendaten muss überprüft werden, in wie vielen Abteilungen das betreffende Mitglied aktiv ist, so dass die Änderungen durchgeführt werden können, da andernfalls unkorrekte Datensätze existieren.
Entsprechend bekommt man Probleme, wenn der Beitrag einer Abteilung geändert werden soll. So muss dies bei allen Mitgliedern dieser Abteilung durchgeführt werden. Wird einer übersehen, ist die Datenbank nicht mehr konsistent (**Änderungsanomalie**).

b) Beim Löschen von Mitgliedern tritt das gleiche Problem auf wie in a).
Unter Umständen wird eine ganze Abteilung gelöscht, falls alle momentan in der Abteilung aktiven Mitglieder gelöscht werden (vgl. M-Nr 4), das Gleiche gilt auch für einen Ort (**Löschanomalie**).

c) Wenn eine neue Abteilung aufgemacht werden soll, kann dies nur in Verbindung mit zugehörigen Mitgliedern geschehen, da andernfalls der Datensatz nicht in diese eine Tabelle eingegeben werden kann, entsprechendes gilt für neue Orte **(Einfügeanomalie).**

Aus diesen Gründen muss man sich eine andere Aufteilung der Daten überlegen, so dass die oben genannten Fehlerquellen bzw. Anomalien nicht auftreten können. Dazu erfasst man zuerst die Informationsstruktur des zu erstellenden Vereinsmodells:

1. Folgende Daten sollen von den einzelnen Mitgliedern verwaltet werden:
   Mitglieds-Nr, Vorname, Nachname, Geburtsdatum, Geschlecht, Straße, PLZ, Wohnort, betriebene Sportart sowie der zu zahlende Abteilungsbeitrag.
2. Ein Mitglied kann in verschiedenen Sportabteilungen aktiv sein; in einer Sportabteilung können aber auch mehrere Mitglieder sein.
3. Von jedem Mitglied wird nur eine Adresse (Hauptwohnsitz) erfasst; aber in einem Ort können mehrere Mitglieder wohnen.

Die weitere effektive Verarbeitung dieser Informationen im obigen Sinne erfordert aber theoretische Grundkenntnisse, die in den folgenden Kapiteln bereitgestellt werden.

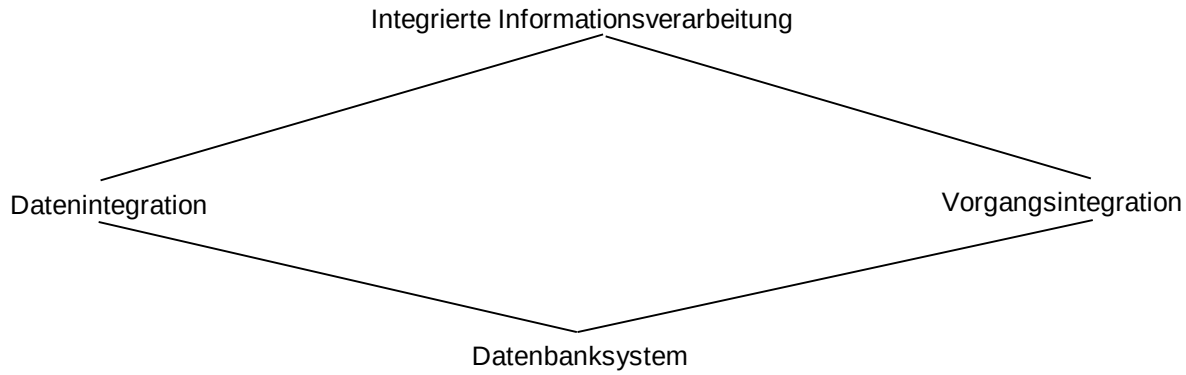
1.2 Integrierte Informationsverarbeitung

Informationen zu sammeln, abzulegen, wiederzufinden und nach verschiedenen Kriterien auszuwerten ist eine wesentliche Tätigkeit im Beruf aber auch im Privatbereich. Recherchetechniken wie das gezielte Durchforsten von Menübäumen bzw. Hyperlinks sowie die effektive Formulierung von Abfragen bekommen in bestimmten Bereichen die Bedeutung von Kulturtechniken wie Lesen, Schreiben und Rechnen.

Um die immer weiter anwachsende Informationsflut in den Griff zu bekommen, werden in zunehmendem Maße Datenbanken zur Verwaltung der Informationen eingesetzt. Kenntnisse darüber, mit welchen Verfahren solche Informationen verknüpft und ausgewertet werden können, tragen auf zwei Arten zur Mündigkeit des Bürgers bei: Als Anwender kann er die ihm zur Verfügung stehenden Daten effizienter für sich nutzen; er kann aber auch besser abschätzen, welche Aussagen aus der Zusammenführung von Daten über ihn gewonnen werden können.

Durch unkontrolliert wachsende Datenbestände ist jedoch in bestimmten Bereichen ein Datenchaos entstanden, das weiter fortschreitet und zum Jahrhundertproblem der Informatik werden kann. Verursacht wird das Datenchaos u. a. durch die eigenständige isolierte Datenhaltung für einzelne Anwendungen. Beispielsweise kann es in einem Betrieb vorkommen, dass Kundendaten für unterschiedliche Anwendungen jeweils neu gespeichert werden, z. B. für Abrechnungen, für die Zusendung von Informationsmaterial, für Service und Beratung. Mit der Mehrfachspeicherung gleicher Informationen (Redundanz) wird nicht nur Speicherplatz verschwendet, sondern sie führt auch dazu, dass die Informationsverarbeitung nicht effizient genug ist (mehrfache Datenpflege ist notwendig). Die schwerwiegendsten Fehler können aber dadurch entstehen, dass bei Mehrfachspeicherung der Daten unterschiedliche Änderungen in den einzelnen Anwendungen vorgenommen werden; dadurch sind die Daten nicht mehr widerspruchsfrei (Verletzung der Datenintegrität). Dieser Zustand führt dann zum Chaos in der Datenorganisation. Was sich anhört wie eine Erzählung aus längst vergangenen Zeiten der Informationstechnik, ist in vielen Organisationen bittere Realität.

Zur Lösung dieser Probleme soll die **integrierte Informationsverarbeitung** beitragen. Sie bedeutet **Datenintegration** und **Vorgangsintegration**.



Datenintegration wird durch eine Datenbasis (Datenbank) erreicht, die von mehreren Anwendungen in unterschiedlichen betrieblichen Funktionsbereichen gemeinsam genutzt wird. Ziel ist die zusammenfassende Abbildung der gesamten Organisation und ihrer Beziehungen zur Umwelt in einer einzigen Datenbasis (globales Modell).

Vorgangs-, Funktions- oder Prozessintegration erfolgen durch die EDV-technische Verknüpfung von arbeitsteilig in verschiedenen Abteilungen abzuwickelnden Vorgängen zu Ablaufketten (z. B. Auftragsabwicklung vom Kundenauftrag bis zur Auslieferung und Entsorgung).

1.3 Ziele der Datenorganisation

Unter dem Begriff **Datenorganisation** werden alle Verfahren zusammengefasst, die dazu dienen, Daten zu **strukturieren** und auf Datenträgern zu **speichern** (schreibender Zugriff) und für den lesenden Zugriff verfügbar zu halten. **Ziele** der Datenorganisation sind:

1) Datenunabhängigkeit

- Unabhängigkeit vom Anwendungsprogramm: Die Daten sind anwendungsneutral gespeichert, d. h. unabhängig vom erzeugenden oder benutzenden Anwendungsprogramm (im Gegensatz zur integrierten Verarbeitung mit Dateiorganisation).
- Unabhängigkeit der logischen von der physischen Datenorganisation: Der Benutzer muss nur die Datenstrukturen kennen. Methoden zum Suchen, Ändern, Einfügen und Löschen von Datensätzen werden vom Datenbankverwaltungssystem zur Verfügung gestellt.
- Physische Datenunabhängigkeit: Das Datenbankverwaltungssystem steuert und überwacht (im Zusammenspiel mit dem Betriebssystem) die peripheren Geräte, blockt bzw. entblockt Sätze, kontrolliert Überlaufbereiche, belegt Speicherräume oder gibt sie frei usw.

2) Benutzerfreundlichkeit

Leicht zu erlernende Benutzersprachen ermöglichen sowohl dem professionellen Benutzer (Systementwickler, Programmierer) als auch dem Endbenutzer eine einfache Handhabung der Daten. Die Benutzersprachen sollten durch grafische Bedienoberflächen unterstützt werden.

3) Mehrfachzugriff

Jeder, der autorisiert ist, darf im Mehrbenutzerbetrieb auf die gespeicherten Daten zugreifen.

4) Flexibilität

Die Daten müssen in beliebiger Form verknüpfbar sein (mehrdimensionaler Zugriff, Vielfachzugriff). Sie müssen sowohl den fortlaufenden als auch den wahlfreien Zugriff ermöglichen.

5) Effizienz

Die Zeiten für die Abfrage und für die Verarbeitung müssen kurz sein, ebenso für Änderungen und Ergänzungen des Datenbestandes.

6) Datenschutz

Die Daten sind vor unbefugtem Zugriff (Missbrauch) zu schützen. Typische Fragen sind

- Ist der Teilnehmer überhaupt zugriffsberechtigt?
- Ist der Teilnehmer nur zu bestimmten Daten zugriffsberechtigt?
- Ist der Teilnehmer nur zu Abfragen oder auch zu Änderungen berechtigt?

7) Datensicherheit

Die Daten müssen gegen Programmfehler und Hardware-Ausfälle gesichert sein. Das Datenbanksystem soll nach Störungsfällen den korrekten Zustand wiederherstellen (recovery). Die Speicherung langlebiger Daten wird auch als Datenpersistenz bezeichnet.

8) Datenintegrität

Die Daten müssen vollständig, korrekt und widerspruchsfrei sein. Beispielsweise muss jeder Wert eines Fremdschlüssels in einem verknüpften Primärschlüssel auch als Wert im entsprechenden Primärschlüssel vorkommen (**referentielle Integrität**). Daten, die redundant gespeichert sind, müssen dasselbe aussagen (Datenkonsistenz). Die Forderung nach Datensicherheit wird gelegentlich in die Datenintegrität einbezogen.

9) Redundanzfreiheit

Jedes Datenelement sollte möglichst nur einmal gespeichert werden, z. B. die Kundenanschrift nicht wie in der Dateiorganisation gleichzeitig bei der Auftragsbearbeitung, der Fakturierung und der Debitorenbuchhaltung.

Die genannten Anforderungen sind idealtypisch und stehen miteinander in Konkurrenz. Mehr Redundanz wird z. B. mit geringerer Flexibilität und Effizienz erkauft.

1.4 Dateisysteme und Datenbanksysteme

Bei den weiteren Überlegungen sollen ausschließlich solche Datenbanksysteme betrachtet werden, mit denen die in Kapiteln 1.2 und 1.3 aufgestellten Forderungen weitestgehend erfüllt werden können.
Systeme, bei denen die Daten in verschiedenen Dateien gespeichert werden, die nicht miteinander verknüpft werden können, sollen als Dateisysteme bezeichnet werden. Typisch für die konventionelle Datenverarbeitung auf der Basis von Dateisystemen ist, dass die Dateien in der Regel für eine Anwendung oder für eng zusammenhängende Anwendungen entworfen werden. Jeder Programmierer baut sich seine Dateien selbst auf, unabhängig und vielleicht sogar ohne Kenntnis der Dateien anderer Programmierer. Der Dateiaufbau ist unmittelbar an die jeweilige Verarbeitung angepasst, und in dieser Form ist die Datei auch abgespeichert. Die grundsätzliche Situation ist in Bild 1-1 zusammengefasst.

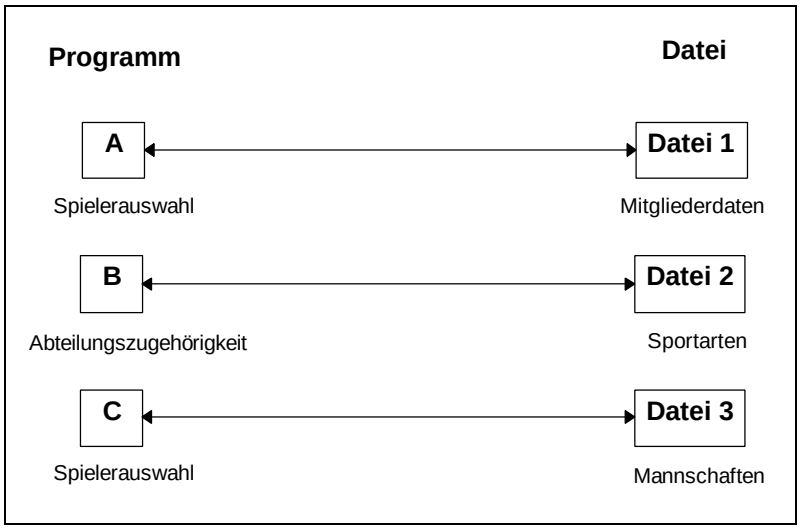


Bild 1-1: Enge Kopplung zwischen Programm und Datei bei der Dateiverwaltung

Diese Vorgehensweise kann bei der Verwaltung von Daten zu schwerwiegenden Problemen führen:

(1) Redundanz
Da die Daten jeweils speziell für bestimmte Anwendungen entworfen werden, werden dieselben Daten in verschiedenen Dateien wieder auftauchen (z. B. Namen und Adressen von Lehrern in der Datei für die Lehrer und in der Datei für die Klassen). Redundanz führt zu Speicherverschwendung und zu erhöhten Verarbeitungskosten, vor allem bei Änderungen. Schlimmer jedoch ist es, dass diese Redundanz in der Regel nicht zentral kontrolliert wird, so dass Konsistenzprobleme auftreten.

(2) Inkonsistenz
Die Konsistenz der Daten (d. h. die logische Übereinstimmung der Datei-Inhalte) kann nur schwer gewährleistet werden. Bei der Änderung einer Größe müssten alle Dateien geändert werden, die diese Größe beinhalten und diese Änderungen müssten so miteinander abgestimmt geschehen, dass nicht verschiedene Programme zum selben Zeitpunkt unterschiedliche Werte derselben Größe sehen können.

(3) Daten-Programm-Abhängigkeit
Ändert sich der Aufbau einer Datei oder ihrer Organisationsform, so müssen darauf basierende Programme geändert werden. Wird beispielsweise für eine Anwendung ein weiteres Datenelement in einem Satztyp benötigt (z. B. zweite Telefonnr. eines Schülers), so müssen infolge der notwendigen Neudefinitionen der Datei alle Programme geändert werden, ob sie dieses neue Datenelement sehen wollen oder nicht.

(4) Inflexibilität
Da die Daten nicht in ihrer Gesamtheit sondern nur anwendungsbezogen gesehen werden, ist es in vielen Fällen sehr kompliziert, neue Anwendungen oder Auswertungen vorhandener Daten zu realisieren. Dies gilt insbesondere für Auswertungen, die Daten aus verschiedenen Dateien benötigen. Die Organisation nach diesem konventionellen Vorgehen ist sehr wenig anpassungsfähig an die sich verändernden Anforderungen in einem Unternehmen bzw. in einer Schule.

2 Datenbankarchitektur

2.1 Das Konzept des Datenbanksystems

Unter einem **Datenbanksystem** stellt man sich ein System vor, das es erlaubt, große Datenmengen abzuspeichern, nach beliebigen Kriterien Daten wiederzufinden und Daten zu verändern. Beispielsweise möchte man, wenn man die Daten über alle Schüler einer Schule abgespeichert hat, die Anfrage stellen können:

Liste alle Schüler auf, die nicht in Augsburg wohnen und 18 Jahre alt sind.

Datenbanksysteme werden im folgenden als ein Organisationsmittel betrachtet, das folgende Aufgaben in einer Organisation abdeckt:

- Die Daten der Organisation sind für alle Benutzer in einer gemeinsamen Datenbasis (der Datenbank) abgespeichert.
- Viele Benutzer mit unterschiedlichen Anforderungen arbeiten mit diesen Daten. Das Datenbanksystem übernimmt den Zugriff und die Darstellung der gewünschten Daten.
- Das Datenbanksystem kontrolliert den Zugang zu den Daten, es zeigt Daten nur berechtigten Benutzern.

Beispiel 2-1:

Verschiedene Benutzersichten für Schulleiter, Lehrer und Programmierer auf die Datenbank:

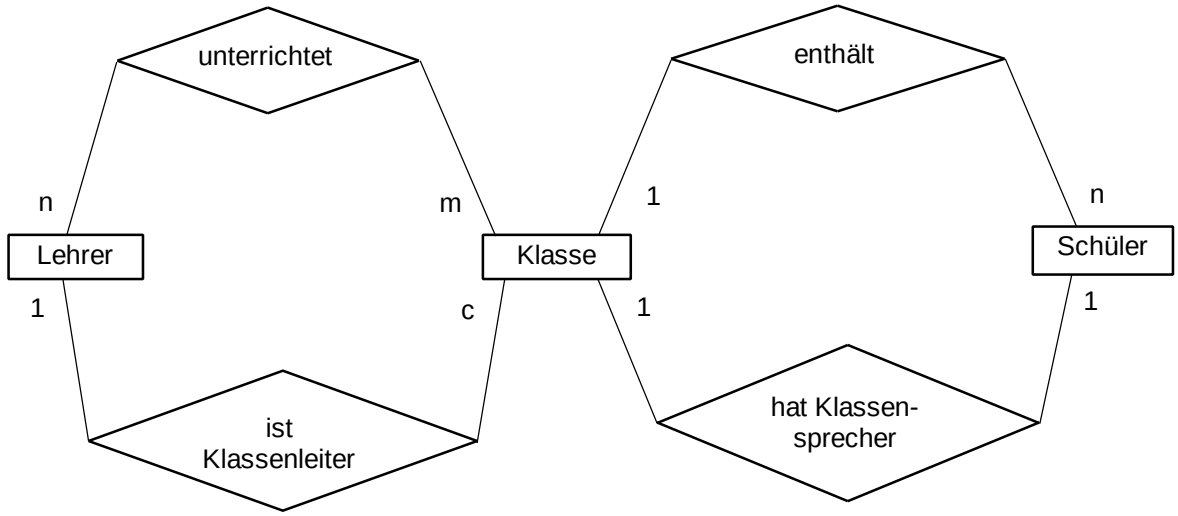
Herr Spieking, ein Englischlehrer, benötigt folgende Sicht auf die Daten:

S-Nr	Schüler-Name	Vorname	Klasse	Fach	Fachlehrer	Note
427	Abel	Franz	12TA	E	Spieking	3
428	Bond	Regine	12TA	E	Spieking	1
429	...	...	...	...	...	...
430	...	...	...	...	...	...

Der Schulleiter interessiert sich momentan nur für alle Lehrer mit der Fächerkombination M/Ph, er möchte deshalb seine Daten folgendermaßen sehen:

L-Nr	Lehrer-Name	Lehrer-Vorname	Amtsbez	Fächerkombination
2	Kalkulus	Carl-Friedrich	StD	M/Ph
6	Nenner	Marie	StRin	M/Ph/Inf
...	...	...	...	...

Ein Programmierer möchte die Daten vielleicht so sehen:



Wie die Daten tatsächlich in der Datenbank gespeichert sind, sieht keiner der drei Benutzer - es ist für sie auch völlig unerheblich.

In Bild 2-1 ist das Konzept eines Datenbanksystems in seinen wesentlichen Grundzügen zusammengefasst.

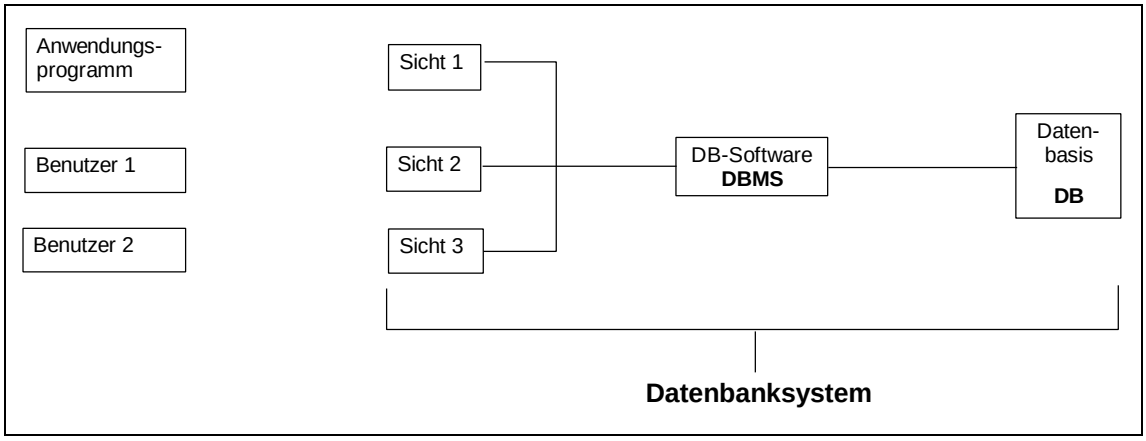


Bild 2-1: **Konzept des Datenbanksystems**

Der Begriff **Datenbank** wird als Bezeichnung für die Datenbasis eines Datenbanksystems verwendet. In der Literatur wird der Begriff auch als Kurzwort für das gesamte Datenbanksystem benutzt.

Eine Datenbank kann folgendermaßen beschrieben werden:

Eine Datenbank ist eine integrierte Ansammlung von Daten, die Benutzern verschiedener Anwendungen als gemeinsame Basis für die Pflege und Gewinnung von Informationen dient. Die Daten sind so strukturiert, dass jede - auch zunächst noch ungeplante - Anwendung in der benötigten Weise auf die Daten zugreifen kann. Die Abspeicherung der Daten geschieht mit Hilfe von Tabellen. Dabei entspricht eine Zeile der Tabelle einem Datensatz und eine Spalte der Tabelle einem Datenfeld. Die Datensätze werden von verschiedenen Benutzern nach unterschiedlichen Kriterien sortiert. Infolgedessen spielt die Reihenfolge der Eingabe der Datensätze keine Rolle.

Eine **Datenbank** bietet die **Möglichkeit** des **gleichzeitigen Zugriffs auf mehrere Dateien bzw. Tabellen** (Vielfachzugriff), um **Daten aus den unterschiedlichen Dateien (Tabellen) miteinander zu verknüpfen** (Flexibilität), und auf diese Weise zusätzliche Informationen aus den bereits bestehenden Dateien (Tabellen) zu gewinnen. Hinzu kommt die Möglichkeit der komfortablen Beantwortung von Anfragen des Benutzers mit Hilfe einer integrierten Abfragesprache.

2.2 Aufbau von Datenbanksystemen

Die Hauptkomponenten eines Datenbanksystems sind:

1. die **Datenbasis** (Datenbank)
   (z. B. Tabellen wie *Schüler, Lehrer, Kurse*)
2. das **Datenbankmanagementsystem**, auch Datenbankverwaltungssystem (DBMS)
   (z. B. *dBase, Oracle, Paradox, MS-Access*)

Ein Datenbanksystem ermöglicht es dem Benutzer, über ein Datenbankmanagementsystem (DBMS = Data Base Management System)

- die Struktur einer Datenbasis aufzubauen (Datendefinition),
- Daten zu pflegen: Datensätze eingeben, ändern und löschen (Datenmanipulation),
- Informationen aus der Datenbasis zu gewinnen (Datenabfrage),
- Zugangs- und Zugriffsrechte zu verwalten (Datenkontrolle),
- Daten zu sichern, zu exportieren und zu importieren (Datenübertragung).

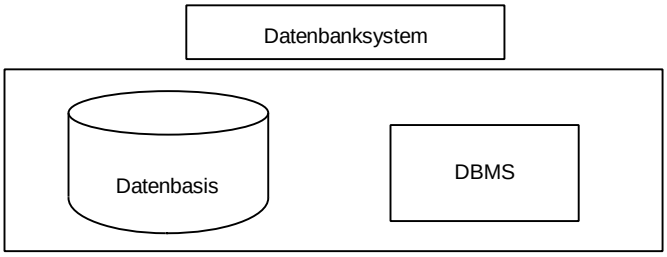


Bild 2-2: Komponenten eines Datenbanksystems

2.2.1 Das ANSI-Architekturmodell (3-Schichten-Architektur)

Urheber des ANSI-Architekturmodells ist das **A**merican **N**ational **S**tandards **I**nstitute, d. h. der nationale Normenausschuss der USA, der dem DIN in der Bundesrepublik Deutschland entspricht. Mit dem im Jahre 1975 herausgegebenen Architekturmodell unterbreitete ANSI einen Vorschlag für die prinzipielle Architektur von Datenbanksystemen, an dem sich sowohl die Hersteller von Datenbanksoftware als auch die Betreiber von Datenbanken orientieren sollten.

Der Kern dieser Architektur ist die konzeptionelle Ebene (konzeptionelles Schema). Hier wird der Teil der Realität, den das Datenbanksystem (DBS) nachbilden soll, in seiner logischen Gesamtheit beschrieben. In dieser Ebene werden alle Daten und ihre Beziehungen zueinander modelliert. Darüber hinaus ist sie unabhängig von den hardwaremäßigen Gegebenheiten und den Anforderungen einzelner Benutzer.

Über bzw. unter der konzeptionellen Ebene liegen die interne und die externe Ebene. In der **internen Ebene** wird die Organisation der Daten und ihrer Zugriffspfade auf den physischen Speicher festgelegt. Die externe Ebene legt fest, welche Daten bestimmte Benutzer bzw. Programme sehen und bearbeiten können.

Außerdem legt das ANSI-Architekturmodell noch die ebenenweise Zuordnung von personellen Instanzen fest, die als Anwendungs-, Unternehmens- und Datenbankadministrator bezeichnet werden und für die externe, konzeptionelle und interne Ebene zuständig sind.

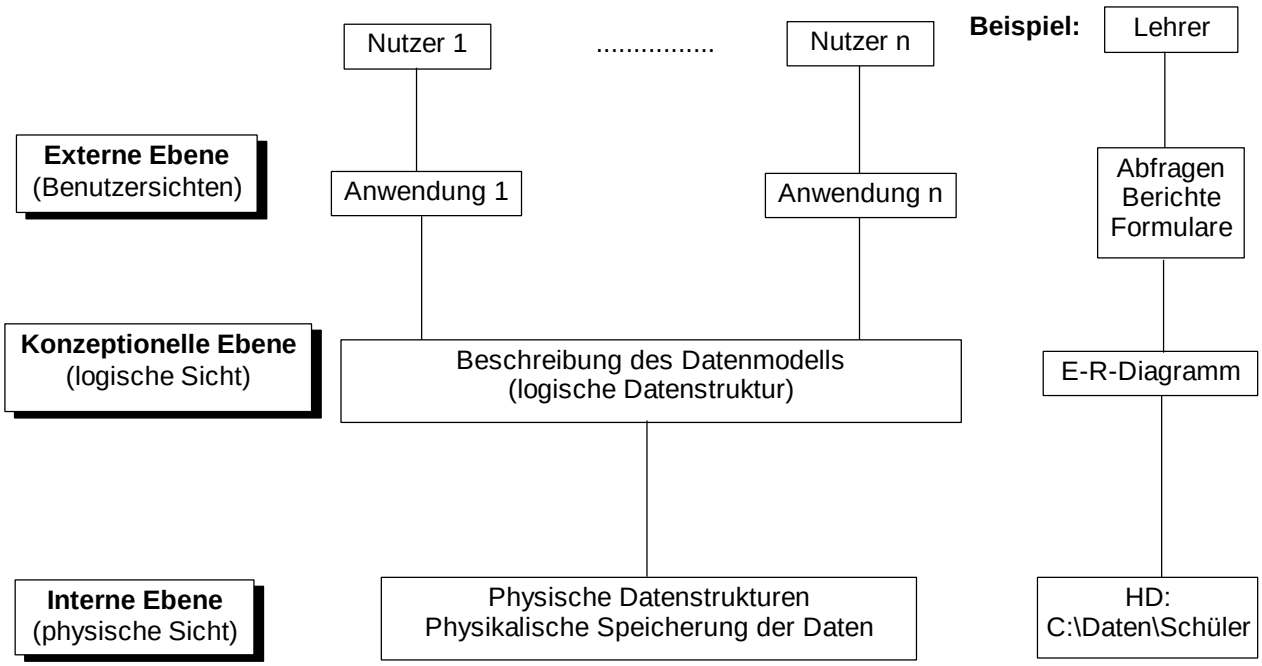


Bild 2-3: **Datenbankebenen im ANSI-Architekturmodell**

Hinweis: Das ANSI-Architekturmodell zeigt nicht die Schritte der Datenbankentwicklung, sondern die verschiedenen Teile eines Datenbankkonzepts.

2.2.2 Konzeptionelles Schema

Ein konzeptionelles Schema benennt und beschreibt alle logischen Dateneinheiten sowie die Beziehungen zwischen den Dateneinheiten für den einer Datenbank zugrunde liegenden Realitätsausschnitt. Die Form der Beschreibung und die Beschreibungsmöglichkeiten werden durch das Datenmodell festgelegt, das zur Erstellung des konzeptionellen Schemas herangezogen wird. So kann ein hierarchisches, ein netzwerkartiges oder ein relationales Datenmodell in Frage kommen. Keinesfalls enthält ein konzeptionelles Schema Angaben zur physischen Organisation von Datenstrukturen.

Als ein anwendungsübergreifendes Instrument zur Strukturierung und Beschreibung der Datenwelt eines Unternehmens zeichnet sich das konzeptionelle Schema durch folgende Vorteile aus:

- Es bildet eine relativ stabile informationelle Datenbankbeschreibungsbasis für alle aktuellen und künftigen Anwendungen eines Unternehmens.
- Es dokumentiert die Informationszusammenhänge eines Unternehmens in einer einheitlichen Form.
- Es ändert sich im Vergleich zu den einzelnen Anwendungen nur langsam.

Es sei darauf hingewiesen, dass das konzeptionelle Schema eines Unternehmens keine Dateninhalte enthält. Es beschreibt lediglich die Datenwelt eines Unternehmens in anwendungsübergreifender Form.

2.2.3 Internes Schema

Als internes Schema bezeichnet man die Beschreibung der physikalischen Organisation der in einem konzeptionellen Schema definierten logischen Datenstrukturen. Ein internes Schema legt also die physikalische Realisierung eines konzeptionellen Schemas auf Speichermedien fest. Es enthält daher Angaben zur Länge und zum Typ von Dateneinheiten, zur Speicherungsform von zusammengehörigen Dateneinheiten, zu Zugriffspfaden usw.

2.2.4 Externes Schema

Ein externes Schema beschreibt einen Ausschnitt aus dem konzeptionellen Schema eines Unternehmens, der auf die spezielle Datensicht einer bestimmten Benutzergruppe zugeschnitten ist. Da ein externes Schema nur einen Teil der konzeptionellen Gesamtsicht wiedergibt, bezeichnet man es auch als Subschema. Das auf die Bedürfnisse einer Benutzergruppe abgestimmte externe Schema soll die Dateneinheiten und Beziehungen nicht enthalten, die diese Benutzer nicht sehen wollen oder nicht sehen sollen. Ein externes Schema verbirgt also die logische Gesamtsicht vor der betroffenen Benutzergruppe; es gibt nur den Teil der logischen Gesamtsicht preis, der für die Anwendungen der Benutzergruppe von Interesse ist.

In einem Unternehmen werden in der Regel mehrere Benutzergruppen auftreten. Es sind daher mehrere, unterschiedliche Subschemata zu entwickeln - je eines pro Benutzergruppe (vgl. auch Beispiel 2-1 sowie die Bilder 2-1 und 2-3).

2.3 Arbeitsweise

Datenbankverwaltungssysteme (DBMS) stellen die Verbindung zwischen der Datenbasis und den Datenbankbenutzern bzw. Anwendungsprogrammen her. Dabei erfolgt der Zugriff der Anwendungen auf die Datenbasis nicht direkt, sondern nur über das DBMS (vgl. Bild 2-1).

Das folgende Diagramm veranschaulicht, wie das Datenbankverwaltungssystem eine Anfrage (Query) abarbeitet.

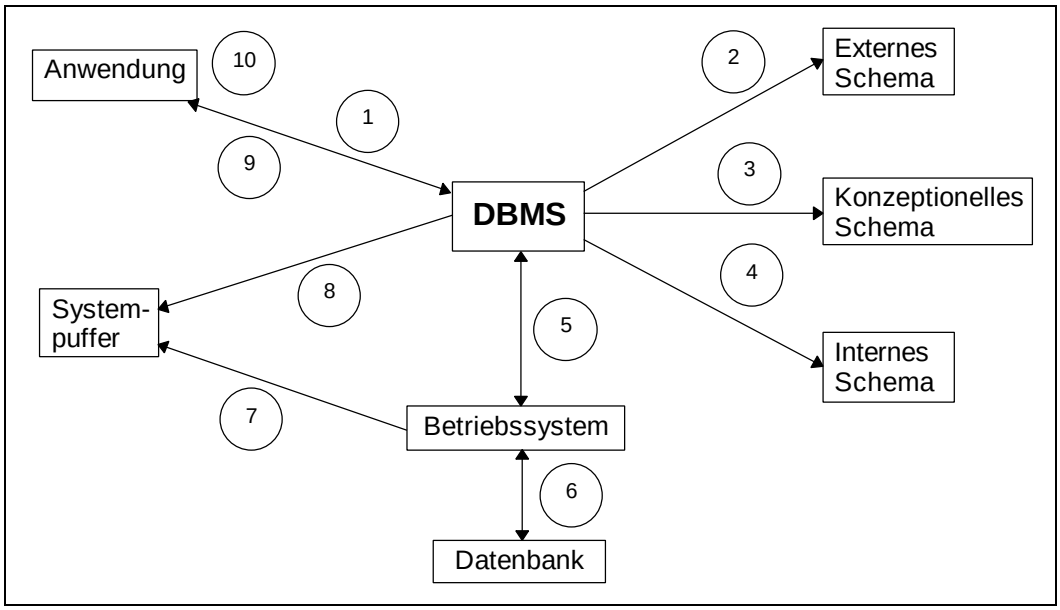


Bild 2-4: Abarbeitung einer Anfrage

Beispiel: Der Klassenleiter benötigt für die Einladung zum Elternabend die Adressetiketten der Eltern aller minderjährigen Schüler.

1. Das DBMS empfängt den Befehl des Anwendungsprogramms, ein bestimmtes Objekt zu lesen. *(Abfrage: Klasse - minderjährige Schüler - Eltern - Etiketten)*
2. Das DBMS holt sich die benötigten Definitionen des entsprechenden Objekttyps aus dem zugehörigen externen Schema. *(Abfrage öffnen)*
3. Das DBMS stellt fest, auf welche konzeptionellen Objekte sich die Anfrage bezieht. *(Klasse - Schüler)*
4. Das DBMS stellt fest, welche physischen Objekte zu lesen sind.
5. Das DBMS übergibt dem Betriebssystem die Nummern der zu lesenden Speicherblöcke.
6. Das Betriebssystem greift auf die physischen Speicherblöcke in der Datenbank zu. *(C:\DATEN\SCHÜLER)*
7. Das Betriebssystem übergibt die verlangten Blöcke an das DBMS in einen Systempuffer.
8. Das DBMS stellt aus den vorhandenen physischen Sätzen im Systempuffer das verlangte Objekt zusammen.
9. Das DBMS übergibt das Objekt dem Anwendungsprogramm in seinen Arbeitsspeicher.
10. Das Anwendungsprogramm verarbeitet die vom DBMS übergebenen Daten. *(→ Ausdruck)*

3 Datenmodellierung

Für die Verwaltung der Klassen in einer Schule soll die Struktur der Datenbasis entworfen und mit einem relationalen Datenbankmanagementsystem verwaltet werden. Das **Datenmodell**, das die für ein Informationssystem notwendigen Daten und Datenbeziehungen beschreibt, wird in **vier Phasen** entwickelt.

Externe Phase — **Informationsstruktur**
- Ermittlung des Informationsbedarfs der Benutzer
- Strukturierung dieser Informationen

Konzeptionelle Phase — **semantisches Modell**
- formale und strukturierte Beschreibung aller relevanten Objekte und deren Beziehungen untereinander.

Logische Phase — **logisches / relationales Modell**
- Umsetzung des semantischen Datenmodells in ein relationales Datenbankmodell.

Physische Phase — **Implementierung mit Software**
- Modellierung der Datenbankstruktur mit einem relationalen Datenbankmanagementsystem (z. B. MS-Access)

DBMS unabhängig

DBMS abhängig

Hinweis: Ein **Modell** ist eine zweckorientiert vereinfachte und strukturgleiche Abbildung der Wirklichkeit. Für Modell wird auch der Begriff Schema verwendet. Zum Zweck der Verarbeitung gebildete Informationen (zweckorientiertes Wissen) heißen **Daten**.

3.1 Informationsstruktur ermitteln

In der externen Phase wird die Informationsstruktur des Datenmodells geplant. Dazu wird der Informationsbedarf der Benutzer ermittelt und strukturiert. Es muss herausgefunden werden, welche Informationen das Datenbanksystem liefern soll (Output) und welche Informationen dafür bereitzustellen sind (Input). Aus dieser Analyse ergibt sich die Informationsstruktur. Der Input bildet später die Datenbasis der Datenbank (z. B. Geschäftsobjekte wie Schüler und Klassen), der Output die zu erzielenden Ergebnisse, die Benutzersichten, z. B. in Form von Berichten und Formularen (Darstellungsobjekte wie Schülererfassungsmasken, Klassenlisten und Zeugnisse).

Es lassen sich zwei grundsätzliche Ansätze unterscheiden:

- **Top-down-Ansatz (globales Datenmodell)**

  Die Informationsanforderungen aller späteren Datenbanknutzer (nicht die einzelne Anwendung) bestimmt die Informationsstruktur. Beispiel: Alle für die Schule relevanten Objekte (Schüler, Lehrer, Klassen, Fächer, Eltern, Räume, Ausstattung usw.) werden erfasst. Es wird zunächst eine grobes Datenmodell entworfen und dann schrittweise verfeinert, so dass einzelne Anwendungen (Applikationen) entstehen.

- **Bottum-up-Ansatz (anwendungsorientiertes Datenmodell)**

  Ein spezielles Problem ist Ausgangspunkt für die Datenbankentwicklung. Für die Lösung des Problems wird eine Anwendung entwickelt. Beispiel: Um Zeit zu sparen, sollen die Zeugnisse und Schulbesuchsbescheinigungen über eine EDV-Anlage ausgefertigt werden. Die Integration der einzelnen Anwendungen kann zu einem globalen Datenmodell führen.

3.1.1 Ermittlung aufgrund von Realitätsbeobachtungen

Dieses Verfahren eignet sich zur Konstruktion eines globalen wie auch eines anwendungsorientierten Grobmodells, kann aber auch bei der Verfeinerung eingesetzt werden.

Betrachtet man die Miniwelt „Schule“, erkennt man gewisse Objekte und Objektmengen wie das Fach *Französisch*, einen Schüler namens *Meier*, einen Lehrer namens *Müller*, einen Schulleiter namens *Direx*, einen Klassenleiter namens *Huber*, einen Raum mit der Nummer *112*. Für Objekt wird auch der Begriff Entität oder Instanz verwendet; entsprechend für Objektmenge der Begriff Entitätsmenge.

Um das Beispiel nicht zu komplex zu gestalten, soll eine Beschränkung auf die Objektmengen ***Schüler***, ***Lehrer*** und ***Klassen*** vorgenommen werden. Zwischen diesen Objekten bestehen Beziehungen, die bestimmte Abläufe (Prozesse) oder Abhängigkeiten in der Miniwelt darstellen. Die Unterrichtsbelegung stellt die Beziehung zwischen einem Lehrer und den von ihm unterrichteten Klassen dar.

Um zum Datenmodell zu gelangen, stellt man Objekte und Beziehungen mit gleichartigen Attributen zu Typen zusammen. Somit ergeben sich für das gewählte Beispiel

- die **Objekttypen** *Schüler, Lehrer* und *Klassen*
- der **Beziehungstyp** *unterrichtet*

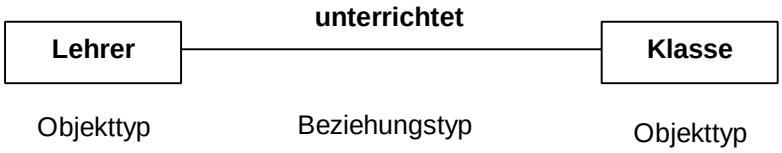


Bild 3-1: **Objekttypen der Klassenverwaltung**

Alle Lehrer bilden die **Objektmenge** *Lehrer*, alle Klassen die Objektmenge *Klasse* und alle Unterrichtsbelegungen die **Beziehungsmenge** *unterrichtet*.

Jeder Objekt- und Beziehungstyp definiert bestimmte **Eigenschaften (Merkmale, Attribute)**. Die Attribute des Objekttyps Lehrer sind: *Lehrernummer, Nachname, Vorname, Amtsbezeichnung, Fächerkombination*.

Für den Objekttyp Klasse lassen sich z. B. folgende Eigenschaften feststellen:

Klassenbezeichnung, Ausbildungsrichtung, Klassenleiter L-Nr, Klassensprecher S-Nr.

Dem Beziehungstyp *unterrichtet* lassen sich z. B. die Merkmale *L-Nr, Klassenbez, Fach* zuordnen.

Hinweis: Die Begriffe Entität und Entitätsmenge werden häufig synonym verwendet.

3.1.2 Ermittlung aufgrund von Benutzersichtanalysen

Die Benutzersicht ist die Sicht, aus der der einzelne Benutzer die Daten sieht. Benutzersichten stellen zum Beispiel Formulare und Berichte dar.

Bei der Benutzersichtanalyse können z. B. die Klassenlisten, Lehrerlisten und Zeugnisausdrucke als Grundlage für die Datenanalyse herangezogen werden. So kann aus dem Zeugnis oder einer Klassenliste die Informationsstruktur abgeleitet werden.

Mit Hilfe dieses Verfahrens lässt sich ein Grobmodell verfeinern und ein detailliertes, anwendungsorientiertes Datenmodell erstellen.

3.1.3 Ermittlung aufgrund von Datenbestandsanalysen

Dieses Verfahren ermöglicht die Integration existierender Datenbestände in ein neues Datenmodell. Beispielsweise soll in der Schulverwaltung von der Dateiorganisation zur Datenbankorganisation übergegangen werden. Der Übergang von einem System auf ein anderes wird als **Migration** bezeichnet.

3.2 Datenstruktur modellieren

Um eine Datenbank aufzubauen, muss ein möglichst exaktes Abbild der relevanten Welt definiert werden, ein **Datenmodell**. Der Teilausschnitt der Welt, der zu analysieren ist, wird **Miniwelt** genannt. Das Datenmodell soll die Miniwelt mit ihrem vollständigen Bedeutungsgehalt (semantisch vollständig) abbilden. Dazu müssen alle relevanten Objekte und Beziehungen zwischen den Objekten in einem semantischen Datenmodell erfasst und beschrieben werden. Diesen Abschnitt bezeichnet man auch als **konzeptionelle Phase**.

Bei der Strukturierung, Bearbeitung und Auswertung von Daten soll mit dem Entity-Relationship-Modell (E-R-Modell) gearbeitet werden. Danach wird eine Transformation in ein relationales Modell durchgeführt, da sich das relationale Datenbankmodell als das leistungsfähigste bewährt hat. Die Elemente des E-R-Modells sind **Entitäten, Beziehungen zwischen Entitäten** und **Attribute** zur Charakterisierung von Entitäten.

3.2.1 Elemente des E-R-Modells

3.2.1.1 Entitäten

Eine Entität kann sein:

- ein Individuum (z. B. der Schüler *Meier*)
- ein reales Objekt (z. B. der Raum mit der Nr *112*)
- ein abstraktes Konzept (z. B. der Kurs *Informatik*)
- ein Ereignis (z. B. eine mündliche Prüfung)

Definition:	Eine **Entität** (engl. entity) ist eine eindeutig identifizierbare Einheit.

Definition:	Eine **Entitätsmenge** (engl. entity set) fasst alle Entitäten zusammen, die durch gleiche Merkmale, nicht notwendigerweise aber durch gleiche Merkmalsausprägungen, charakterisiert sind.

Zur grafischen Darstellung von Entitätsmengen werden Rechtecke verwendet. Bild 3-2 zeigt einige Beispiele.

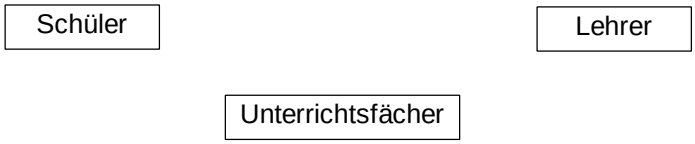


Bild 3-2: Entitäten

Die einzelnen Entitäten einer Entitätsmenge unterscheiden sich in verschiedenen Werten der Attribute bzw. Attributkombinationen (vgl. Bild 3-7).

3.2.1.2 Beziehungen zwischen Entitäten

Eine Beziehung assoziiert wechselseitig zwei (oder mehrere) Entitäten.

Assoziation bedeutet, dass eine Entität eine andere Entität kennt und mit ihr in Wechselwirkung steht.

Die **Kardinalität** einer Assoziation *a(E1, E2)* gibt an, wie viel Entitäten der Entitätsmenge *E2* einer beliebigen Entität der Entitätsmenge *E1* zugeordnet sein können.

Definition:	Eine **Beziehung** zwischen zwei Entitätsmengen *E1* und *E2* besteht aus der Assoziation *a(E1, E2)* und der dieser Assoziation entgegen gerichteten Assoziation *a*(E2, E1)*.

Beispiel: Eine Klasse kann von mehreren Schülern besucht werden (*a(Klasse, Schüler)*). Ein Schüler besucht genau eine Klasse (*a*(Schüler, Klasse)*).

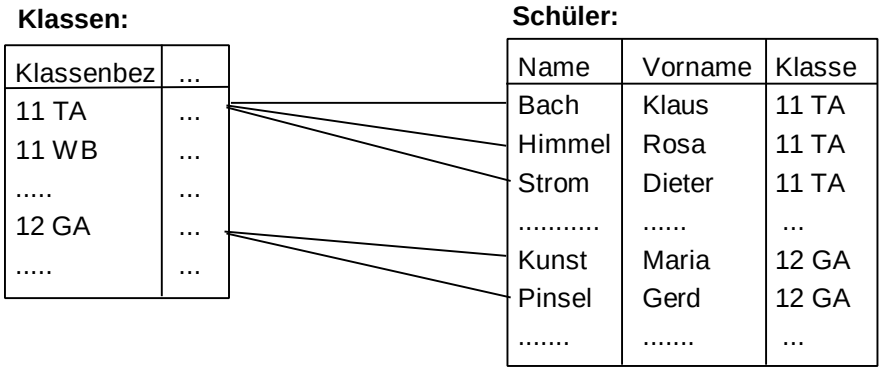


Bild 3-3: Beziehung zwischen den Entitäten *Schüler* und *Klassen*

Definition:	Die Kardinalität einer **Assoziation** *a(E1, E2)* gibt an, wie viele Entitäten der Entitätsmenge *E2* einer beliebigen Entität der Entitätsmenge *E1* zugeordnet sein können.

Zur grafischen Darstellung eines Beziehungstyps verwendet man, wie Bild 3-4 zu entnehmen ist, eine Raute. Dabei werden die Verbindungen zu den entsprechenden Entitätsmengen durch Linien repräsentiert.

Hinweis: Für den Kardinalitätstyp *mehrfach* werden die Symbole m oder n verwendet.

Bild 3-4: Grafische Darstellung eines Beziehungstyps

Wichtig für die späteren Betrachtungen sind drei verschiedene Arten von Beziehungen:

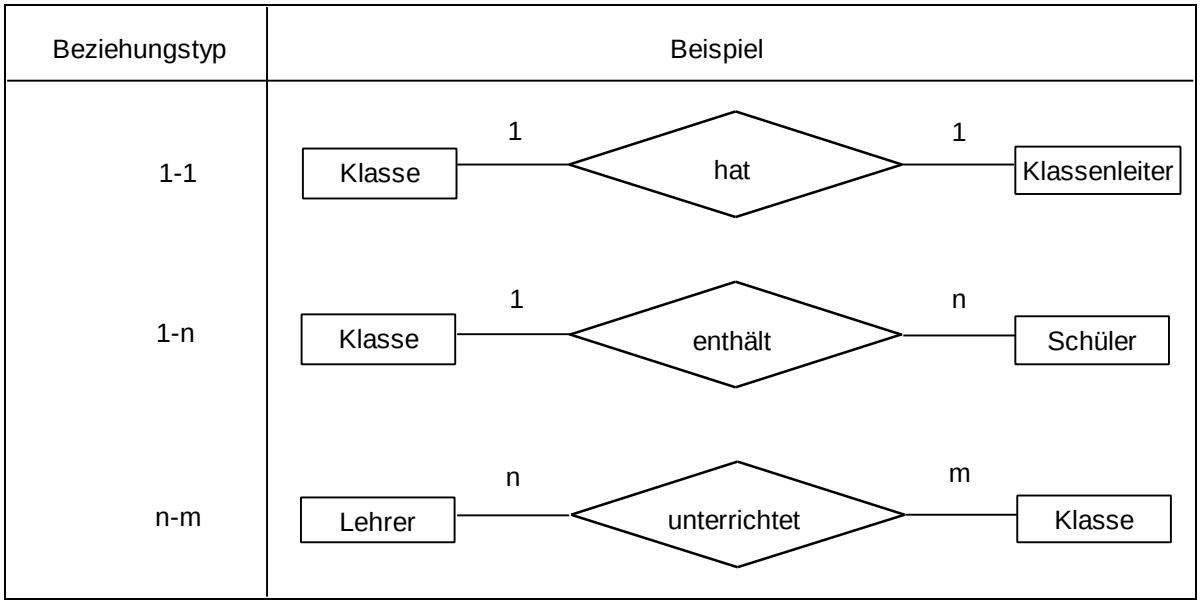


Bild 3-5: Beispiele für Beziehungen

3.2.1.3 Attribute

Über das Wesen der abgebildeten Phänomene machen Entitätsmengen und Beziehungen nur grobe Aussagen. Interessierende Eigenschaften von Phänomenen lassen sich durch Attribute (Merkmale) erfassen, die man den Entitätsmengen und Beziehungen zuordnet. Auf der physikalischen Ebene wird für Attribut auch der Begriff Feld verwendet.

Definition:	Ein **Attribut** (engl. attribute) beschreibt eine bestimmte Eigenschaft, die sämtliche Entitäten einer Entitätsmenge oder sämtliche Einzelbeziehungen einer Beziehung aufweisen.

Wie bei einer Variablen ist bei einem Attribut zwischen seinem Namen und seinem Wert zu unterscheiden. Während der Name ein Attribut benennt und identifiziert, gibt der Wert die konkrete Ausprägung des Attributs für eine bestimmte Entität oder Beziehung an. Welche diskreten Werte ein Attribut annehmen kann, legt sein Wertebereich (domain) fest.

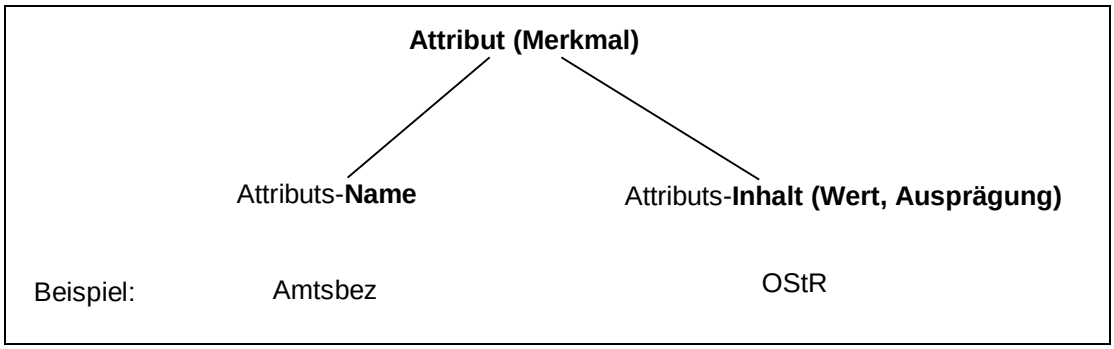


Bild 3-6: Attributnamen und -wert

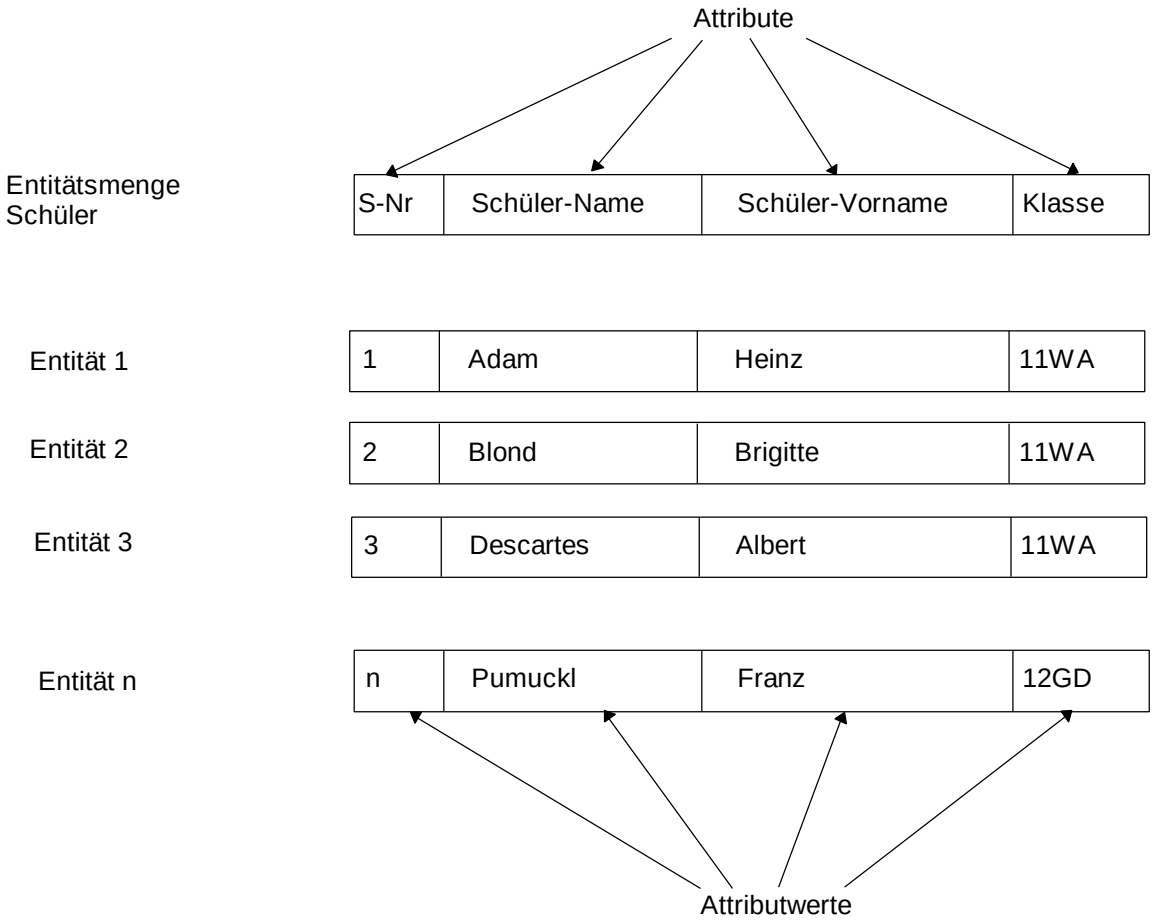


Bild 3-7: Attribute und Attributwerte der Entitätsmenge Schüler

Wertebereiche (Domänen) der Attribute:

Wertebereich von *S-Nr*:	[1..2000],
Wertebereich von *Schüler-Name*:	Zeichenkette der Länge 50,
Wertebereich von *Schüler-Vorname*:	Zeichenkette der Länge 40,
Wertebereich von *Klasse*:	Zeichenkette der Länge 5.

3.2.2 Umsetzung des semantischen Modells in ein logisches Datenbankmodell

Das semantische Modell hat zwei Konstruktionselemente: Entitätsmengen und Beziehungen.
Das logische Modell (relationale Modell) kennt aber nur ein Konstruktionselement: die Tabelle
Aus diesem Grund können 1-1- und 1-n-Beziehungen unmittelbar vom semantischen ins logische Modell abgebildet werden, aber keine n-m-Beziehungen.

Der Übergang vom semantischen Datenmodell zum relationalen Datenmodell ist nicht ohne weiteres möglich, da das relationale Modell zwar 1-1- und 1-n-Beziehungen unmittelbar abbilden kann, aber **keine komplex-komplexen Beziehungen** (n-m-Beziehungen). Die im semantischen Modell definierten Klassen entsprechen deshalb in der Regel nicht der Struktur der Relationen (Tabellen) des logischen Datenbankschemas. Komplex-komplexe Beziehungen können jedoch in 1-n-Beziehungen (einfach-komplexe Beziehungen) aufgelöst werden, wenn für die Beziehungsmenge eine eigenständige Tabelle (Relation) definiert wird. Auch wenn für die Beziehungsmenge eine eigene Relation konstruiert wurde, kann zwischen dieser Relation und einer Entitätsmenge noch eine n-m-Beziehung bestehen. Die Beziehungsmengen-Relation muss dann in eine zweite Beziehungsmengen-Relation (Detailtabelle) unterteilt werden.

Eine feste Beziehung (Verknüpfung) zwischen den Relationen (Tabellen) wird über Fremdschlüssel hergestellt. Der Fremdschlüssel muss in der anderen Relation (Tabelle) ein Primärschlüssel (Identifikationsschlüssel) sein.

3.2.3 Identifikationsschlüssel

Jeder Datensatz einer Tabelle muss eindeutig identifizierbar sein. Dies kann durch ein Feld oder eine Kombination von Feldern gewährleistet werden.

Ein **Identifikationsschlüssel** (engl. identification key) besteht aus einem Attribut oder aus einer Kombination von Attributen, welche jede Entität einer Entitätsmenge eindeutig identifiziert.

Bei der in Bild 3-7 angegebenen Entitätsmenge *Schüler* ist z. B. das Attribut *S-Nr* oder die Attributkombination aus *Name, Vorname* und *Klasse* grundsätzlich als Identifikationsschlüssel geeignet. Nicht geeignet ist dagegen ein einzelnes Attribut (außer *S-Nr*), da es für verschiedene Entitäten den gleichen Wert annehmen kann.

Hinzuweisen ist noch auf zwei Schlüsselbegriffe, den Primärschlüssel und den Sekundärschlüssel. Ein **Primärschlüssel** (engl. primary key) identifiziert eindeutig die Entitäten einer Entitätsmenge und ist daher zugleich auch ein Identifikationsschlüssel. Dagegen ist ein **Sekundärschlüssel** ein Attribut, welches für mehrere Entitäten den gleichen Wert annehmen kann. Ein Sekundärschlüssel dient also der Zusammenfassung von Entitäten zu Teilmengen mit einer gleichen Eigenschaft und hilft zum schnelleren Suchen bzw. Sortieren. Beispielsweise könnte das Attribut *Klasse* einer Entitätsmenge *Schüler* als Sekundärschlüssel Verwendung finden.

Ein **Fremdschlüssel** *FS* in einer Tabelle 2 ist ein Feld oder eine Feldkombination, welche(s) in einer Tabelle 1 den Primärschlüssel *PS* bildet. *PS* ist mit *FS* verknüpft. Bei dem Beispiel in Bild 3-7 ist die *Klassenbezeichnung* Fremdschlüssel der Tabelle (Entitätsmenge) *Schüler* und Primärschlüssel in einer Tabelle (Entitätsmenge) *Klassen*. In dem später behandelten Beispiel der Vereinsdatenbank werden die Primärschlüssel in der Feldbezeichnung mit der Endung *ID* gekennzeichnet und die zugehörigen Fremdschlüssel mit der Endung *Nr* (z. B. *M-ID* bzw. *M-Nr* sowie *Ort-ID* bzw. *Orts-Nr* und *Sport-ID* bzw. *Sport-Nr*). Dies dient der besseren Verständlichkeit, ist aber nicht notwendig; genauso könnten die beiden Felder denselben Feldnamen besitzen.

3.2.4 Entwicklung des relationalen Datenbankschemas mit Hilfe von Abbildungsregeln

Das semantischen Modell kann im Einzelnen mit den nachfolgenden **Abbildungsregeln** in ein logisches Modell umgesetzt werden:

Entitätsmenge, Beziehung		Abbildungsregel für das relationale Modell
Entitätsmenge	**Regel 1**	Jede Entitätsmenge muss als eigenständige Relation definiert werden.
1:1-Beziehung	**Regel 2**	Es genügt eine Relation. Aus Datenschutzgründen oder für selten benötigte Informationen kann eine eigene Relation definiert werden.
1:n-Beziehung	**Regel 3**	Primärschlüssel der 1-Relation ist Fremdschlüssel in der n-Relation. Keine eigene Beziehungsmengen-Relation notwendig.
m:n- -Beziehung	**Regel 4**	Jede komplex-komplexe Beziehungsmenge muss als eigenständige Relation definiert werden. Die Primärschlüssel der zugehörigen Entitätsmengen treten als Fremdschlüssel in der Beziehungsmengen-Relation auf.

Tabelle 3-1: Abbildungsregeln

3.2.5 Normalformen

Das Verständnis der nachfolgenden Regeln hilft, die im vorherigen Abschnitt verwendeten Abbildungsregeln zu untermauern. Bei einem sauberen Entwurf nach dem E-R-Modell und Anwendung der Abbildungsregeln befinden sich die erzeugten Relationen häufig schon in der 3. Normalform. Da aber beim Entwurf des semantischen Datenmodells die Attribute nicht systematisch daraufhin untersucht werden, ob sie nur elementare Werte enthalten und ob nicht unerwünschte funktionale Abhängigkeiten zwischen ihnen bestehen, sind die Normalformen nicht immer automatisch erfüllt. Es ist deshalb zweckmäßig, mit Hilfe der nachfolgenden Regeln zu überprüfen, ob die Relationen den Normalformen genügen.

Tabellen (Relationen) sollten so geplant werden, dass

- logische Widersprüche (Inkonsistenzen, Anomalien) in der Datenbasis und
- Datenredundanz (Mehrfachspeicherung gleicher Daten) vermieden sowie
- eine höchstmögliche Flexibilität und
- schneller Zugriff gewährleistet werden.

Deshalb sollten beim Entwurf von Tabellen die nachfolgenden Normalformen beachtet werden:

Erste Normalform (1 NF)

Eine Relation (Tabelle) ist in der ersten Normalform, wenn die Werte der Attribute elementar (atomar) sind.

Damit sind z. B. Mengen, Folgen und Arrays als Attributwerte ausgeschlossen. Elementar bedeutet, dass der Wert als Ganzes zu sehen ist und keine Teile des Wertes getrennt verarbeitet werden sollen.

Beispiel 1:

M-ID	Mitglied		Geburtsdatum
1	Ulrich Becker Maxweg 14 85408 Gammelsdorf	Tel.: 08234 / 123 XYZ	01-02-73
2	Julia Berger Fischweg 22 85395 Attenkirchen	Tel.: 08123 / 456 ABC	02-04-80

Im obigen Beispiel sind die Werte des Attributs *Mitglied* nicht elementar. Zum Beispiel wird bei einem Schreiben an das Mitglied im Adressfeld die Telefonnummer nicht gebraucht. Das Attribut *Mitglied* sollte in die Attribute *Nachname, Vorname, Straße, PLZ, Ort*, und *Telefon* aufgespaltet werden.

Das Geburtsdatum setzt sich zwar aus den Komponenten Tag, Monat und Jahr zusammen, es ist aber als Ganzes anzusehen. Dem widerspricht nicht, dass über spezielle Funktionen auf einzelne Komponenten des Datums zugegriffen werden kann (z. B.: Gruppiere die Geburtstage der Mitglieder nach Monaten).

In einigen relationalen Datenbanksystemen können auch komplexe Objekte wie Bilder, Videosequenzen und Tonfolgen als elementare Attribute gespeichert werden. Dies bedingt, dass keine Abfragen auf bestimmte Inhalte der Attribute durchgeführt werden können (z. B.: Suche nach einer bestimmten Tonfolge).

Wichtiger Hinweis: Die weiteren Normalformen setzen die jeweils vorhergehende Normalform voraus, ohne dass dies im Folgenden ausdrücklich gesagt wird.

Zweite Normalform (2 NF)

Die zweite Normalform ist nur relevant, wenn sich der Primärschlüssel aus mehreren Attributen zusammensetzt.

Eine Relation (Tabelle) ist in der zweiten Normalform, wenn alle Nichtschlüssel-Attribute voll funktional abhängig vom Primärschlüssel sind.

Beispiel 2: Der Primärschlüssel wird durch die Attribute *M-Nr* und *Sport-Nr* gebildet.

Relation *Abteilungszugehörigkeit*

M-Nr	**Sport-Nr**	Abteilungsname
1	1	Fußball
1	2	Handball
2	3	Schwimmen
3	1	Fußball

Kritik:

Wäre der *Abteilungsname* vom zusammengesetzten Primärschlüssel *M-Nr, Sport-Nr* voll funktional abhängig, dann müsste er einen anderen Wert annehmen, wenn sich der zusammengesetzte Primärschlüssel ändert. Dies ist jedoch nicht der Fall. Vielmehr ist der *Abteilungsname* nur von einem Teil des Primärschlüssels abhängig, nämlich der *Sport-Nr*. Ändert sich die *Sport-Nr*, dann ändert sich auch der *Abteilungsname*. Es besteht also keine volle funktionale Abhängigkeit des Attributs *Abteilungsname* vom Primärschlüssel. Die Relation *Abteilungszugehörigkeit* verstößt damit gegen die zweite Normalform.

Lösung:

Das Attribut *Abteilungsname* gehört zu einer eigenen Relation *Sportabteilungen* und nicht zur Relation *Abteilungszugehörigkeit*.

Dritte Normalform (3 NF)

Eine Relation befindet sich in der dritten Normalform, wenn alle Nichtschlüssel-Attribute voneinander funktional unabhängig sind.

Beispiel 3: Der Primärschlüssel ist die M-ID.

Relation *Mitglieder*

M-ID	Nachname	Geburtsdatum	Alter
1	Becker	01-02-73	25
2	Berger	02-04-80	18
3	Fischer	05-06-81	17

Kritik:

Das Alter ist vom Geburtsdatum abhängig. Ändert sich der Zeitpunkt der Ansicht der Tabelle, so kann sich auch das Alter ändern (auf jeden Fall ein Jahr später). Das Geburtsdatum ist kein Primärschlüsselattribut und somit ein Nichtschlüsselattribut. Also ist das Alter von einem Nichtschlüssel-Attribut funktional abhängig. Damit liegt ein Verstoß gegen die dritte Normalform vor.

Lösung:

Das Alter kann aus dem Geburtsdatum jederzeit in einer Abfrage berechnet werden. Es ist also überflüssig, es als Attributwert zu speichern.

Beispiel 4:

M-ID	Nachname	Vorname	...	Abteilung1	Abteilung2
1	Becker	Ulrich	...	Fußball	Handball
2	Berger	Julia	...	Schwimmen	
3	Fischer	Otto	...	Leichtathletik	Fußball

Kritik:

Damit ist zwar die dritte Normalform erfüllt (kein Nichtschlüssel-Attribut ist von einem anderen Nichtschlüssel-Attribut abhängig). In der konkreten Anwendung kann diese Lösung jedoch zu Problemen führen:

- Ein Mitglied kann in mehr als zwei Abteilungen sein, so dass neue Spalten eingefügt werden müssen.
- Die Formulierung von Abfragen wird unnötig verkompliziert: Eine Abfrage „Welche Mitglieder sind in der Fußball-Abteilung?“ wäre zu formulieren als „Bei welchen Mitgliedern ist Abteilung1 = *Fußball* oder „bei welchen Mitgliedern ist Abteilung2 = *Fußball*“?

Saubere Lösung des Problems: Für die Aufzählungswerte eine eigene Tabelle definieren.

Mitglieder

M-ID	Nachname
1	Becker
2	Berger
3	Fischer

1 : n

Abteilungszugehörigkeit

M-Nr	**Sport-Nr**
1	1
1	2
2	3
3	1
3	4

n : 1

Sportabteilungen

Sport-ID	Abteilung
1	Fußball
2	Handball
3	Schwimmen
4	Leichtathletik

3.2.6 Strukturelle Integritätsbedingungen

Unter dem Begriff Integrität oder Konsistenz versteht man die Widerspruchsfreiheit von Datenbeständen. Eine Datenbank ist integer oder konsistent, falls die gespeicherten Daten fehlerfrei erfasst sind und den gewünschten Informationsgehalt korrekt wiedergeben. Die Datenintegrität ist dagegen verletzt, wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte auftreten. Bei einer konsistenten Tabelle *Lehrer* setzt man beispielsweise voraus, dass *L-Namen, L-Vornamen, Amtsbezeichnungen* etc., korrekt sind und real auch existieren.

Strukturelle Integritätsbedingungen zur Gewährleistung der Integrität sind solche Regeln, die durch das Datenbankschema selbst ausgedrückt werden können. Bei relationalen Datenbanken zählen die folgenden Integritätsbedingungen zu den strukturellen:

- **Die Eindeutigkeitsbedingung**: Jede Tabelle besitzt einen Identifikationsschlüssel (Merkmal oder Merkmalskombination), der jedes Tupel in der Tabelle auf eindeutige Art bestimmt.
- **Die Wertebereichsbedingung**: Die Merkmale einer Tabelle können nur Datenwerte aus einem vordefinierten Wertebereich annehmen.

- **Die referentielle Integritätsbedingung**: Jeder Wert eines Fremdschlüssels muss effektiv als Primärschlüsselwert in der referenzierten Tabelle existieren.

Die **Eindeutigkeitsbedingung** verlangt pro Tabelle einen festgelegten Schlüssel. Natürlich können innerhalb derselben Tabelle mehrere Schlüsselkandidaten vorkommen. In diesem Fall muss aufgrund der Eindeutigkeitsbedingung ein Schlüssel als **Primärschlüssel** deklariert werden. Das Prüfen auf Eindeutigkeit von Primärschlüsseln wird dabei vom Datenbanksystem übernommen.

Die **Wertebereichsbedingung** kann hingegen nicht vollständig vom Datenbanksystem gewährleistet werden. Zwar können die Wertebereiche einzelner Tabellenspalten spezifiziert werden, doch decken solche Angaben nur einen geringen Teil der Validierungsregeln ab. Bei der Prüfung von Ortsbezeichnungen oder Straßennamen auf Korrektheit ist es mit der Definition eines Wertebereichs nicht getan; so sagt beispielsweise die Spezifikation *Character (20)* für eine Zeichenkette nichts über sinnvolle Orts- oder Straßennamen aus. Wie weit man bei einer Validierung von Tabelleninhalten gehen möchte, bleibt weitgehend dem Anwender überlassen.

Eine wichtige Klasse von Prüfregeln wird unter dem Begriff **referentielle Integrität** zusammengefasst. Eine relationale Datenbank erfüllt die referentielle Integritätsbedingung, wenn jeder Wert eines Fremdschlüssels als Wert beim zugehörigen Primärschlüssel vorkommt. Das Bild 3-8 macht dies deutlich: Die Tabelle *Klassen* hat die Klassenbezeichnung *Klassenbez* als Primärschlüssel. Dieser wird in der Tabelle *Schüler* als Fremdschlüssel mit dem Merkmal *Klasse* verwendet, um die Klassenzugehörigkeit eines Schülers zu fixieren. Die Fremd-Primärschlüssel-Beziehung erfüllt die Regel der referentiellen Integrität, falls alle Klassenbezeichnungen des Fremdschlüssels aus der Tabelle *Schüler* in der Tabelle *Klassen* als Primärschlüsselwerte aufgeführt sind. In dem Beispiel sieht man, dass keine Unterstellung die Regel der referentiellen Integrität verletzt.

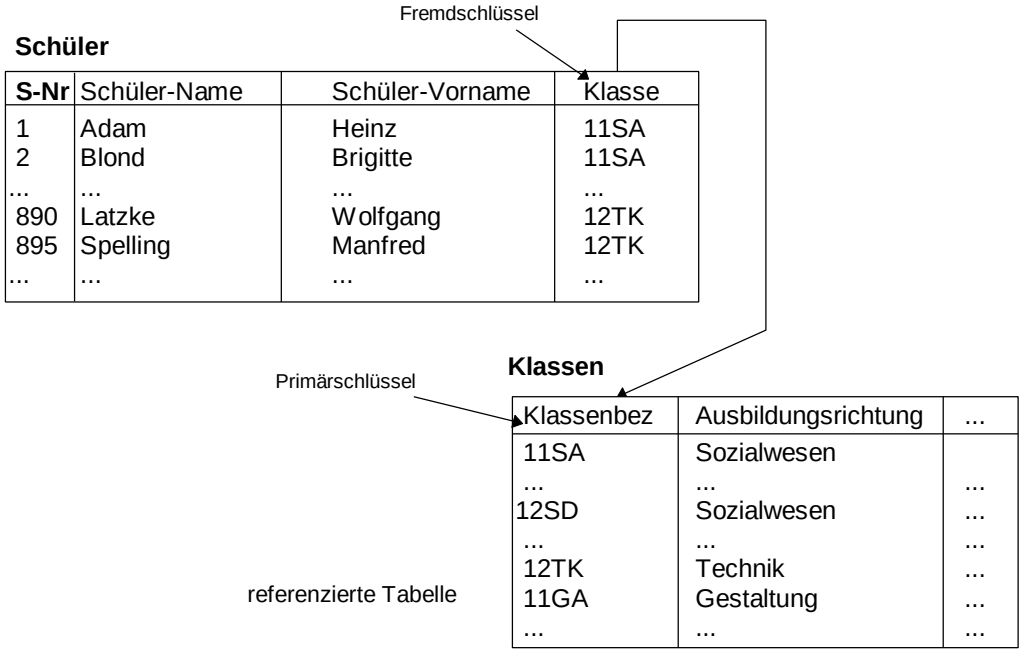


Bild 3-8: Gewährleistung der referentiellen Integrität

Nimmt man an, man möchte in die Tabelle *Schüler*, wie sie Bild 3-8 zeigt, ein neues Tupel *1567, Müller, Tobias, 12SF* einfügen. Die Einfügeoperation wird abgewiesen, falls das Datenbanksystem die referentielle Integrität unterstützt. Der Wert 12SF wird nämlich als ungültig erklärt, da dieser Wert in der referenzierten Tabelle *Klassen* nicht vorkommt.

Die Gewährleistung der referentiellen Integrität hat neben der Einfügeproblematik natürlich Einfluss auf die übrigen Datenbankoperationen. Wird ein Datensatz in einer Tabelle gelöscht, der von anderen Datensätzen aus einer Fremdtabelle referenziert wird, so sind mehrere Varianten von Systemreaktionen möglich (eine davon sei hier erwähnt):

Bei der sogenannten **restriktiven Löschung** wird eine Löschoperation nicht ausgeführt, solange der zu löschende Datensatz auch nur von einem Datensatz einer Tabelle referenziert wird. Möchte man z. B. den Datensatz *12TK, Technik, ...* in Bild 3-8 entfernen, so wird nach der restriktiven Löschregel die Operation verweigert, da die beiden Schüler *Spelling* und *Latzke* noch die Klasse 12TK besuchen. Erst nach dem Löschen sämtlicher Schüler der Klasse 12 TK ist ein Löschen der gesamten Klasse möglich.

4 Beispiel einer Datenbank, Teil II

4.1 Semantisches Modell

Aufgrund der Informationsstruktur aus Kapitel 1.1 ergibt sich das folgende semantische Datenmodell:

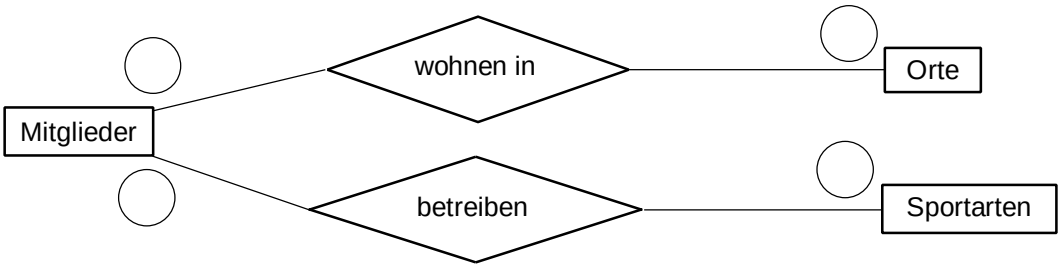


Bild 4-1: E-R-Diagramm des Modells *Sportverein*

Die Ermittlung der Beziehungstypen (Kardinalität der Beziehungen) geschieht folgendermaßen:

Beziehung *Mitglieder wohnen in Orte*:

a (Mitglieder, Orte)

1 ? In wie vielen Orten kann ein Mitglied wohnen?
1 Ein Mitglied kann in einem Ort wohnen.
Kardinalität der Assoziation: 1

a(Orte, Mitglieder)*

1 ? Wie viele Mitglieder können in einem Ort wohnen?
n In einem Ort können mehrere Mitglieder wohnen.
Kardinalität der Assoziation: n

Ergebnis 1: Die Beziehung *Mitglieder wohnen in Orte* ist eine 1-n-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Regel 3).

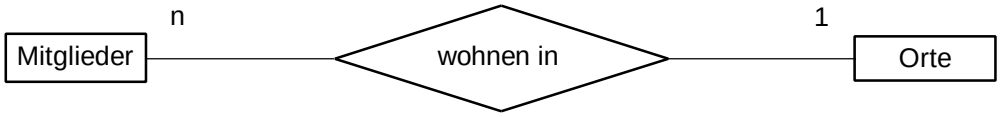


Bild 4-2: Beziehung *Mitglieder wohnen in Orte*

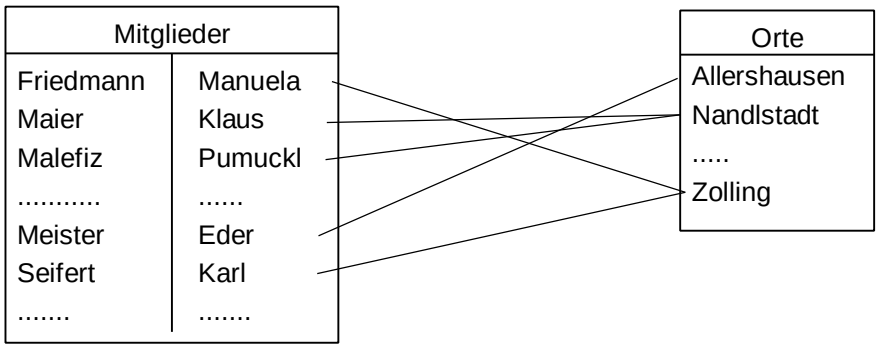


Bild 4-3: Beispieldaten für die Beziehung *Mitglieder wohnen in Orte*

Beziehung *Mitglieder betreiben Sportarten*:

a(Mitglieder, Sportarten)

1 ? Wie viele Sportarten kann ein Mitglied betreiben?
n Ein Mitglied kann eine oder mehrere Sportarten betreiben.
Kardinalität der Assoziation: n

a(Sportarten, Mitglieder)*

1 ? Wie viele Mitglieder können eine Sportart betreiben?
m Eine Sportart kann von mehreren Mitgliedern betrieben werden.
Kardinalität der Assoziation: m

Ergebnis 2: Die Beziehung *Mitglieder betreiben Sportarten* ist eine n-m-Beziehung; es ist eine eigene Beziehungsmengen-Relation notwendig (Regel 4).

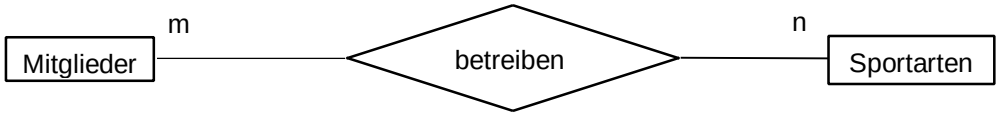


Bild 4-4: Beziehung *Mitglieder betreiben Sportarten*

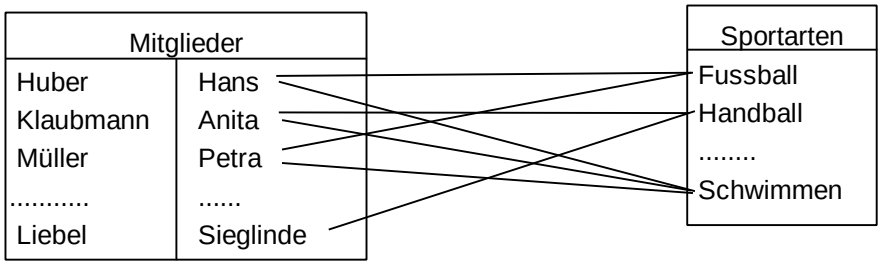


Bild 4-5: Beispieldaten für die Beziehung *Mitglieder betreiben Sportarten*

Es liegt also eine komlex-komplexe Beziehung (Regel 4) vor. Nach den Abbildungsregeln muss folglich für die Beziehungsmenge *betreiben* eine eigene Relation *Zuordnung: M-Nr-Sport-Nr* konstruiert werden.

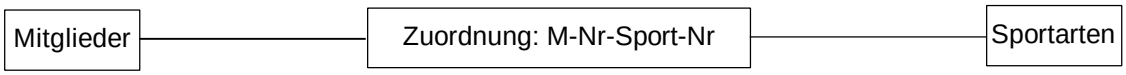


Bild 4-6: Relationen *Mitglieder, Zuordnung: M-Nr-Sport-Nr, Sportarten*

Im nächsten Schritt sind die sich aus der Informationsstruktur ergebenden Kardinalitätstypen der Assoziationen zu ermitteln:

Beziehung *Mitglieder - Zuordnung: M-Nr-Sport-Nr*

a(Mitglieder, Zuordnung: M-Nr-Sport-Nr)

1 ? Wie viele Sportarten kann ein bestimmtes Mitglied betreiben?
n Eine oder mehrere
Kardinalität der Assoziation: n

a(Zuordnung: M-Nr-Sport-Nr, Mitglieder)*

1 ? Wie oft kann ein bestimmtes Mitglied eine Sportart betreiben?
1 Einmal.
Kardinalität der Assoziation: 1

Ergebnis 3: Die Beziehung *Mitglieder - Zuordnung: M-Nr-Sport-Nr* ist eine 1-n-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Regel 3).

Bild 4-7: Beziehung *Mitglieder- Zuordnung: M-Nr-Sport-Nr*

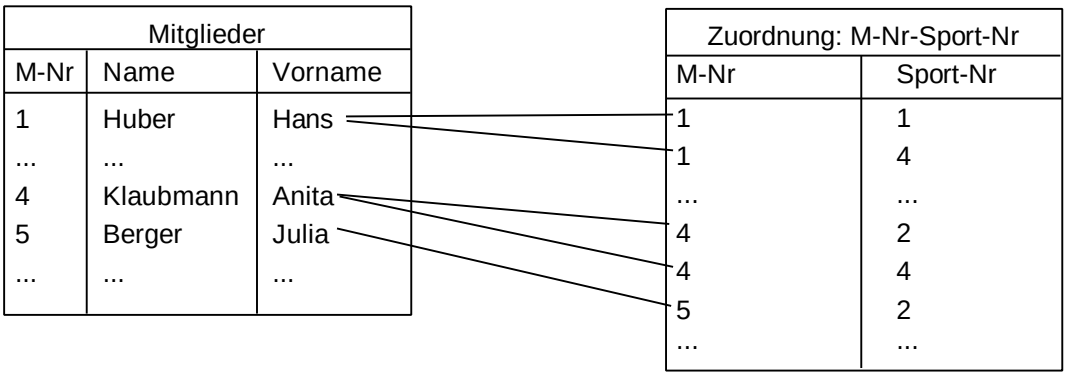


Bild 4-8: Beispieldaten für die Beziehung *Mitglieder - Zuordnung: M-Nr-Sport-Nr*

Beziehung *Zuordnung: M-Nr-Sport-Nr - Sportarten*:

a(Zuordnung: M-Nr-Sport-Nr, Sportarten)

1 ? Wie oft kann eine Sportart von einem bestimmten Mitglied betrieben werden?
1 Ein bestimmtes Mitglied kann eine Sportart einmal betreiben.
Kardinalität der Assoziation: 1

a(Sportarten, Zuordnung: M-Nr-Sport-Nr)*

1 ? Von wie vielen verschiedenen Mitgliedern kann eine Sportart betrieben werden?
n Eine Sportart kann von einem oder mehreren Mitgliedern betrieben werden.
Kardinalität der Assoziation: n

Ergebnis 4: Die Beziehung *Zuordnung: M-Nr-Sport-Nr - Sportarten* ist eine n-1-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Regel 3).

Bild 4-9: Beziehung *Zuordnung: M-Nr-Sport-Nr - Sportarten*

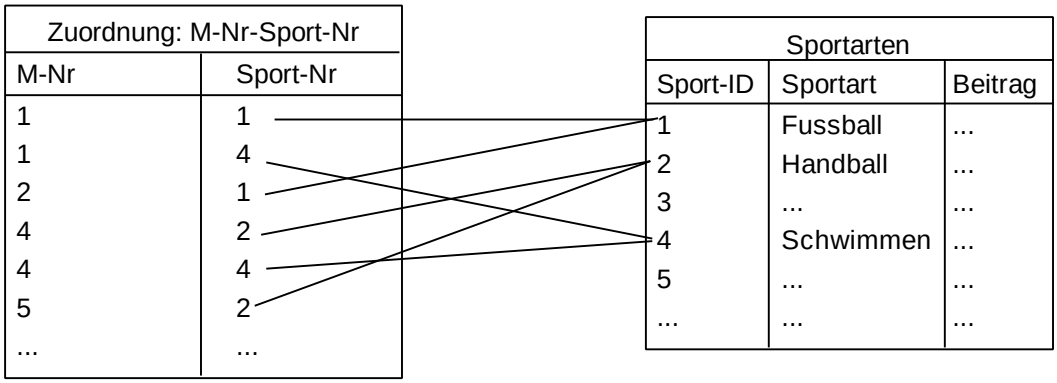


Bild 4-10: Beispieldaten für die Beziehung *Zuordnung: M-Nr-Sport-Nr - Sportarten*

Zusammenfassung: Insgesamt wurde die n-m-Beziehung*: Mitglieder-betreiben-Sportarten* durch die Konstruktion der Beziehungsmengen-Relation *Zuordnung: M-Nr-Sport-Nr* in die zwei Beziehungen *Mitglieder - Zuordnung: M-Nr-Sport-Nr* (1-m-Beziehung) und *Zuordnung: M-Nr-Sport-Nr - Sportarten* (n-1-Beziehung) aufgelöst. Die Beziehungsmengen-Relation *Zuordnung: M-Nr-Sport-Nr* muss auf der logischen Ebene als eigenständige Tabelle dargestellt werden.

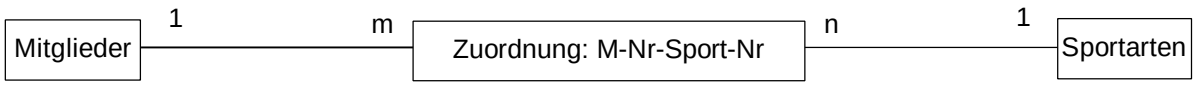


Bild 4-11: Beziehungen mit der Beziehungsmengen-Relation *Zuordnung: M-Nr-Sport-Nr*

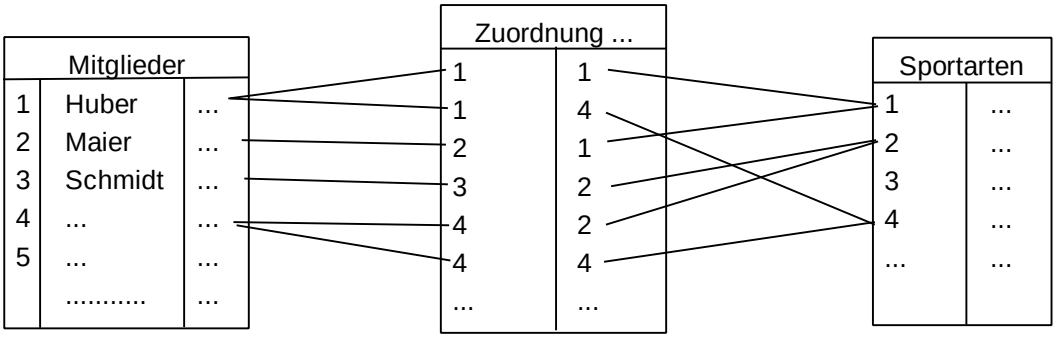


Bild 4-12: Beispieldaten für die Beziehungen *Zuordnung: M-Nr-Sport-Nr*

4.2 Zusammenfassung und relationales Modell

Regel 1: Jede Entitätsmenge muß als eigenständige Tabelle mit einem eindeutigen Primärschlüssel definiert werden.

Mitglieder
M-ID
Nachname
Vorname
Geburtsdatum
Geschlecht
Straße

PLZ
Ort-ID
PLZ
Ort

Sportarten
Sport-ID
Sportart
Beitrag

Regel 4: Bei komplex-komplexen Beziehungen muß eine eigenständige Beziehungsmengen-Tabelle definiert werden: die Primärschlüssel der zugehörigen Entitätsmengen treten als Fremdschlüssel in der Beziehungsmengen-Tabelle auf.

Zuordnung: M-Nr-Sport-Nr
M-Nr
Sport-Nr

Regel 3: Für alle 1:1- und 1:n-Beziehungen gilt:
Eine eigene Beziehungsmengen-Tabelle ist nicht notwendig: Aufnahme des Primärschlüssels der 1-Relation in der n-Relation (hier:Orts-Nr).

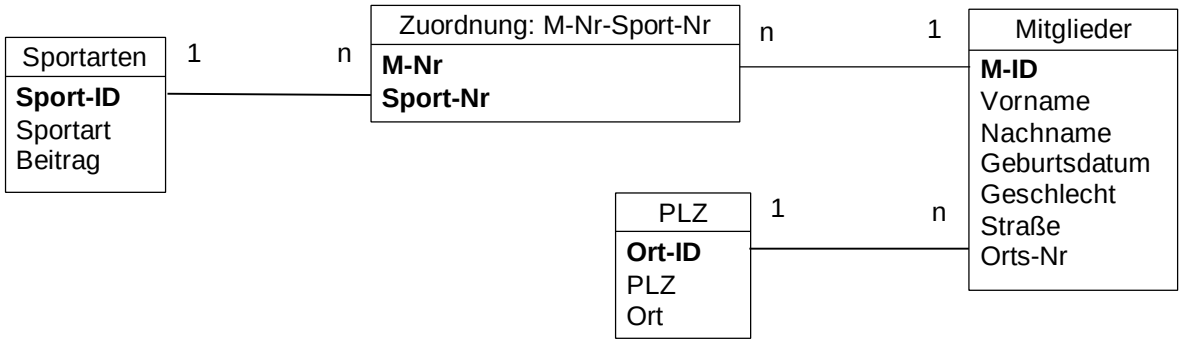


Bild 4-13: Entstehung des relationalen Datenbankschemas des Beispiels *Verein* im Überblick

Hinweis: Die Festlegung der Regel-Nummerierung erfolgt auf Seite 15!

4.3 Erweiterung des Modells

4.3.1 Fall 1: *Verein5* (ein Trainer / Betreuer pro Mannschaft)

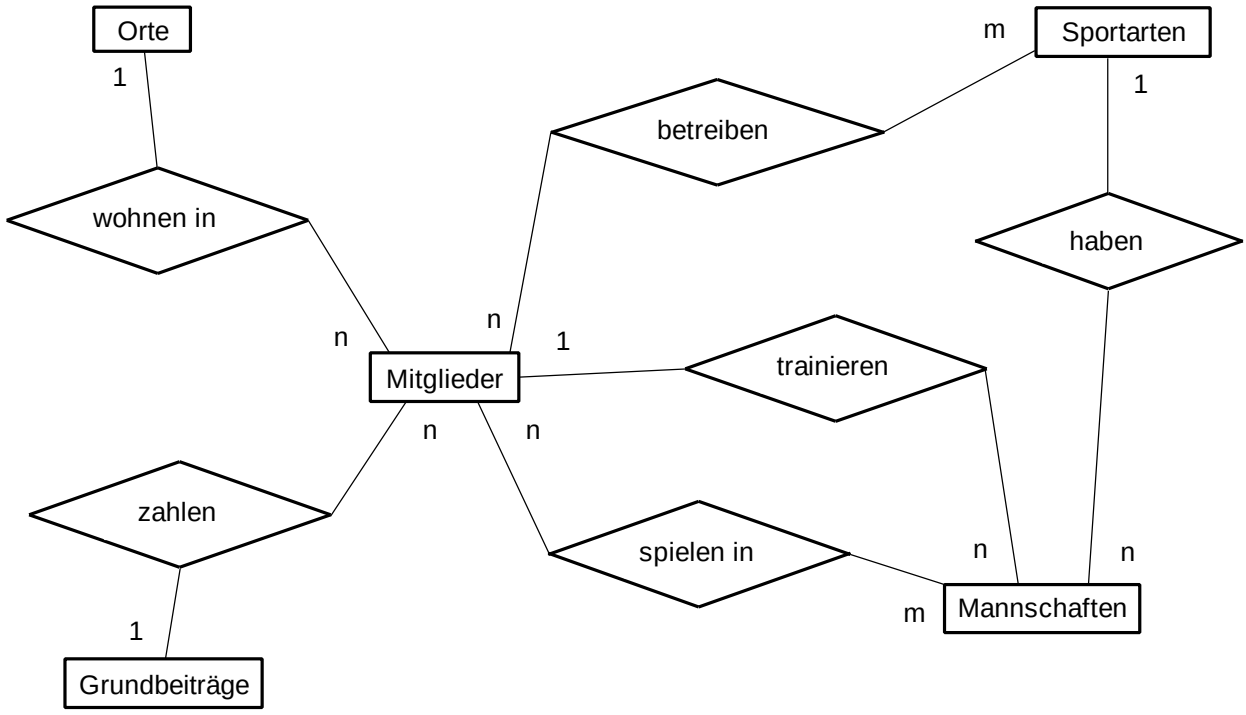


Bild 4-14: E-R-Diagramm der Datenbank *Verein5*

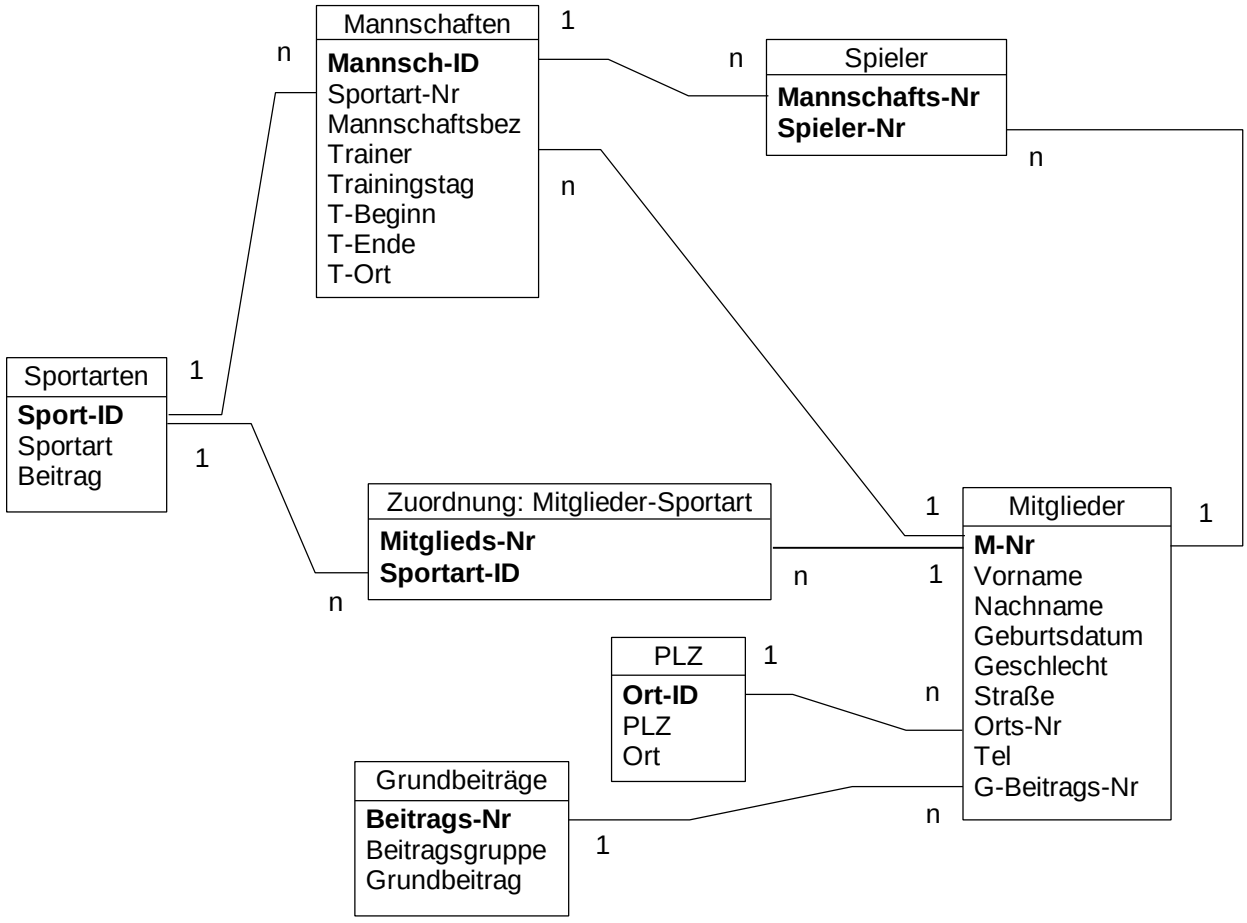


Bild 4-15: Relationales Modell der Datenbank *Verein5*

4.3.2 Fall 2: *Verein6* (mehrere Trainer / Betreuer pro Mannschaft)

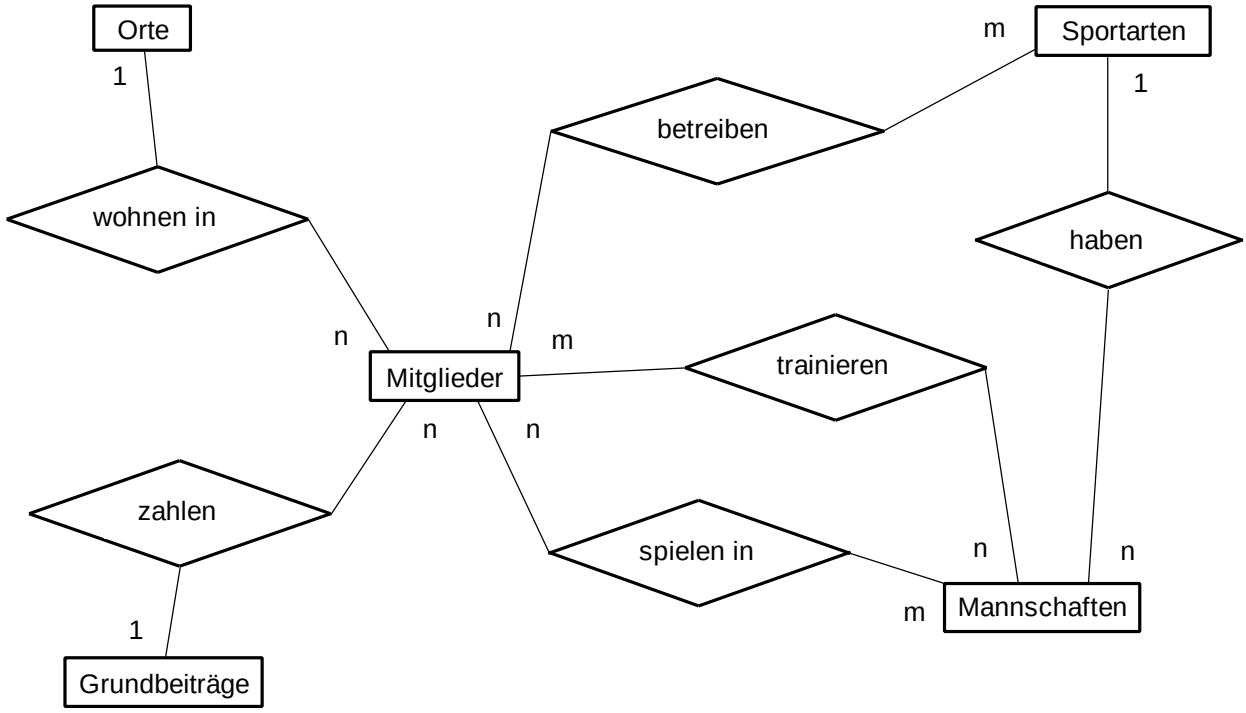


Bild 4-16: E-R-Diagramm der Datenbank *Verein6*

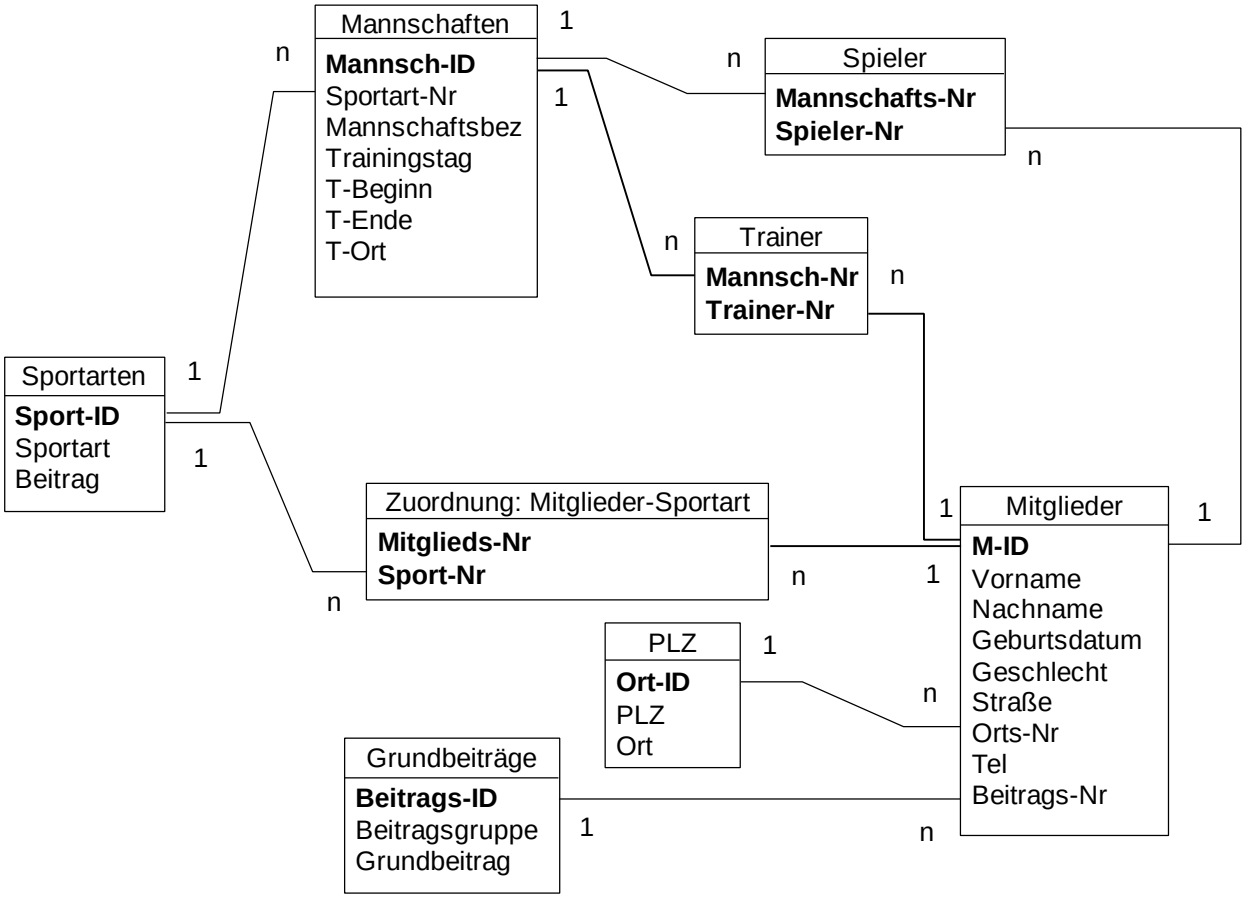


Bild 4-17: Relationales Modell der Datenbank *Verein6*

4.4 Abfragen

4.4.1 Grundlagen

Es gibt **drei grundlegende Tabellenabfrageoperationen:**

- Das Ausblenden von Spalten (**Projektion**)
- Das Auswählen von Zeilen (**Selektion, Restriktion**)
- Das Verknüpfen (Verbinden) von Tabellen (**Join**)

Bei der **Projektion** werden Spalten ausgeblendet, d. h. man reduziert mit der Projektion alle verfügbaren Datenfelder auf die gewünschten Datenfelder. Mit der **Selektion** kann man Zeilen aus einer Tabelle auswählen, die bestimmte Eigenschaften und Bedingungen erfüllen. Beim **Join** verbindet man in der Regel zwei bestehende Tabellen zu einer neuen Tabelle. Mit dem **Auto Join** (Self Join) lassen sich Verknüpfungen innerhalb einer einzigen Tabelle herstellen; eine Tabelle wird also mit sich selbst verknüpft.

Auswählen von Feldern: Man muss nicht alle Felder einer Tabelle in eine Abfrage aufnehmen. Man kann z. B. Namen und Vornamen von Mitgliedern ansehen, ohne dabei die Adressen oder andere Felder anzuzeigen. Aus Gründen des Datenschutzes will man evtl. die Zahl der Felder einschränken, die von Dritten bearbeitet werden können. Hierzu kann man eine Abfrage erstellen, die nur diejenigen Felder anzeigt, die man zur allgemeinen Ansicht freigeben möchte.

Einschränken von Datensätzen: Man kann Kriterien festlegen, denen die Datensätze genügen müssen, um in das Ergebnis der Abfrage aufgenommen zu werden. Beispielsweise sollen nur die Spieler einer bestimmten Mannschaft eingesehen werden können.

Sortieren von Datensätzen: Für viele Zwecke möchte man die Datensätze in einer bestimmten Reihenfolge anzeigen. So lassen sich z. B. die Mitgliederdatensätze alphabetisch geordnet nach Namen anzeigen.

Stellen von Fragen über Daten in mehreren Tabellen: Man kann eine Abfrage verwenden, um eine Frage über Daten aus mehr als einer Tabelle zu beantworten. Das Ergebnis wird dann auf einem einzigen Datenblatt angezeigt.

Berechnen von Gesamtbeträgen: Es lassen sich mit den Daten der Tabellen Berechnungen durchführen, z. B. das Alter eines Mitglieds.

Erstellen von Formularen und Berichten auf der Grundlage von Abfragen: Damit die richtigen Daten für ein Formular oder einen Bericht ausgewählt werden, ist eine Auswahlabfrage zu erstellen, auf welcher das Formular / der Bericht basieren. Bei jedem Öffnen des Formulars oder Drucken des Berichts ruft die Abfrage dann die aktuellsten Informationen aus den Tabellen ab.

Erstellen von Abfragen auf der Grundlage von Abfragen: Nach dem Entwerfen und Speichern einer Abfrage kann man eine weitere Abfrage erstellen, die Fragen über die Daten stellt, die durch die erste Abfrage ausgewählt wurden. Durch Erstellen einer Abfrage auf der Grundlage einer anderen Abfrage können z. B. an der ersten Abfrage kleinere Änderungen vorgenommen werden ohne gleichzeitig die ursprüngliche Abfrage zu verändern.

Erstellen von Diagrammen auf der Grundlage von Abfragen: Man kann den Formularen oder Berichten Diagramme hinzufügen, um die Daten aus den Tabellen grafisch darzustellen.

4.4.2 Verschiedene Arten von Abfragen

Das Datenbankmanagementsystem *MS-Access* stellt verschiedene Möglichkeiten für Abfragen zur Verfügung:

1. **Auswahlabfrage**
2. **Tabellenerstellungsabfrage**
3. **Aktualisierungsabfrage**
4. **Anfügeabfrage**
5. **Löschabfrage**
6. **Kreuztabellenabfrage**

Die am häufigsten benutzte Art der Abfrage ist die **Auswahlabfrage,** auf der die anderen Abfragearten im Prinzip aufbauen (ausführliche Behandlung in den Kapiteln 4.4.4 bis 4.4.6).

Die Abfragen 2. - 5. werden auch als **Aktionsabfragen** bezeichnet, da sie bestimmte Aktionen auslösen. **Vorsicht** beim wiederholten Ausführen, die vorgesehene Aktion wird jedes Mal neu ausgelöst (z. B. wird bei einer Tabellenerstellungsabfrage die betreffende Tabelle jedes Mal neu erzeugt, eine Preiserhöhung eines bestimmten Artikels (Aktualisierungsabfrage) wird bei jeder Ausführung der Abfrage erneut durchgeführt).

Eine **Tabellenerstellungsabfrage** erstellt aus den in der zugehörigen Abfrage ausgewählten Datenfeldern eine neue Tabelle.

Eine **Aktualisierungsabfrage** ändert Daten in einer ausgewählten Gruppe von Datensätzen.

Eine **Anfügeabfrage** fügt eine Gruppe von Datensätze an eine andere Tabelle an.

Eine **Löschabfrage** löscht Datensätze aus einer oder mehreren Tabellen.

Neben den bisher beschriebenen Auswahl- und Aktionsabfragen gibt es noch die Möglichkeit einer **Kreuztabellenabfrage**. Bei einer normalen Tabelle in einer Datenbank hat jede Zelle, also jedes einzelne Attribut eines Datensatzes, nur einen Titel, den sogenannten Spaltentitel. Auf diese Weise bildet jede Zeile der Tabelle einen Datensatz. In einer Kreuztabelle ist die Aufteilung etwas anders: es gibt *Zeilenüberschriften*, *Spaltenüberschriften* und *Tabelleneinträge*. Jeder Tabelleneintrag (Zelleneintrag) bezieht sich sowohl auf die Zeilen- als auch auf die Spaltenüberschrift. Die Daten innerhalb einer Zeile gehören jedoch nicht zu einem Datensatz, da sie alle unterschiedliche Ausprägungen einer bestimmten Eigenschaft darstellen.

Beispiel: Die Abteilungen des Sportvereins (Verein6) mit ihren Mitgliederzahlen pro Beitragsgruppe.

Sportart	Gesamt	Erwachsener	Familienbeitrag	Familienmitglied	Kind / Student
Aerobic	10	7			3
Basketball	22	7	1		14
Boxen	1				1
Fußball	108	9		6	93
Handball	22	1		1	20
Judo	25	1		1	23
Leichtathletik	40	1		1	38
Schwimmen	55	1	1	2	51
Turnen	28	10	1		17
Volleyball	20	5		1	14

4.4.3 Verknüpfungseigenschaften

Die **Verknüpfung von Tabellen** kann prinzipiell auf drei verschiedene Arten vorgenommen werden:

1. **Equi Join (Gleichheitsverknüpfung):** Die Inhalte der verknüpften Felder beider Tabellen sind gleich.

2a. **Outer Join (Inklusionsverknüpfung: Left-Join):** Das Ergebnis beinhaltet alle Datensätze aus der linken Tabelle und nur die Datensätze aus der rechten Tabelle, bei denen die Inhalte der verknüpften Felder beider Tabellen gleich sind.

2b. **Outer Join (Inklusionsverknüpfung: Right-Join):** Das Ergebnis beinhaltet alle Datensätze aus der rechten Tabelle und nur die Datensätze aus der linken Tabelle, bei denen die Inhalte der verknüpften Felder beider Tabellen gleich sind.

3 **Auto Join:** Stellt einen Selbstbezug einer Tabelle auf sich dar.

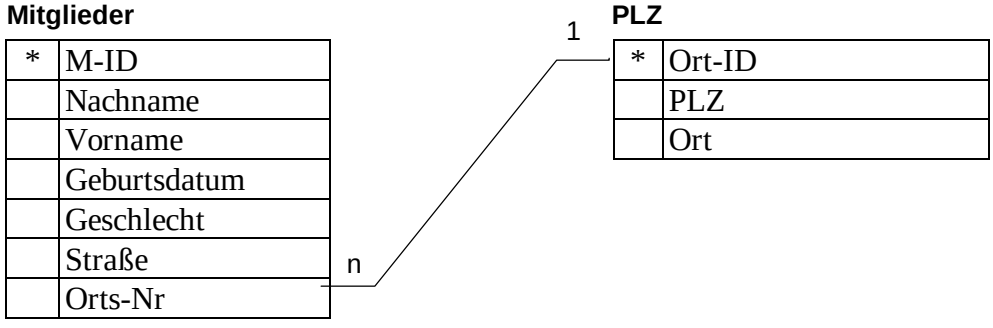
Die Standardverknüpfung zwischen Tabellen ist der Equi-Join. Für bestimmte Abfragen wird aber gelegentlich der Outer-Join 2a. bzw. 2b. benötigt (vgl. Beispiel 6 aus Kapitel 4.4.5 und Übungsaufgaben).

4.4.4 Elementare Anfragen an die Datenbank

In diesem Abschnitt wird gezeigt, wie aus einer Tabelle Spalten ausgewählt werden (Projektion), wie Zeilen ausgewählt werden (Selektion), wie die Ergebnisse sortiert und wie Funktionen benutzt werden können.

Als Grundoperation auf Tabellen haben wir die Projektion kennengelernt. Bei der Projektion werden Spalten aus einer Tabelle gestrichen. Übrig bleiben die gewünschten Spalten. Will man die Nachnamen und Vornamen aller Mitglieder ermitteln, so streicht man in der Tabelle *Mitglieder* alle Spalten bis auf die Spalten Nachname und Vorname und es ergibt sich die gewünschte Tabelle.

Alle Abfragen der Beispiele 1 - 6 werden aus den Tabellen *Mitglieder* und *PLZ* vorgenommen.



Beispiel 1: Ermitteln Sie die Namen und Vornamen aller Mitglieder (alphabetisch sortiert). Dabei werden in QBE die mit "x" gekennzeichneten Felder auch angezeigt.

Entwurfssprache:

Tabelle(n):	Mitglieder	
Felder:	Nachname, Vorname	**ç** Projektion
wobei (Auswahlkriterien)		**ç** Selektion
Sortiert nach:	Nachname, Vorname	**ç** Sortierung

Beispieltabelle (QBE): Tabelle *Mitglieder*

Feld:	Nachname	Vorname
Sortierung:	Aufsteigend	Aufsteigend
Anzeigen:	x	x
Kriterien:		
oder:		

Beispiel 2: Ermitteln Sie die Mitgliedsnummern, die Nachnamen und die Vornamen aller weiblichen Mitglieder (alphabetisch sortiert).

Entwurfssprache:

Tabelle(n):	Mitglieder	
Felder:	M-ID, Nachname, Vorname, Geschlecht	**ç** Projektion
wobei (Auswahlkriterien)	Geschlecht = „w“	**ç** Selektion
Sortiert nach:	Nachname	**ç** Sortierung

Beispieltabelle (QBE): Tabelle *Mitglieder*

Feld:	M-ID	Nachname	Vorname	Geschlecht
Sortierung:		Aufsteigend		
Anzeigen:	x	x	x	x
Kriterien:				”w“
oder:				

Beispiel 3: Ermitteln Sie die Mitgliedsnummern, die Nachnamen, Vornamen und das Geburtsdatum aller Mitglieder, deren Familienname mit einem „B“ beginnt (alphabetisch sortiert).

Entwurfssprache:

Tabelle(n):	Mitglieder	
Felder:	alle ohne Geschlecht	**ç** Projektion
wobei (Auswahlkriterien)	der Anfangsbuchstabe des Familiennamens „B“ ist.	**ç** Selektion
Sortiert nach:	Nachname, Vorname	**ç** Sortierung

Beispieltabelle (QBE): Tabelle *Mitglieder*

Feld:	M-ID	Nachname	Vorname	Geburtsdatum
Sortierung:		Aufsteigend	Aufsteigend	
Anzeigen:	x	x	x	x
Kriterien:		Wie ”B*“		
oder:				

Beispiel 4: Ermitteln Sie die Mitgliedsnummern, die Nachnamen, Vornamen, das Geburtsdatum und das Alter aller Mitglieder (nach Alter sortiert).

Entwurfssprache:

Tabelle(n):	Mitglieder	
Felder:	M-ID, Nachname, Vorname, Geburtsdatum	**ç** Projektion
wobei (Auswahlkriterien)		**ç** Selektion
Sortiert nach:	Alter	**ç** Sortierung

Beispieltabelle (QBE): Tabelle *Mitarbeiter*

Feld:	M-ID	Nachname	Vorname	Alter: Int((Datum()-[Geburtsdatum])/365,25)	Geburtsdatum
Sortierung:				absteigend	
Anzeigen:	x	x	x	x	x
Kriterien:					

Die Funktion ***Int*** nimmt den ganzzahligen Anteil eines Ergebnisses. Das Format der Spalte *Alter* ist: **Festkommazahl mit 0 Dezimalstellen**.

Weitere Beispiele siehe Übungsaufgaben bzw. praktischer Teil (Kapitel 4.4.5 bzw. Kapitel 5.1.4)

4.4.5 Komplexe Anfragen an die Datenbank

In diesem Abschnitt lernen wir die "teuerste" (zeitaufwendigste) Operation in einem relationalen Datenbanksystem kennen: Die Verbindung von Tabellen zu einer neuen Tabelle, den **Join**.

Beispiel 5: Ermitteln Sie von jedem Mitglied die Mitgliedsnummer, den Nach- und Vornamen, das Geburtsdatum, die Straße sowie PLZ und Ort.

Wie schon aus den zu ermittelnden Merkmalen hervorgeht, sind zur Lösung dieses Problems die beiden Tabellen *Mitglieder* und *PLZ* notwendig.

Gibt man die Beispieltabelle aus, ohne vorher bestimmte Merkmale der beiden Ausgangstabellen *Mitglieder* und *PLZ* miteinander zu verknüpfen, so werden alle möglichen Kombinationen zwischen den beiden Tabellen *Mitglieder* und *PLZ* gebildet. Man erhält in unserem Fall (bei 200 Mitgliedern und 29 Orten) 29 x 200 = 5800 Datensätze der Beispieltabelle, anstatt der erwarteten 200 Datensätze für die 200 Mitglieder. Die Verknüpfung der beiden Beispieltabellen erfolgt über die Attribute *Ort-ID* und *Orts-Nr* (1:n-Beziehung).

Entwurfssprache:

Tabelle(n):	Mitglieder, PLZ Mitglieder.Orts-Nr = PLZ.Ort-ID	**ç** Join
Felder:	M-ID, Nachname, Vorname, Geburtsdatum, Straße, PLZ, Ort	**ç** Projektion
wobei (Auswahlkriterien)		**ç** Selektion
Sortiert nach:	Ort	**ç** Sortierung

Beispieltabelle (QBE): Tabellen: *Mitglieder, PLZ*

Feld:	M-ID	Nachname	Vorname	Geburtsdatum	Straße	PLZ	Ort
Sortierung:							aufsteigend
Anzeigen:	x	x	x	x	x	x	x
Kriterien:							
oder:							

In SQL-Notation sieht die Abfrage wie folgt aus:

```
SELECT   Mitglieder.[M-ID], Mitglieder.Vorname, Mitglieder.Nachname, Mitglie-
         der.Geburtsdatum, Mitglieder.Geschlecht, Mitglieder.Straße, PLZ.PLZ, PLZ.Ort
FROM     PLZ INNER JOIN Mitglieder ON PLZ.[Ort-ID] = Mitglieder.[Orts-Nr]
ORDER    BY PLZ.Ort, Mitglieder.Nachname, Mitglieder.Vorname;
```

Lässt man die INNER JOIN-Bedingung weg, so wird auch in diesem Fall das kartesische Produkt der beiden Tabellen gebildet.

Beispiel 6: Ermitteln Sie alle nichtaktiven Mitglieder (die momentan keiner Sportart zugeordnet sind). Geben Sie für jedes dieser Mitglieder die M-ID, den Nachnamen, den Vornamen, Geburtsdatum, Geschlecht, Straße, PLZ und Ort an.

Entwurfssprache:

Tabelle(n):	Mitglieder, PLZ, Zuordnung: M-Nr-Sport-Nr PLZ.Ort-ID = Mitglieder.Orts-Nr Mitglieder.M-ID = Zuordnung....M-Nr	ç Join ç Outer-Join
Felder:	M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, PLZ, Ort	ç Projektion
wobei (Auswahlkriterien)	Zuordnung: Mitglieder-Sportart. M-Nr besitzt keinen Eintrag	ç Selektion
Sortiert nach:	Nachname	ç Sortierung

Beispieltabelle (QBE): Tabellen: *Mitglieder, PLZ, Zuordnung: Mitglieder-Sportart*

Feld:	M-ID	Nachname	Vorname	Geschlecht	Geb-dat	PLZ	Ort	M-Nr
Sortierung:								
Anzeigen:	x	x	x	x	x	x	x	
Kriterien:								Ist Null
oder:								

In SQL-Notation sieht die Abfrage wie folgt aus:

```
SELECT   Mitglieder.[M-ID], Mitglieder.Vorname, Mitglieder.Nachname, Mitglie-
         der.Geburtsdatum, Mitglieder.Geschlecht, Mitglieder.Straße, PLZ.PLZ, PLZ.Ort
FROM     (PLZ INNER JOIN Mitglieder ON PLZ.[Ort-ID] = Mitglieder.[Orts-Nr]) LEFT JOIN [Zu-
         ordnung: Mitglieder-Sportart] ON Mitglieder.[M-ID] = [Zuordnung: Mitglieder-
         Sportart].[M-Nr]
WHERE    (([Zuordnung: Mitglieder-Sportart].[M-Nr] Is Null));
```

4.4.6 Übungsaufgaben

Für die Datenbank ***Verein3*** bzw. ***Verein4*** ergeben sich u. a. folgende mögliche Abfragen:

1. Geben Sie alle Mitglieder nach Geburtsdatum sortiert aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum.

1.1 Geben Sie alle Mitglieder alphabetisch sortiert aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, PLZ, Ort, Tel.

1.2 Geben Sie alle Mitglieder aus, deren Nachname mit „A“ beginnt. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum.

2. Geben Sie alle männlichen Mitglieder aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname.

2.1 Geben Sie alle weiblichen Mitglieder aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname.

3. Geben Sie alle Mitglieder nach Orten und Namen sortiert aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Straße, PLZ, Ort, Tel.

4. Geben Sie alle Mitglieder nach Alter sortiert aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Alter, Geburtsdatum.

5. Geben Sie alle Mitglieder, die vor dem 01-01-1980 geboren sind nach (Geburtsdatum sortiert) aus. Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum.

6. Geben Sie alle Mitglieder aus, die mindestens 20 Jahre alt sind (nach Alter absteigend sortiert). Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Alter, Geburtsdatum

7. Geben Sie alle Mitglieder aus, die im August geboren sind (nach Geburtstag aufsteigend und nach Alter absteigend sortiert). Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum, Geburtstag, Alter.

7.1 Geben Sie eine Geburtstagsliste aller Mitglieder aus (nach Monat und Tag aufsteigend und nach Alter absteigend sortiert). Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum, Monat, Tag, Alter, Monatsname.

7.2 Geben Sie aller Mitglieder aus, die zwischen einem Anfangsdatum und einem Enddatum geboren sind (nach Geburtsdatum sortiert).
Mit Anzeige folgender Datenfelder: Vorname, Nachname, Geburtsdatum.

7.3 Geben Sie aller Mitglieder aus, die heute Geburtstag haben.
Mit Anzeige folgender Datenfelder: Vorname, Nachname, Geburtsdatum.

7.4 Geben Sie alle männlichen Fußballer aus, die zwischen 16 und 18 Jahre alt sind (nach Alter sortiert). Mit Anzeige folgender Datenfelder: M-ID, Vorname, Nachname, Geburtsdatum, Alter.

8. Geben Sie alle Sportarten und ihre Mitglieder aus. Die Mitglieder sollen innerhalb der Sportarten alphabetisch sortiert sein (nach Name bzw. Vorname). Die Sportarten sollen aufsteigend sortiert sein.
Mit Anzeige folgender Datenfelder in der angegebenen Reihenfolge:
M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, PLZ, Ort, Tel, Sportart, Beitrag.

8.1 Wie Aufgabe 8, nur Sportart als Parameter.

9. Geben Sie alle Orte mit ihren Mitgliederzahlen aus; sortiert nach Mitgliederzahlen (absteigend). Mit Anzeige folgender Datenfelder: Ort-ID, PLZ, Ort, Mitgliederzahl.

10. Geben Sie alle Mitglieder an, die nicht aktiv sind (in keiner Sportart registriert). Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum.

10.1 Geben Sie alle Sportarten an, in denen keine Mitglieder registriert sind.
Mit Anzeige folgender Datenfelder: Sport-ID, Sportart, Beitrag.

10.2 Geben Sie alle Mitglieder an, auch die nicht aktiv sind. Mit Anzeige folgender Datenfelder:
M-ID, Nachname, Vorname, Geburtsdatum.

10.3 Geben Sie alle Sportarten an, auch die, in denen keine Mitglieder registriert sind.
Mit Anzeige folgender Datenfelder: Sport-ID, Sportart, Beitrag.

10.4 Geben Sie alle Orte an, in denen keine Mitglieder registriert sind.
Mit Anzeige folgender Datenfelder: Ort-ID, PLZ, Ort.

11. Geben Sie alle Sportarten sowie deren Gesamtmitgliederzahl an.
Mit Anzeige folgender Datenfelder: Sportart, Mitgliederzahl.

11.1 Geben Sie die Sportarten sowie deren Mitgliederzahl nach Geschlecht spezifiziert an.
Mit Anzeige folgender Datenfelder: Sportart, Gesamt, m, w. Lösung als Kreuztabelle

11.2 Wie 11.1, aber alle Sportarten, d. h. auch diejenigen ohne Mitglieder.

11.3 Wie 11.1, aber Lösung mit Unterabfragen.

11.4 Geben Sie die Sportarten sowie deren Mitgliederzahl nach Orten spezifiziert an. Mit Anzeige folgender Datenfelder: Ort, Sportarten, zugehörige Mitgliederzahlen. Lösung als Kreuztabelle

12. Geben Sie alle Sportarten sowie deren Mitgliederzahl und das Durchschnittsalter (1 Dezimalstelle genau) der jeweiligen Abteilung an.

13. Geben Sie alle Mitglieder aus, die im selben Ort wohnen wie ein bestimmtes Mitglied (Name bekannt, z. B. *Meister*). Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Straße, PLZ, Ort, Tel.

13.1 Wie 13 nur der Name als Parameter.

14. Geben Sie alle Mitglieder aus, die in der gleichen Abteilung sind wie ein bestimmtes Mitglied (Name bekannt, z. B. *Fischer*).
Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Geburtsdatum, Sportart.

14.1 Wie 14 nur der Name als Parameter.

14.2 Erstellen Sie eine Abfrage bei der diejenigen Mitglieder ausgegeben werden, deren Nachnamen mit der eingegeben Zeichenfolge übereinstimmt.
Mit Anzeige folgender Datenfelder: Vorname, Nachname, Straße, PLZ, Ort, Tel.

15. Geben Sie die Mitglieder aus, deren Vornamen mit „A“ und deren Nachnamen mit einem Buchstaben nach „M“ beginnt.
Mit Anzeige folgender Datenfelder: Vorname, Nachname.

15.1 Geben Sie die Mitglieder aus, deren Nachnamen mit „M“ beginnt und mit „er“ endet.
Mit Anzeige folgender Datenfelder: Vorname, Nachname.

16. Geben Sie die Mitglieder aus, deren Vornamen mit „A“ oder deren Nachnamen mit einem Buchstaben nach „S“ beginnt.
Mit Anzeige folgender Datenfelder: Vorname, Nachname.

16.1 Geben Sie die Mitglieder aus, deren Nachnamen mit „A“ oder mit „K“ beginnt.
Mit Anzeige folgender Datenfelder: Vorname, Nachname.

17. Geben Sie alle Mitglieder aus, 1980 geboren sind (aufsteigend nach Geburtsdatum sortiert).
Mit Anzeige folgender Datenfelder: Vorname, Nachname, Geburtsdatum.

18. Berechnen Sie einen Zuschuss für alle Mitglieder: Alle, die nach dem 1.1.1980 geboren sind, zahlen 15 €, die anderen zahlen 30 € (aufsteigend nach Geburtsdatum sortiert).
Mit Anzeige folgender Datenfelder: Vorname, Nachname, Geburtsdatum, Zuschuss.

18.1 Geben Sie die Mitglieder aus und berechnen Sie deren Anrede (Frau bzw. Herrn) mit Hilfe einer Funktion, die in einem Modul als Basic-Kode definiert ist.
Mit Anzeige folgender Datenfelder: Anrede, Vorname, Nachname, Geschlecht.

18.2 Berechnen Sie das Alter sämtlicher Mitglieder mit Hilfe einer Funktion, die in einem Modul als Basic-Kode definiert ist (Hinweis: Funktion „DateSerial“).
Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Alter, Geburtsdatum

19. Berechnen Sie für jeden Teilnehmer aus dem Feld „Tel“ die Position des Trennzeichens „/“ zwischen der Vorwahl und der eigentlichen Telefonnummer. Mit Anzeige folgender Datenfelder:
M-ID, Vorname, Nachname, Tel-Nr, Position.

19.1 Berechnen Sie für jeden Ort die zugehörige Ortsvorwahl aus der Telefonnr. (nach Vorwahl sortiert).
Mit Anzeige folgender Datenfelder: PLZ, Ort, Vorwahl.

19.2 Berechnen Sie für jeden Teilnehmer aus der Telefonnr die zugehörige Ortsvorwahl sowie die eigentliche Telefonnummer (alphabetisch sortiert). Mit Anzeige folgender Datenfelder:
M-ID, Vorname, Nachname, Straße, PLZ, Ort, Tel, Vorwahl, Tel-Nr.

20. Erstellen Sie eine Abfrage bei der aus dem Feld Strasse der Straßenname sowie die Hausnummer bestimmt und in den Feldern Straßenname bzw. Hausnr getrennt ausgegeben werden.
Mit Anzeige folgender Datenfelder: Straße, Straßenname, Hausnr.

20.1 Geben Sie die Mitglieder aus, die in der gleichen Straße wohnen, wie ein bestimmtes Mitglied (Parameter). Einmal nur Straße identisch, einmal auch Ort identisch.
Mit Anzeige folgender Datenfelder: M-ID, Nachname, Vorname, Straße, PLZ, Ort.

Die Lösung der Abfragen befindet sich in der Datenbank Verein4_97.mdb; dabei beginnt z B. die Bezeichnung der zu Aufgabe 1.1 zugehörigen Abfrage mit Ü01_1....usw.

Für die Datenbank ***Verein5*** bzw. ***Verein6*** ergeben sich zusätzlich u. a. folgende mögliche Abfragen:

1. Geben Sie alle Trainer und die zugehörigen Fußballmannschaften (nach Mannschaften sortiert) aus. Dabei sollen folgende Datenfelder gezeigt werden: Sportart, Mannschaftsbez, Nachname, Vorname, Straße, PLZ, Ort, Tel, Trainingstag, T-Beginn, T-Ende, T-Ort.

2. Geben Sie alle Spieler und ihre zugehörigen Fußball-Mannschaften (nach Mannschaften sortiert, innerhalb der Mannschaften alphabetisch sortiert) aus. Dabei sollen folgende Datenfelder gezeigt werden: M-Nr, Sportart, Mannschaftsbez, Nachname, Vorname, Tel.

3. Geben Sie alle Mitglieder an, die nicht in einer Mannschaft spielen. Dabei sollen folgende Datenfelder gezeigt werden: M-Nr, Nachname, Vorname, Geburtsdatum.

4. Geben Sie alle Sportarten sowie deren Gesamtmitgliederzahl und die Zahl der Mitglieder pro Beitragsgruppe in jeder Sportart an.

5. Geben Sie Einnahmen der Grundbeiträge nach Beitragsgruppen aus. Dabei sollen folgende Datenfelder gezeigt werden: Beitragsgruppe, Mitgliederzahl/Beitragsgruppe, Grundbeitrag/Beitragsgruppe, Gesamtbeitrag/Beitragsgruppe.

6. Geben Sie Einnahmen der Abteilungsbeiträge nach Sportarten aus. Dabei sollen folgende Datenfelder gezeigt werden:
Sportart, Mitgliederzahl/Sportart, Abteilungsbeitrag/Sportart, Gesamtbeitrag/Sportart.

5 Realisierung des logischen Modells mit MS-Access

Aufgabe

Für einen Sportverein sollen die wichtigsten Daten der Mitglieder in einer Tabelle erfasst werden.

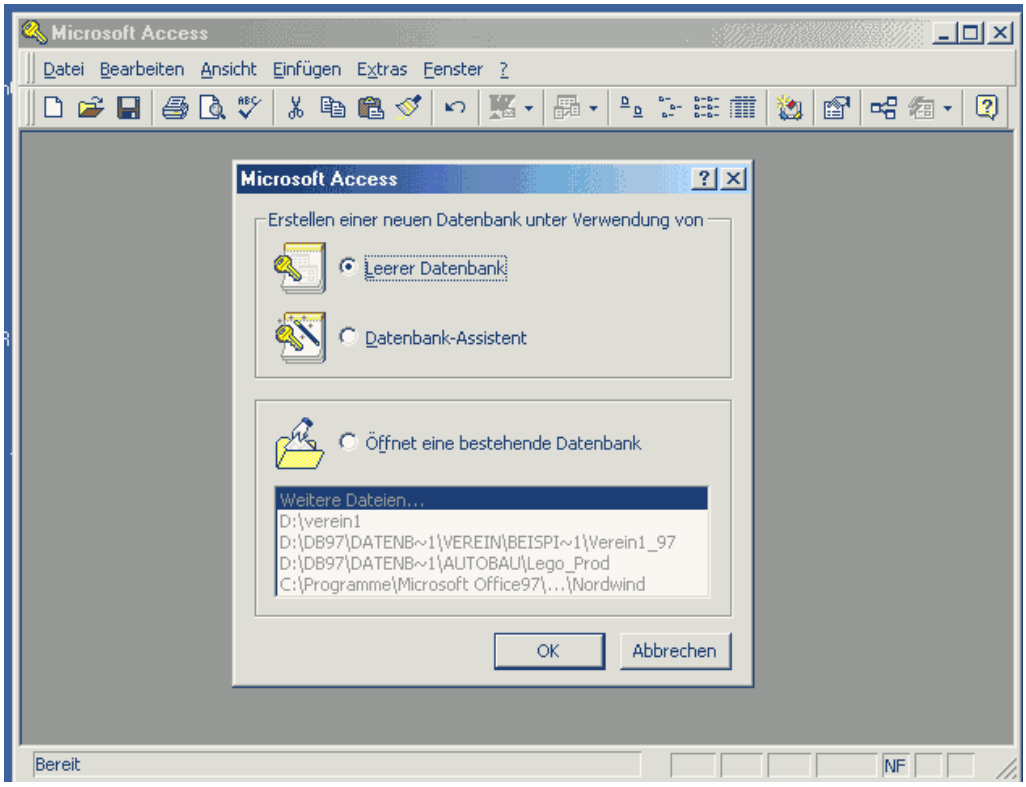
Da dies ein sehr einfaches Beispiel ist, wird hier zunächst auf die Erstellung des Entity-Relationship-Diagramms sowie des logischen Modells verzichtet.

Informationsbedarf des Sportvereins

M-Nr	Vorname	Nachname	m/w	Straße	PLZ	Ort	Sportart	Beitrag
1	Ulrich	Becker	m	Maxweg 14	85408	Gammelsdorf	S	80,00 €
2	Ulrich	Becker	m	Maxweg 41	85408	Gammelsdorf	H	55,00 €
3	Julia	Berger	w	Fischweg 22	85395	Attenkirchen	H	55,00 €
4	Manuela	Fiedmann	w	Bahnhofstr 23	85406	Zolling	L	55,00 €
5	Otto	Fischer	m	Karlweg 12	85375	Neufarn	F	75,00 €
6	Otto	Fischer	M	Karlweg 12	85375	Neufarn	L	55,00 €
7	Georg	Frohmann	m	Meierweg 99	85408	Daberg	F	75,00 €
8	Georg	Frohmann	m	Meierweg 99	85408	Daberg	S	80,00 €
9	Hans	Huber	m	Postweg 12	85368	Brugschlag	S	80,00 €
10	Hans	Huber	m	Postweg 12	85368	Brugschlag	F	75,00 €

5.1 Anlegen einer neuen Datenbank

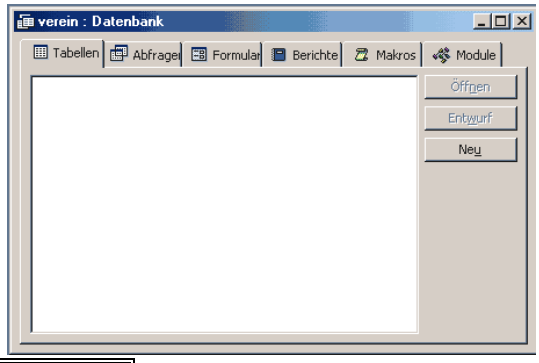
Access ist zu starten und das Icon *Neue Datenbank* anzuklicken.



Nun wird ein Name (hier: A:Verein1) vergeben und mit *OK* bestätigt. Im Laufwerk A: muss sich eine Diskette befinden! Alternativ kann die Datei auch auf einem eigenen Unterverzeichnis auf der Festplatte D: oder auf dem Server abgelegt werden. Es muss dann aber immer der gleiche Rechner benutzt werden!

Hinweis: Die Festlegung des Dateinamens der Datenbank sowie des Pfades kann von Access aus nur beim Erstellen der Datenbank vorgenommen werden, da alle Objekte der Datenbank unter diesem Dateinamen abgelegt sind. Spätere Änderungen des Dateinamens können nicht von Access aus durchgeführt werden, da die Option *Datei speichern unter*, wie sie aus anderen Programmen bekannt ist, in Access für die gesamte Datenbank nicht vorhanden ist. Dies gilt nur für einzelne Objekte wie Tabellen, Abfragen, Formulare, Berichte etc.. Die Ursache liegt darin begründet, dass diese Datenbankobjekte alle unter einem Namen, dem Dateinamen der Datenbank abgespeichert sind. Änderungen des Dateinamens der Datenbank können nur vom Explorer / Dateimanager aus vorgenommen werden.

5.1.1 Erstellen und Ändern von Tabellen im Modell *Verein1*



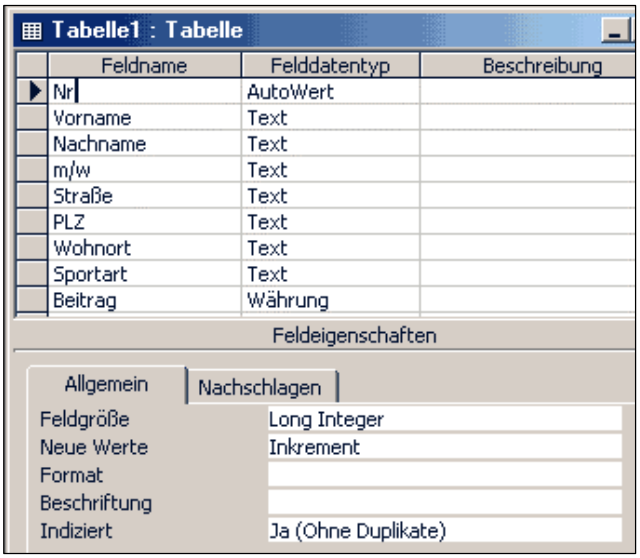
Access erstellt jetzt die Datenbank und es wird das nebenstehende Programmfenster angezeigt:

Da der Informationsbedarf des Sportvereins bekannt ist, kann mit der Eintragung in eine Access-Tabelle begonnen werden. Dazu wird im Datenbankfenster das Registerblatt *Tabellen* ausgewählt und dann auf die Schaltfläche *Neu* geklickt. Aus der nun erscheinenden Auswahl ist *Entwurfsansicht* zu wählen.

Danach sind folgende Eintragungen vorzunehmen:

Feldname	Felddatentyp	Feldgröße/Format	Indiziert
Nr	AutoWert	Long Integer	Ja (ohne Duplikate)
Vorname	Text	20	Ja (Duplikate möglich)
Nachname	Text	20	Ja (Duplikate möglich)
m/w	Text	1	Ja (Duplikate möglich)
Straße	Text	20	Ja (Duplikate möglich)
PLZ	Text	5	Ja (Duplikate möglich)
Wohnort	Text	20	Ja (Duplikate möglich)
Sportart	Text	1	Ja (Duplikate möglich)
Beitrag	Währung	Währung	nein

Entwurfsansicht der Tabelle:



kurze Erklärung zum Primärschlüssel:

Der Primärschlüssel ist eine Hilfsfunktion, mit der jeder Eintrag in der Tabelle eindeutig identifiziert werden kann (vgl. Kapitel 3.2.3)

Felddatentyp und -größe

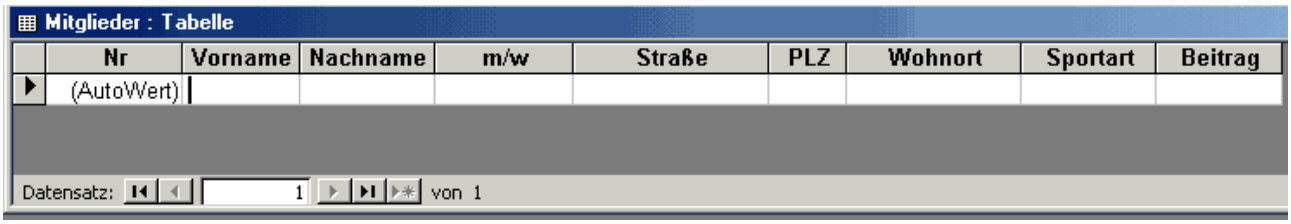
Da alle Eintragungen einer Tabellenspalte dasselbe Format besitzen, wird verhindert, dass z. B. in Zahlenfelder Texte eingegeben werden, mit denen später nicht gerechnet werden kann. Access führt bereits bei der Eingabe **„Plausibilitätsprüfungen"** durch und weist Fehleingaben zurück. Der ausgewählte **Datentyp** legt fest, aus welchem **Wertebereich** („Domäne") Daten in die Datenfelder eingetragen werden:

Datentyp	**Feldgröße (*Speicherbedarf*)**	**Beispiel, Wertebereiche**
Text	bis 255 Zeichen (*je Zeichen 1 Byte*)	„Huber"; „Postweg 12"
Memo	Längere Textfelder (*bis 64000 Byte*)	Besondere Merkmale, Hinweise
Zahl	Byte (*1 Byte*) Integer (*2 Byte; 0 Dezimalstellen*) Long Integer (*4 Byte; 0 Dez.stellen*) Single (*4 Byte; max. 7 Dez.stellen*), Double (*8 Byte; max. 15 Dez.stellen*)	0 ...255 -32768 ...32767 -2,147 Mrd. ... 2,147 Mrd. $-3{,}4\cdot10^{38}$ ...$3{,}4\cdot10^{38}$ $-1{,}8\cdot10^{308}$ ... $1{,}8\cdot10^{308}$
Datum/Zeit	Termine und Zeiten (*8 Byte*)	1.1.100 ...31.12.9999
Währung	Geldbeträge mit Dez.stellen (*8 Byte*)	180,00 €
Zähler	Automatische Nummerierung (*4 Byte*)	1,2,3....(Long Integer!)
Ja/Nein	Wahrheitswerte Wahr/Falsch (*1 Byte*)	männlich: Ja/Nein
OLE	OLE-Objekte (*128 MB Binärdaten*)	Bilder, Tabellen, Klänge

In der linken oberen Ecke befindet sich die Schaltfläche, mit der man in die Entwurfansicht bzw. in die Datenblattansicht umschalten kann.

Schaltet man jetzt in die Datenblattansicht, kann nach Angabe des Titels (**Mitglieder**) die Eingabe in

die Tabelle erfolgen. Der Wert für die Nr. des Mitglieds wird dabei automatisch von Access weitergezählt und muss nicht eingegeben werden. Eingegebene Datensätze werden automatisch gespeichert, wenn man mit dem Cursor die entsprechende Zeile verlässt.



Bewegung in der Tabelle:

In der Tabelle kann man mit der Maus, der Return-Taste, der Tab-Taste und den Pfeil-Tasten die Felder wechseln und dann Eintragungen vornehmen.

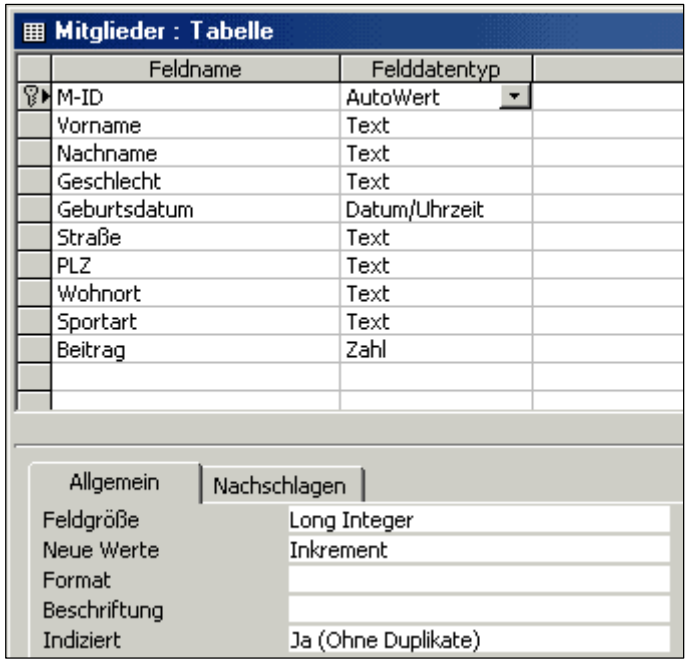
Nach Beendigung der Eingabe wird die Tabelle mit *Datei - Schließen* geschlossen. Das Datenbankfenster zeigt jetzt die erstellte Tabelle:

Diese Tabelle (Relation) ist nun die Datenbasis, die auch von anderen Anwendungen (z.B. in einer Textverarbeitung zur Erstellung eines Serienbriefes oder in einer Kalkulation zur Auswertung und grafischen Darstellung) genutzt werden kann.

Aufgabe: Definieren Sie für das konkrete Beispiel die Begriffe **Entitätsmenge** und **Entität, Attribut** und **Attributwerte** sowie **Wertebereiche der Attribute**

5.1.2 Änderungen in einer Tabelle

Damit die Tabelle **Mitglieder** für das Modell *Verein3* angepasst werden kann, müssen einige Änderungen vorgenommen werden. Zu diesem Zweck ist die Tabelle in der Entwurfsansicht zu laden.



Überschriften (Feldnamen) der Spalten ändern: Feldname “Nr“ ist in “M-ID“,und “m/w“ in “Geschlecht“ zu ändern.

Neue Zeile einfügen: Die Zeile (Straße) markieren, vor der eine Zeile eingefügt wird. Anschließend aus dem Menü *Einfügen* die Option *Zeile* auswählen. Es wird jetzt eine zusätzliche Zeile eingefügt. Als Feldname ist Geburtsdatum einzugeben (Datentyp: Datum/Zeit).

Die nebenstehende Abbildung zeigt alle durchgeführten Änderungen in der Entwurfsansicht.

Schaltet man jetzt in die Datenblattansicht, so ist die auftretende Abfrage „*Möchten Sie die Tabelle jetzt speichern“* mit „*Ja“* zu beantworten. In der Datenblattansicht sind dann die vorgenommenen Änderungen sichtbar.

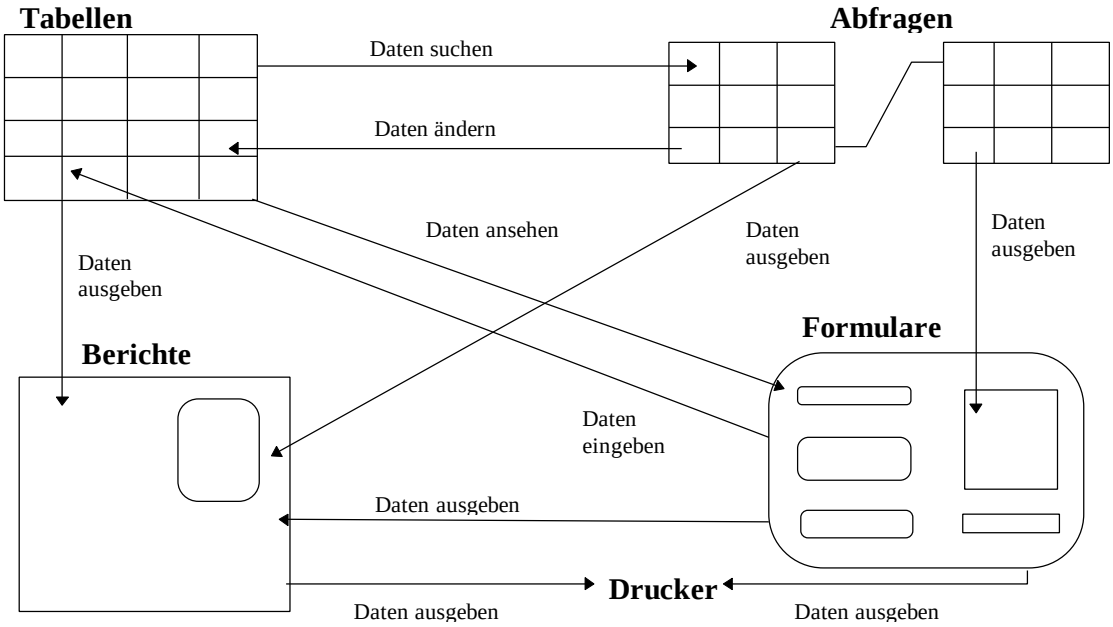
In der Spalte Geburtsdatum werden die entsprechenden Geburtstage eingefügt.

5.1.3 Zusammenspiel der Datenbankkomponenten von Access

Mit Hilfe des DBMS Access lassen sich verschiedene benutzerfreundliche Anwendungen erstellen, die auf die Datenbasis zugreifen:

Tabellen (Datensammlungen zu einem bestimmten Thema)
Abfragen (z.B. alle weiblichen Handballspielerinnen)
Formulare (z.B. Bildschirmmasken zur Datenpflege)
Berichte (z.B. Ausdruck der Quittung für den Jahresbeitrag)
Makros (werden einmal aufgezeichnet und erledigen Standardaufgaben automatisch)
Module (in Access-Basic erstellte Programme zur individuellen Datennutzung)

Zusammenwirken der einzelnen Komponenten von Access:



5.1.4 Einfache Abfragen im Modell *Verein1*

Die Basis für die Auswertung der Daten stellen Abfragen dar. Die Grundlagen hierfür wurden ausführlich im Kapitel 4.4 gelegt. An dieser Stelle soll nur noch kurz in den Umgang mit der Abfragekomponente von *Access* eingeführt werden.

Für die Abfrage von Daten wird wie folgt vorgegangen: Im Datenbankfenster die Registerkarte *Abfragen* auswählen und dann die Schaltfläche *Neu* klicken. Nach *Neue Abfrage* kann die Tabelle **Adressen** durch *Hinzufügen* gewählt und das Dialogfeld durch *Schließen* geschlossen werden.

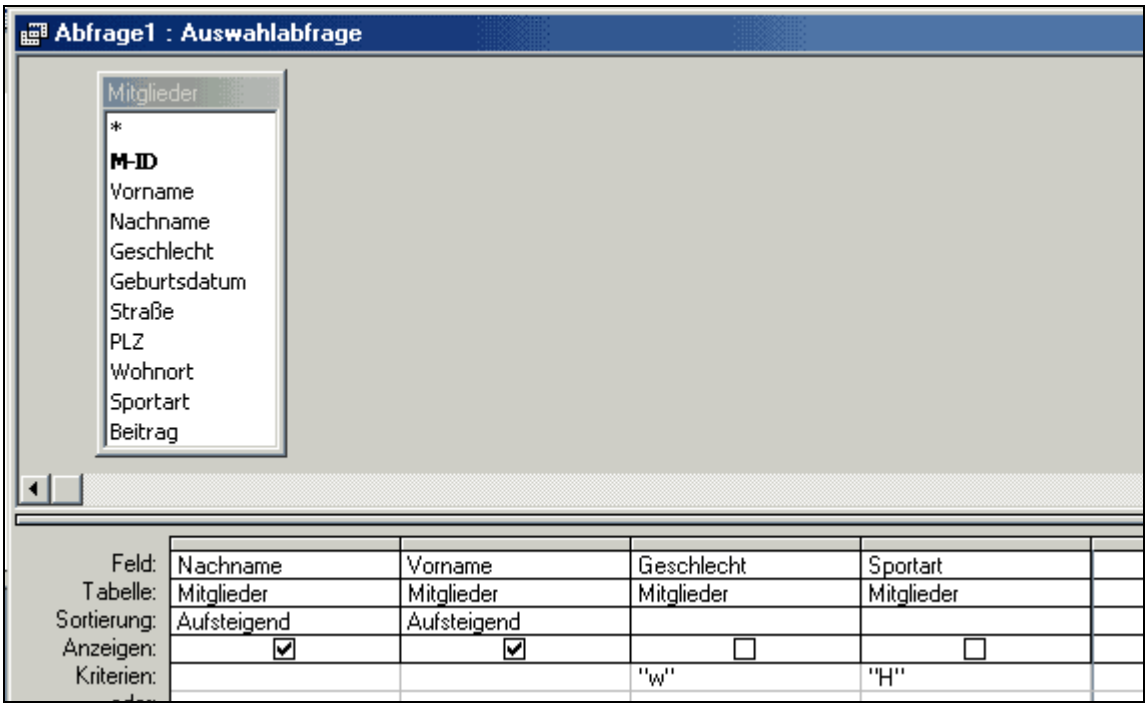
1. Abfrage

Von allen Handballspielerinnen soll eine alphabetisch sortierte Liste mit Nachnamen und Vornamen erstellt werden; das Geschlecht und die Sportart sollen nicht angezeigt werden.

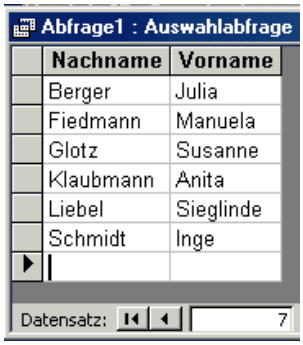
Vorgehensweise:

Im Fenster **Adressen**:

1. Doppelklick auf die Felder *Nachname; Vorname; Geschlecht; Sportart* (Transport der Felder in den Entwurfsbereich)
2. Im Feld *Nachname* in die *Zeile Sortierung* klicken, dann den *sichtbaren Pfeil* anklicken und *Aufsteigend* markieren.
3. Im Feld Vorname ist analog zu verfahren.
4. Im Feld *Geschlecht* ist in die Zeile Kriterien zu klicken und der Buchstabe w einzugeben.
5. Im Feld Sportart ist für Kriterium H einzugeben
6. Da die Felder *Geschlecht* und *Sportart* nicht angezeigt werden sollen, ist die Anzeige durch Mausklick im Quadrat zu deaktivieren.



7. Durch Anklicken des *Ausrufezeichens*  wird die Abfrage ausgeführt. Die nebenstehende Abbildung zeigt das Ergebnis der Abfrage.



Durch Anklicken von *Datei - Speichern* kann die Abfrage gespeichert werden

Name: **Handballspieler (w)**

2. Abfrage

Von allen männlichen Handballspielern soll eine alphabetisch sortierte Liste mit Nachnamen, Vornamen und Sportart erstellt werden.

3. Abfrage

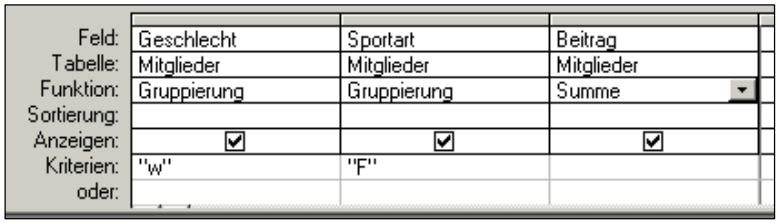
Von allen Fußballspielerinnen ist der insgesamt zu zahlende Beitrag zu ermitteln.

Dabei ist wie folgt zu verfahren:

Nach Auswahl der Felder *Geschlecht, Sportart, Beitrag* und Festlegen der Kriterien „w“ und „F“ sind mit dem Symbol  *Funktionen* aufzurufen. Access fügt die Zeile *Funktion* ein und setzt die Felder automatisch auf *Gruppierung*.

In der *Spalte Beiträge* ist das Wort *Gruppierung* und anschließend der *sichtbare Pfeil* anzuklicken und *Summe* auszuwählen.

3. Abfrage:



Ergebnis:

5.1.5 Erstellen und Anpassen eines einfachen Formulars im Modell *Verein1*

Die Eingabe und Pflege von Datensätzen in die Tabelle kann durch ein Eingabeformular wesentlich vereinfacht werden. An dieser Stelle soll ein erster Einblick in den Aufbau und die Arbeitsweise mit einem Formular gegeben werden (tiefergehende Informationen siehe auch Kapitel 5.3 und 5.5).

Im Datenbankfenster ist die Registerkarte *Formulare* auszuwählen und dann die Schaltfläche *Neu* anzuklikken. Nach Auswahl des *Formularassistenten* und der Tabelle **Adressen** werden mit  alle verfügbaren Felder übernommen. Die weitere Wahl ist *einspaltige Darstellung* und *farbig 2*. Als Titel geben wir **Mitgliederdaten** ein und *stellen fertig*.

In das links stehende Formular, das jetzt aufgebaut wird, können die Daten eingegeben werden. Die angezeigte Menüleiste ermöglicht das Blättern im Formular:

Erster, vorheriger, nächster, letzter, nächster freier **Datensatz**

Das Formular ist jetzt mit *Datei - Speichern - Speichern unter* **F Mitgliederdaten** zu sichern.

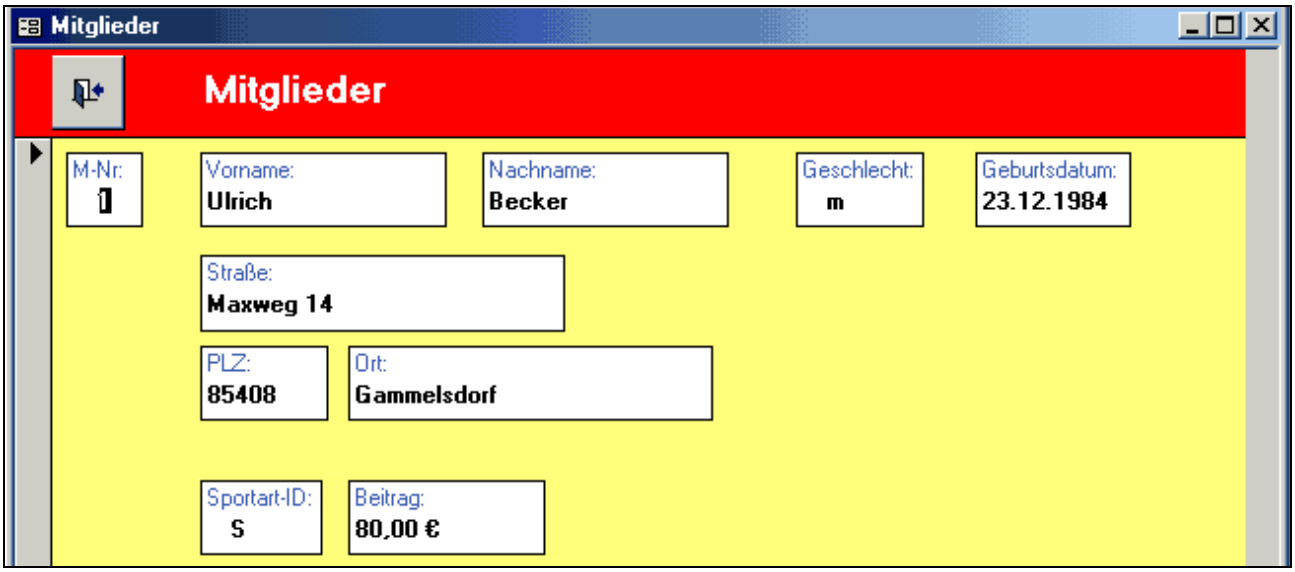
Für weitere Eingaben sowie Pflege der Datensätze kann das Eingabeformular über *Formular- F Mitgliederdaten* wieder aufgerufen werden.

5.1.6 Anpassen des Eingabeformulars zur Datenpflege

Wenn man das Formular zur Datenpflege geöffnet hat, so kann durch Anklicken von  in die Entwurfsansicht umgeschaltet werden. Hier kann dann durch Anklicken der einzelnen Beschriftungselemente und Eingabebereiche mit anschließender Verschiebung das Formular an die entsprechenden Bedürfnisse angepasst werden. **Hinweis:** Will man das Textfeld, das zugehörige Steuerelement und den entsprechenden Rahmen miteinander verschieben, so muss man diese Elemente gleichzeitig markieren und verschieben. Dies erreicht man am besten, indem man bei gedrückter linker Maustaste einen Rahmen um die entsprechenden Elemente zieht, so dass sie alle markiert sind; dies ist auch für mehrere Felder gleichzeitig möglich (z. B. Verschieben einer ganzen Zeile).

Der Aufruf der TOOLBOX - Schaltflächen durch  erleichtert die Neugestaltung des Formblattes wesentlich. Informationen zur Handhabung der einzelnen Schaltflächen erhält man nach ihrem Anklicken und anschließendem Drücken der Taste F1(Hilfetaste).

Abbildung des angepassten Eingabeformulars:

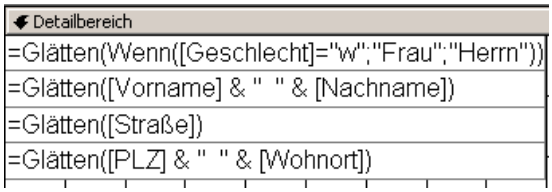


5.1.7 Erstellen eines einfachen Berichts im Modell *Verein1*

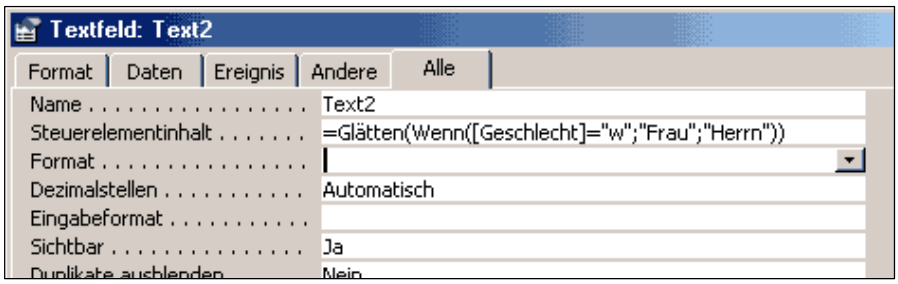
Mit einem Bericht besitzt man eine weitere Möglichkeit, Informationen aus der zugehörigen Datenbank abzurufen und anschaulich zu präsentieren. Ein Bericht unterscheidet sich von den beiden Möglichkeiten eines Formulars (Ausdruck in der Datenblattansicht und Ausdruck in der Formularansicht) durch mehr Flexibilität bei der Erstellung von Gestaltungsmustern und durch weitergehende Funktionen für zusammenfassende Auswertungen. An dieser Stelle soll ein erster Einblick in den Aufbau und die Möglichkeiten eines Berichts gegeben werden (tiefergehende Informationen siehe auch Kapitel 5.4).

Es soll nun ein Bericht für Adressetiketten erstellt werden, der die Mitgliederdaten in alphabetischer Reihenfolge wie unten angezeigt, ausgibt.

1. Es wird eine Abfrage *Mitgliederadressen (alphabetisch)* erstellt, die folgende Datenfelder enthält: *Vorname, Nachname, Geschlecht, Straße, PLZ, Wohnort*. Damit jedes Mitglied nur einfach angezeigt wird, wählt man die Gruppierungsfunktion.
2. Nun wird die Datenbankkomponente *Bericht* ausgewählt und ein neuer Bericht mit dem *Etiketten-Assistenten* auf der Abfrage *Mitgliederadressen (alphabetisch)* erstellt. Nach Auswahl der Etikettengröße und Schriftart sind die Felder zu bestimmen, die ausgegeben werden sollen. Zu beachten ist lediglich, dass in der ersten Zeile das Geschlecht eingegeben wird (Hinweis: daraus wird später mit Hilfe der *wenn - Funktion* die Anrede konstruiert).
3. Nun lässt sich die Sortierung festlegen und ein Name **(Etiketten Mitgliederadressen**) vergeben.
4. Nach *Fertigstellen* kann man den Bericht ansehen bzw. den Entwurf verändern.
5. Es soll nun noch das Geschlecht in die Anrede umgewandelt werden. Dies geschieht mit Hilfe der Funktion: = Wenn([Geschlecht]="w";"Frau";"Herrn"), die in der Entwurfsansicht eingegeben wird.



6. Weitere Feineinstellungen können nach Öffnen des Eigenschaftenmenüs durch Klicken auf  vorgenommen werden.



7. Anschließend hat man folgende Ansicht:

Dies zeigt, dass die Daten nicht nur in Tabellenform, sondern auch praxisbezogen ausgegeben werden können.

5.2 Erweiterung des Modells – die Datenbank Verein3

Wie schon in der Einführung (Kapitel 1.1) festgestellt wurde, enthält das Modell Verein1 viele strukturelle Fehlerquellen. Aus diesem Grund soll es nun erweitert werden.

5.2.1 Die neue Tabelle PLZ - Tabelle für die Postleitzahlen erstellen

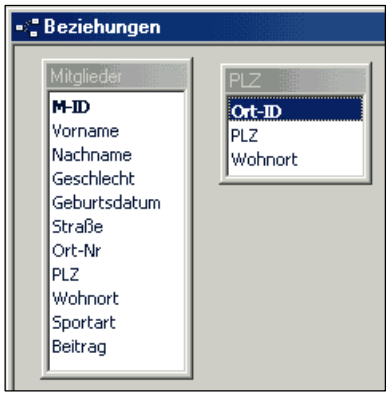
Aus der Tabelle **Adressen** sind die Daten der Spalte *PLZ* und *Wohnort* in jeweils eine gesonderte Tabelle zu bringen. Hierzu sind folgende Einzelschritte notwendig:

1. Zunächst kopiert man im *Explorer* die Datei Verein1.mdb auf Verein3.mdb. (Von Access aus nicht möglich. Siehe auch Hinweis Kapitel 5.1).
2. Verein3.mdb von Access aus öffnen und die Tabelle **Adressen** in **Mitglieder** umbenennen.
3. Neue Abfrage für Tabelle **Mitglieder** erstellen, die nur die PLZen (Sortierung: Aufsteigend) und Orte enthält und Abfrage ausführen. In der Datenblattansicht werden alle 25 vorkommenden PLZen und Orte gezeigt.
4. In der Entwurfsansicht dann aus Menü *Ansicht* die Option *Funktionen* auswählen ⇒Gruppierung ⇒ in der Datenblattansicht werden nur die 20 einfach vorkommenden PLZen und Orte gezeigt.
5. Zurück in den Abfrageentwurf gehen. Im Menü *Abfrage* die Option *Tabellenerstellungsabfrage* auswählen und den Tabellenname **PLZ** eingegeben.
6. Durch Drücken des Icons „**!**“ wird die Tabelle **PLZ** mit den Feldern *PLZ* und *Ort* erstellt sowie die Daten in die zugehörigen Felder eingefügt. Die Datentypen werden aus der bestehenden Tabelle Mitglieder übernommen (Felder: PLZ, Wohnort, Datentypen: Text 5, Text 20).
7. Schließen der Abfrage (Abfrage nicht speichern! Da bei jedem erneuten Ausführen der Tabellenerstellungsabfrage die entsprechende Aktion ausgeführt wird und die bestehende Tabelle überschrieben wird).
8. Nun öffnet man die Entwurfsansicht der Tabelle **PLZ** und fügt an der ersten Position das neue Feld *Ort-ID* vom Typ *AutoWert* ein. Das Feld *Ort-ID* wird dabei zum Primärschlüssel (Einstellung in der Entwurfsansicht der Tabelle durch Anklicken des Schlüssels in der Menüleiste). Beim Umschalten in die Datenblattansicht der Tabelle **PLZ** (inkl. Abspeichern) werden die Werte im Feld *Ort-ID* durch automatisches Hochzählen erstellt.
9. Anschließend wird in der Tabelle **Mitglieder** das Feld *Ort-Nr* an der entsprechenden Stelle eingefügt. ACHTUNG: Der Feldtyp muss **Zahl - Long Integer** sein, da sonst keine Beziehung zur *Ort-ID* vom Typ *AutoWert* möglich ist.

5.2.1.1 Verknüpfen der Tabelle Mitglieder mit der Tabelle PLZ

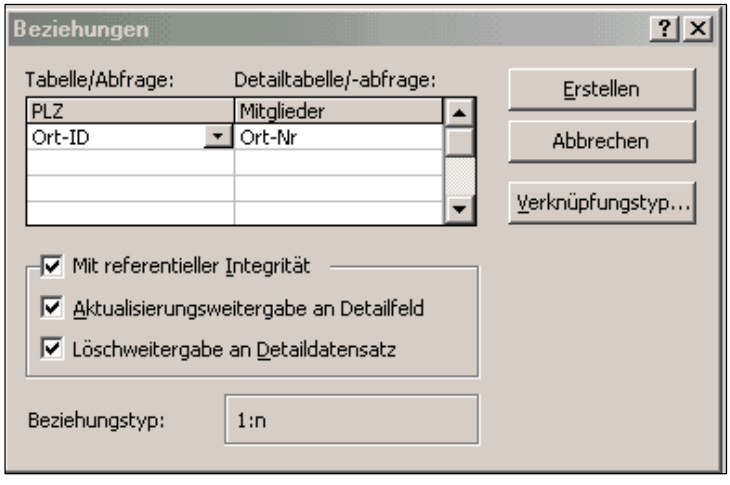
Vom Datenbankfenster aus klickt man das Icon  (Beziehungen) an. Nun wird die Tabelle **Mitglieder** ausgewählt und dann die Taste *Hinzufügen* gedrückt. Dies wird für **PLZ** wiederholt und dann durch die Taste *Schließen* betätigt.

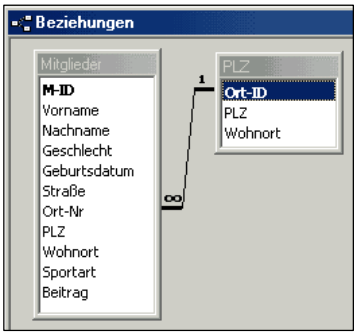
Nebenstehende Abbildung zeigt **Beziehungen** mit Anzeige der beiden Tabellen:



Klickt man jetzt mit der Maus in Tabelle **PLZ** auf *Ort-ID* (**Mastertabelle** bzw. 1-Seite der Beziehung) und zieht den Mauszeiger bei gedrückter Maustaste zur Tabelle **Mitglieder** und hier auf *Ort-Nr* (**Detailtabelle**, n-Seite der Beziehung, d. h. **vom Primärschlüssel zum Fremdschlüssel!**) so erscheint nach dem Loslassen der Maustaste das rechts angezeigte Bild:

Hier werden die entsprechenden Felder angekreuzt und das Fenster mit *Erstellen* geschlossen.





Es wird jetzt die fertige 1-n-Beziehung zwischen den beiden Tabellen angezeigt.

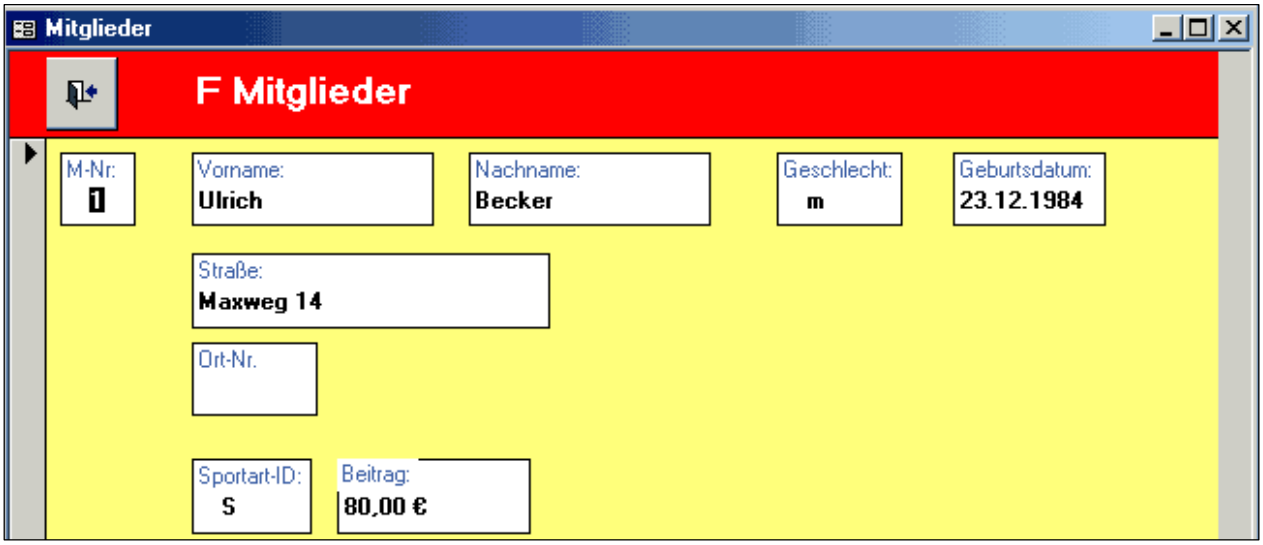
Ort-ID ist Primärschlüssel in der Tabelle **PLZ** und *Ort-Nr* ist Fremdschlüssel in der Tabelle **Mitglieder**.

Nun werden sowohl in der Tabelle **Mitglieder** als auch im zugehörigen Formular die Felder *PLZ* und *Wohnort* gelöscht.

5.2.1.2 Formular für die Dateneingabe in die Tabelle Mitglieder erstellen

Im Datenbankfenster die Registerkarte *Formular* auswählen und dann die Schaltfläche *Neu* anklicken.

Nach Auswahl der Tabelle **Mitglieder** ist mit Hilfe des Formularassistenten folgendes Eingabeformular zu erstellen und unter **F Mitglieder** abzulegen:

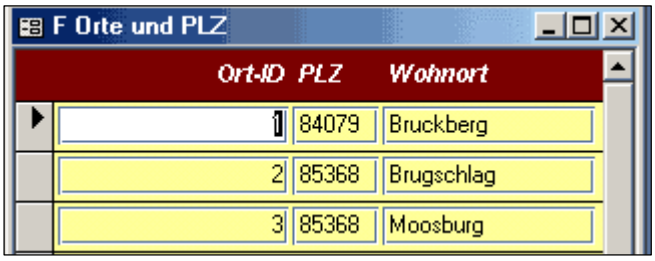


5.2.1.3 Formblatt für die Dateneingabe in die Tabelle PLZ erstellen

Im Datenbankfenster die Registerkarte *Formular* auswählen und dann die Schaltfläche *Neu* anklikken.

Nach Auswahl der Tabelle **PLZ** ist mit Hilfe des Formularassistenten (Tabellarisch) folgendes Eingabeformular zu erstellen:

(Speichern unter **F Orte und PLZ)**



5.2.1.4 Einfügen des Listenfeldes für die Auswahl der Ortskennzahl

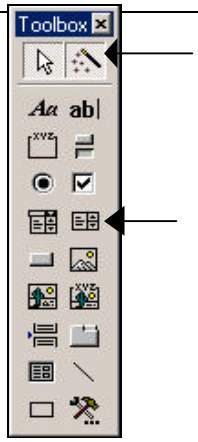
1. Erstellen der Abfrage *Orte (alphabetisch).*

Die Abfrage ist wie die nebenstehende Abbildung des Abfrageentwurfs zeigt, vorzunehmen und anschließend zu speichern.

Feld:	Ort-ID	PLZ	Wohnort
Tabelle:	PLZ	PLZ	PLZ
Sortierung:			Aufsteigend
Anzeigen:	☑	☑	☑
Kriterien:			
oder:			

Hinweis: In den Einstellungen des Eigenschaftsfensters des Listenfeldes (im Feld *Datensatzherkunft*) kann später auch die alphabetische Sortierung der Orte vorgenommen werden ohne dass hierzu eine eigene Abfrage erstellt werden muss (siehe Kapitel 5.3.2).

2. In das Formular **F Mitglieder** gehen und hier in die Entwurfsansicht umschalten.
3. Die Toolbox mit  einschalten und den Steuerelementassistenten aktivieren.



Steuerelementassistenten eingeschaltet

1. Listenfeld mit der Maustaste Anklicken.
2. Mit dem Mauszeiger im Detailbereich an die Position gehen, an der das Listenfeld gewünscht wird.
3. Mit gedrückter Maustaste die Größe des Listenfeldes erstellen.
4. Maustaste loslassen. Es meldet sich der Listenfeldassistent.

4. Listenfeld mit dem Assistenten wie in der Bilderfolge 1-6 einfügen:

   Hinweis zum Einsatz eines Kombinationsfeldes bzw. Unterformulars:

   In Kapitel 5.3 wird die Erstellung eines professionellen Formulars mit Hilfe eines Kombinationsfeldes besprochen.

Bild 1

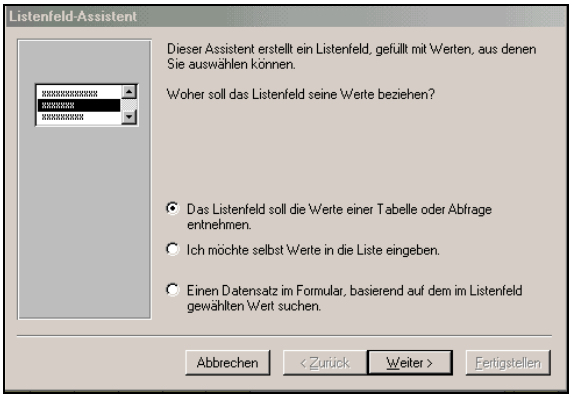


Bild 2

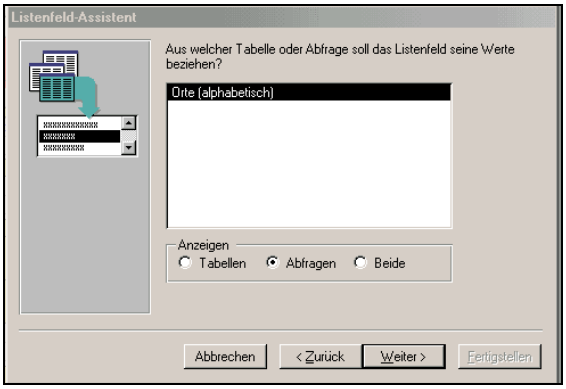


Bild 3

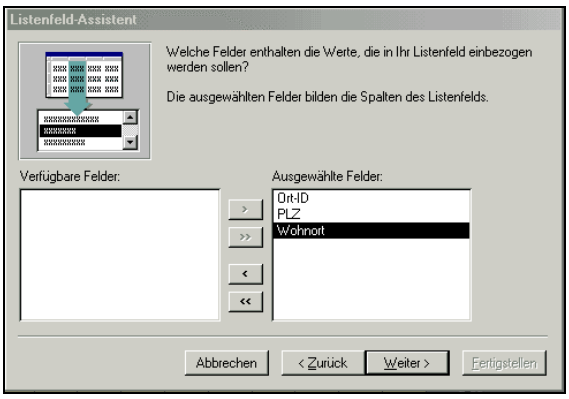


Bild 4

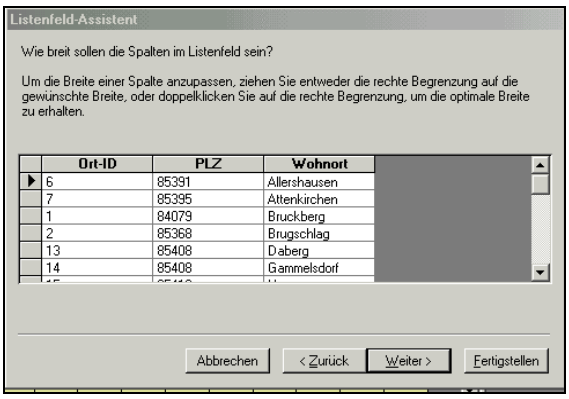


Bild 5

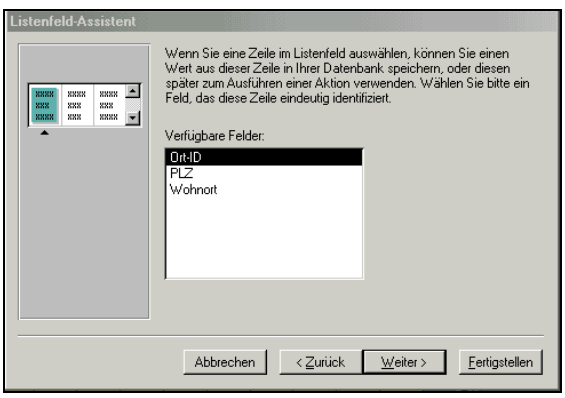
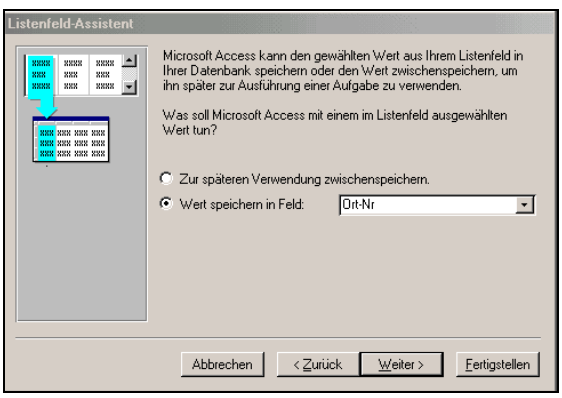


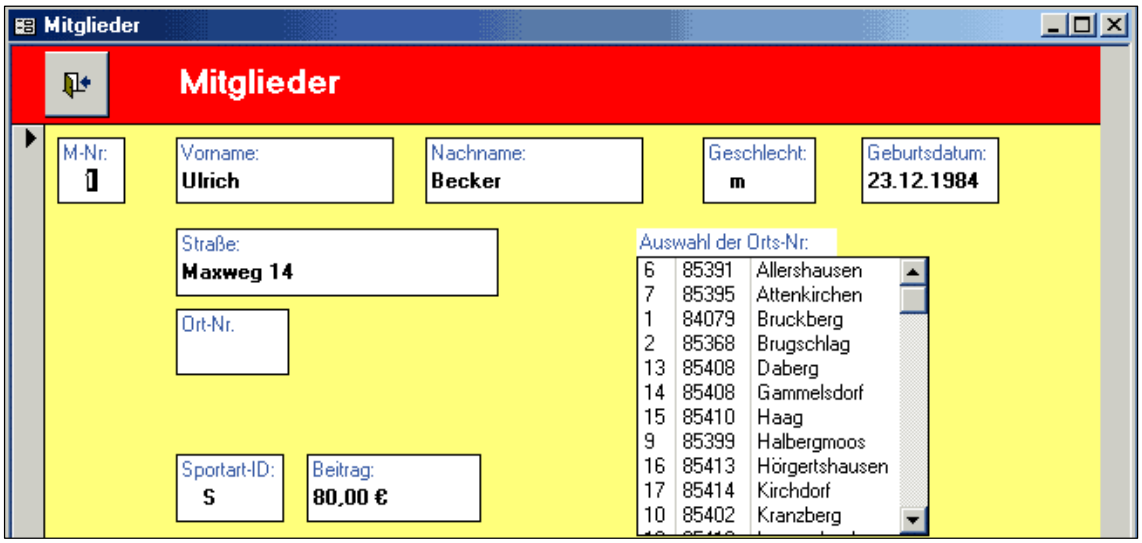
Bild 6



Nach *Weiter* ist noch die Bezeichnung **Orts-Nr-Auswahl** einzugeben und mit *Fertigstellen* abzuschließen.

5. Das Beschriftungs- und Listenfeld in der Breite anpassen und in die Formularansicht umschalten.
6. In der Formularansicht die entsprechenden Ortskennzahlen durch Anklicken im Listenfeld für die einzelnen Mitglieder eingegeben.

Fertiges Formular:



5.2.2 Die neuen Tabellen *Sportarten* und *Zuordnung: M-Nr Sport-Nr*

5.2.2.1 Tabelle für Sportarten erstellen

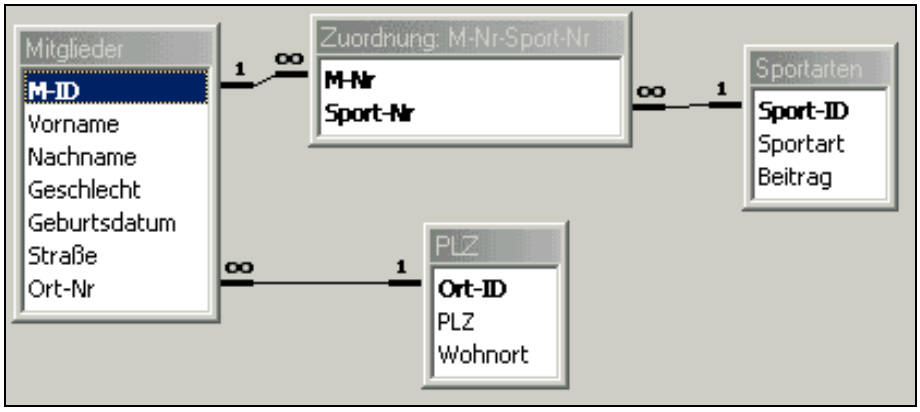
Tabelle **Sportart** erstellen (Felder: *Sport-ID, Sportart, Beitrag*, Datentypen: AutoWert, Text 20, Währung). Das Feld *Sport-ID* wird dabei zum Primärschlüssel (Einstellung in der Entwurfsansicht der Tabelle).

5.2.2.2 Tabelle für Zuordnung: Mitglieder-Sportart erstellen

Tabelle **Zuordnung: M-Nr-Sport-Nr** erstellen (Felder: *M-Nr, Sport-Nr*, Datentypen: jeweils Long Integer). Die Felder *M-Nr* und *Sport-Nr* werden als kombinierter Primärschlüssel festgelegt. Durch diese Festlegung ist sichergestellt, dass die Kombination aus *Mitglieds-Nr* und *Sportabteilung* einmalig bleibt.

5.2.2.3 Verknüpfen aller Tabellen

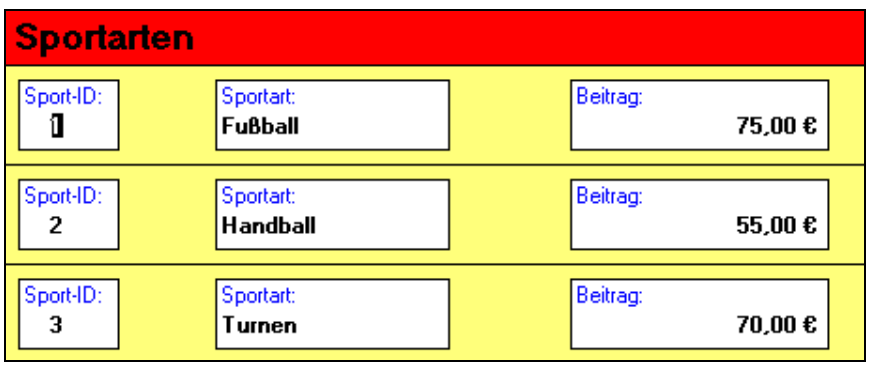
Vom Datenbankfenster aus das Fenster Beziehungen öffnen und alle Tabellen entsprechend der folgenden Abbildung anordnen und verknüpfen.



5.2.2.4 Formular für die Eingaben von Daten in die Tabellen *Sportart*

Formblatt erstellen und Daten in die Tabelle eingeben

Mögliches Formblatt:
Hier als Endlosformular.



5.3 Formular

5.3.1 Grundlagen

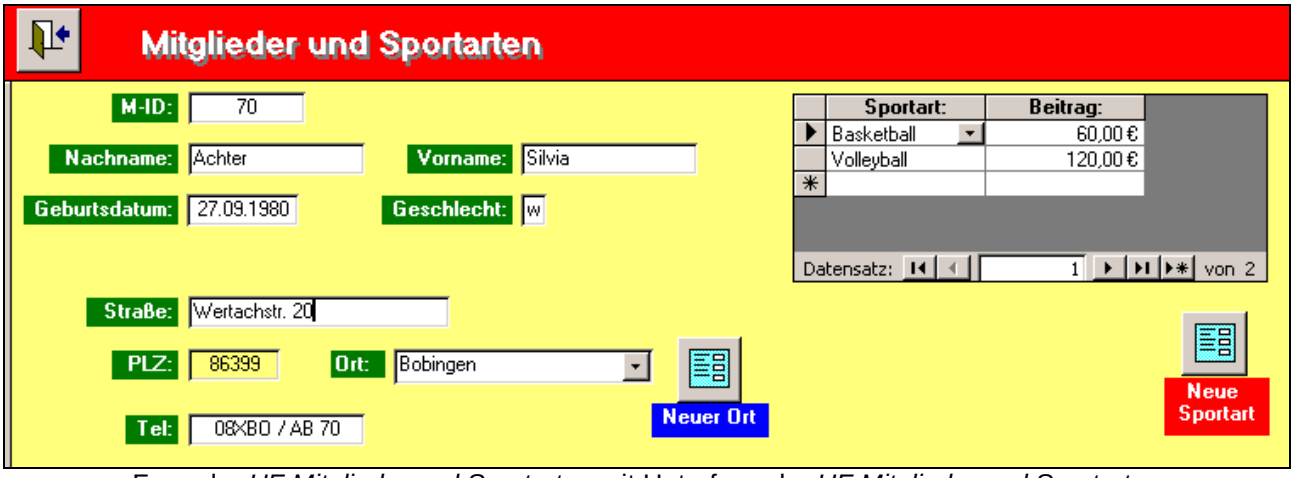
Ein Formular ist die graphisch gestaltete Schnittstelle zwischen Computerbenutzer und Daten. Formulare müssen deshalb mit großer Sorgfalt konstruiert werden.

In den vorangegangenen Kapiteln wurden schon Formulare erstellt, so dass die Technik der Formularerstellung mit Hilfe des Assistenten sowie der grundlegende Aufbau eines Formulars (Formularkopf, Detailbereich und Formularfuß) bekannt sind. Auch die ökonomische Eingabe und Pflege von Daten mit Hilfe von Formularen anstelle von Tabellen ist dem Benutzer vertraut. Außerdem wurden erste Erfahrungen bei der Auswahl von Daten aus Listenfeldern gesammelt. Allerdings waren die bisher erstellten Formulare noch nicht sehr professionell, d. h. bei der Eingabe von Orten, Sportarten, Mitgliedern und deren Zuordnung mussten bisher 4 Formulare bearbeitet werden. Das Ziel sollte sein, dass dies alles von einem Formular aus möglich ist. In diesem Kapitel geht es also in erster Linie darum, Formulare benutzerfreundlich und sachgerecht zu gestalten. Dabei wird, insbesondere bei der Verwendung von Unterformularen, die Kenntnis der Datenbankstruktur des logischen Modells eine wesentliche Rolle spielen, so dass an dieser Stelle die Bedeutung des systematischen Datenbankentwurfs wieder voll zum Tragen kommt.

5.3.2 Das Formular „*HF Mitglieder*“ mit dem Unterformular „*UF Sportarten*“

Ab der Version Access 97 ist der Formularassistent in der Lage in bestimmten Fällen (s. auch Hinweis in Kap. 5.3.3) zu erkennen, ob das Formular in ein Haupt- und ein Unterformular unterteilt werden soll. Dies ist immer dann sinnvoll, wenn in einem Formular Daten aus mehreren Tabellen, die in einer n-m-Beziehung zueinander stehen, zusammengeführt werden.

Das folgende Bild zeigt ein Muster des zu entwickelnden Formulars.



Formular *HF Mitglieder und Sportarten* mit Unterformular *UF Mitglieder und Sportarten*

Man erkennt, dass die Daten des Formulars aus allen 4 Tabellen der Datenbank stammen:

Tabelle	**Felder**
PLZ	*PLZ, Ort*
Mitglieder	*M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, Tel, Orts-Nr*
Zuordnung...	*M-Nr, Sport-Nr*
Sportarten	*Beitrag*

Das Formular enthält Daten aus den Tabellen *Mitglieder* und *Sportarten*, die in einer n-m-Beziehung stehen; folglich ist für die Verwaltung dieser Daten ein Unterformular sinnvoll. Für die Eingabe und Pflege der Daten aus der Tabelle *PLZ* (1-n-Beziehung zu Tabelle *Mitglieder*) genügt ein Kombinationsfeld.

Dabei ist der folgende wesentliche Aspekt zu beachten: die Fremdschlüsselfelder *Orts-Nr*, *M-Nr* ,*Sport-Nr* erscheinen nicht im Formular, sind aber zur Abspeicherung der Daten absolut notwendig (z. B. Abspeicherung der Zugehörigkeit zu einer Sportart bzw. zu einem Wohnort). Folglich erstellt man sich zunächst eine Abfrage (z. B. *Daten für HF*), die alle obengenannten Felder (*PLZ, Ort M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, Tel, Orts-Nr M-Nr, Sport-Nr Beitrag*) enthält. Anschließend kann man mit Hilfe des Formularassistenten ein neues Formular erstellen, das die Abfrage *Daten für HF* zugrunde legt. Man wählt alle Felder der Abfrage *Daten für HF* aus und entscheidet, die Daten nach der Tabelle Mitglieder anzuzeigen. Für das Layout des Unterformulars (UF) wird die Datenblattansicht gewählt, der Stil ist z. B. Standard. Anschließend legt man die Namen für das Hauptformular (HF) und das UF fest (*HF Mitglieder und Sportarten* bzw. *UF Mitglieder und Sportarten*). Und schon sind HF und UF fertig. Es bedarf nun aber noch einiger Nachbesserungen, die in den Kapiteln 5.3.3 und 5.3.4 beschrieben werden.

5.3.3 Befehlsschaltflächen / Kombinationsfelder

Bei dem Symbol mit der sich schließenden Türe im Formularkopf handelt es sich um eine sogenannte Befehlsschaltfläche, mit der eine bestimmte Aktion ausgelöst werden kann (in diesem Fall das Schließen des Formulars). Diese Befehlsschaltfläche wird folgendermaßen erstellt:

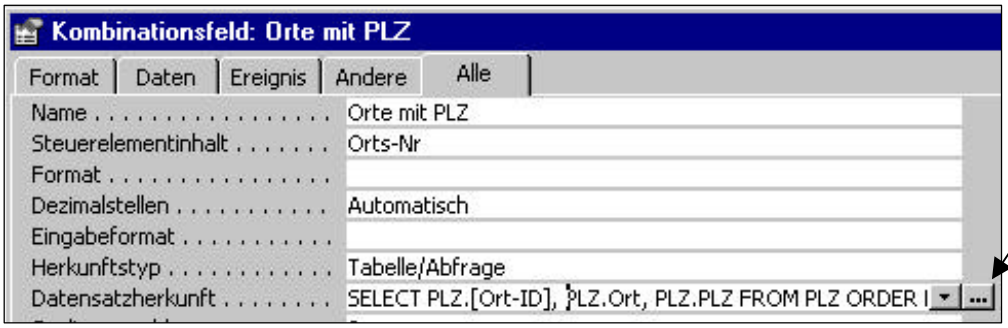
1. Man wählt das Symbol für die Befehlsschaltfläche aus der Toolbox aus und zieht im Entwurfsmodus an der entsprechenden Stelle die gewünschte Fläche auf. Anschließend erscheint ein Menü, das verschiedene Kategorien von Aktionen anbietet. Hier wird die Kategorie *Formularoperationen* sowie die Aktion *Formular schließen* ausgewählt und *weiter* gedrückt. Anschließend kann man sich entscheiden, ob die Befehlsschaltfläche einen Text oder ein Symbol beinhalten soll; hier wurde das entsprechende Symbol gewählt.
2. Die Steuerelemente *Eingabe eines neuen Ortes* bzw. *Eingabe einer neuen Sportart* sind sowohl Befehlsschaltflächen als auch Bezeichnungsfelder. Die Befehlsschaltflächen werden ebenso wie unter 1. erstellt (die Aktion ist diesmal aber *Formular öffnen*, anschließend wird das jeweilige Formular ausgewählt). Die Bezeichnungsfelder werden wiederum mit Hilfe der Toolbox erstellt und anschließend formatiert.

Die Auswahl des Ortes unter dem Bezeichnungsfeld *Ortswahl* erfolgt per Auswahl aus einer vorgegebenen Liste (sogenanntes Kombinationsfeld). Für eine solche Liste stehen Daten zur Verfügung, die entweder aus einer Tabelle / Abfrage ausgewählt werden können. Der Benutzer kann auch Daten für eine bestimmte Liste direkt eingeben. Wie ein derartiges Kombinationsfeld erstellt wird, soll am Beispiel des entsprechenden Feldes für die Auswahl des Ortes gezeigt werden.

3. Als erstes wählt man aus der Toolbox (s. Seite 42) die entsprechende Schaltfläche per Mausklick aus (es ist darauf zu achten, dass der Steuerelementassistent eingeschaltet ist). Nach Auswahl der Befehlsschaltfläche für das Kombinationsfeld zieht man an der passenden Stelle des Formulars ein geeignetes Rechteck auf. In dem sich anschließenden Dialog mit dem Kombinationsfeldassistenten wird interaktiv die folgende Auswahl getroffen:

	Frage(n):	**Antwort:**
1	Werte aus Tabelle/Abfrage Werte selbst eingeben	Werte aus Tabelle/Abfrage
2	Angebot an Tabellen /Abfragen	Auswahl der Tabelle *PLZ*
3	Auswahl der Felder der gewählten Tabelle	In diesem Fall alle Felder, in der Reihenfolge: Ort-ID, Ort, PLZ
4	Vorschau auf die Spalten der ausgewählten Tabelle. Das Primärschlüsselfeld sollte nicht angezeigt werden, wird aber zur eindeutigen Auswahl benötigt. In der Formularansicht wird nur das erste sichtbare Feld angezeigt, dies sollte hier der Ort sein.	Die Spaltenbreiten können noch optimiert werden.
5	Was soll mit dem ausgewählten Wert geschehen? 1. Zur späteren Verwendung zwischenspeichern? 2. Wert speichern in bestimmtem Feld?	Auswahl von 2.: Der Wert des Primärschlüssels *Ort-ID* wird in dem Feld *Orts-Nr* (Fremdschlüssel) abgespeichert.
6	Nun wird nach der Bezeichnung (Erklärung) für das Kombinationsfeld gefragt. Der Benutzer wird aber vom Assistenten **nicht** nach dem Namen für das Kombinationsfeld gefragt. Trotzdem sollte später über das Eigenschaftsfenster ein Name für das Kombinationsfeld vergeben werden.	Hier wird die *Bezeichnung Ortswahl:* eingegeben. In diesem Beispiel bietet sich als Name für das Kombinationsfeld die Bezeichnung *Orte mit PLZ* an.
7	Nun ist das Kombinationsfeld fertig.	Nachbesserungen (z. B. Sortierreihenfolge der Orte, Spaltenanzahl, Spaltenbreiten) können über das zugehörige Eigenschaftsfenster vorgenommen werden.

Eine Nachbesserung ist z. B. die alphabetische Sortierung der Orte im Kombinationsfeld. Dies erreicht man dadurch, dass man im zugehörigen Eigenschaftsfenster den Punkt *Datensatzherkunft* auswählt und auf die in dieser Zeile erscheinenden 3 Punkte klickt. Nun öffnet sich das Entwurfsfenster des zugrunde gelegten SQL-Statements als QBE-Abfrage. Hier können die gewünschten Verbesserungen eingestellt und anschließend abgespeichert werden (z. B. alphabetische Reihenfolge).



Eigenschaftsfenster des Kombinationsfeldes *Orte mit PLZ*

Entsprechend soll die Auswahl der Sportart im Unterformular *UF Mitglieder und Sportarten* als Kombinationsfeld gestaltet werden. Man öffnet das Unterformular in der Entwurfsansicht, erstellt ein Kombinationsfeld, das auf der Tabelle *Sportarten* aufbaut und die Felder *Sportart* und *Sport-ID* enthält. Bei der Einstellung der Spaltenbreiten wird die Spalte mit der *Sport-ID* in der Breite vollständig auf 0,00 cm eingestellt (soll nicht angezeigt werden), aber *Sport-ID* (aus Tabelle Sportarten) wird als der Wert ausgewählt, der in dem Feld *Sport-Nr* (Tabelle *Zuordnung: M-Nr-Sport-Nr*) abgespeichert wird. Die Beschriftung des Kombinationsfeldes ist *Sportart:*. Nun muss man noch den Namen des Kombinationsfeldes festlegen. Dies geschieht im zugehörigen Eigenschaftsfenster in der ersten Zeile (hier wird der Name *Sportart* vergeben). Anschließend wird die Reihenfolge der Datenfelder im Formular bestimmt (Hauptmenüleiste *Ansicht,* Option *Aktivierreihenfolge:* hier wird das per Markierung ausgewählte Feld an die entsprechende Position in der Reihenfolge gezogen). Das jetzt für die Anzeige nicht mehr benötigte Feld *Sport-Nr* wird gelöscht.

Hinweis 1: Nach Erstellen und Austesten der Funktionsweise des Kombinationsfeldes *Orte mit PLZ* sollten die “überflüssigen Felder” *Orts-Nr* und *Ort* wegen der besseren Übersicht wieder entfernt werden.

Hinweis 2: Damit bei PLZ- oder Ortsänderungen eines Mitgliedes diese Änderung nicht auf der 1-Seite der Beziehung (in der Tabelle *PLZ*) erfolgt, soll das Feld *PLZ* des Hauptformulars für Änderungen gesperrt werden. Dies geschieht im Eigenschaftsfenster des Feldes: in den Einstellungen für *aktiviert* (einstellen auf *nein*) und *gesperrt* (einstellen auf *ja*). Eine Änderung der PLZ eines Mitgliedes im Feld *PLZ* (auf der 1-Seite) hätte sonst zur Auswirkung, dass diese Änderung in der Tabelle PLZ (Mastertabelle) durchgeführt wird und somit für alle anderen Mitglieder aus dem gleichen Ort wirksam wird.

Hinweis 3: Wurde ein neuer Datensatz für Ort bzw. Sportart durch einen Wechsel in das entsprechende Formular eingegeben (über die oben konstruierte Schaltfläche) und gelangt der Benutzer in das HF zurück, kann folgende Operation ausgeführt werden, damit die Neueingabe in der Auswahl des betreffenden Kombinationsfeldes sichtbar wird: (Hauptmenü *Datensätze,* Option *Anzeige aktualisieren*). Eine andere Möglichkeit besteht darin, im Eigenschaftsfenster des Kombinationsfeldes dem Ereignis „Beim Hingehen“ eine Ereignisprozedur zuzuordnen, die lediglich den Befehl Me.Refresh enthält.

Zu beachten ist, dass die Strukturen bzw. Verknüpfungen des zugrunde gelegten relationalen Modells entscheidend zum Verständnis der Konstruktion des Formulars und Unterformulars sowie der zugehörigen Abfragen und Kombinationsfelder beitragen.

Weitere Hinweise und „Tricks“ findet man im Kapitel 5.5.3

5.3.4 Details von Haupt- und Unterformular

Wenn man wie in Kapitel 5.3.2 das HF zusammen mit dem UF mit Hilfe des Assistenten erstellt, kann man sich im nachhinein fragen: was hat der Assistent mit der Abfrage *Daten für HF* gemacht? Dies lässt sich leicht feststellen, wenn man im jeweiligen Eigenschaftsfenster des HF bzw. UF unter der Rubrik Datenherkunft nachsieht. Hier stellt man fest: der Assistent löst die Abfrage in zwei SQL-Statements auf!

Dabei enthält das SQL-Statement des HF nur die Tabellen *PLZ* und *Mitglieder* sowie die für das HF notwendigen Felder: *M-ID, Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, Tel, Orts-Nr, PLZ, Ort*. Nach Erstellen des Kombinationsfeldes *Orte mit PLZ* könnte das Feld *Ort* auch noch aus dem SQL-Statement entfernt werden. Wie oben aber schon erwähnt, muss das Feld *Orts-Nr* bestehen bleiben, obwohl es im Formular nicht erscheint, da es zur Abspeicherung der aus dem Kombinationsfeld ausgewählten *Ort-ID* dient (Auswahl erfolgt aber über den Ortsnamen und zugehörige PLZ).

Entsprechend enthält das SQL-Statement des UF nur die Tabellen *Zuordnung:M-Nr-Sport-Nr* und *Sportarten* sowie die für das Formular notwendigen Felder: *M-Nr, Sport-Nr* und *Beitrag*. Alle Felder sind notwendig, obwohl im UF später nur das Feld Beitrag sichtbar ist. Die *Sport-Nr* dient wie im HF die *Orts-Nr* zur Abspeicherung der aus dem Kombinationsfeld für die Sportarten ausgewählten *Sport-ID*. Außerdem gilt: das UF muss mit dem HF verknüpft werden, damit bei jedem Mitglied nur die von ihm betriebenen Sportarten angezeigt werden und nicht alle Einträge aus der Tabelle *Zuordnung:M-Nr-Sport-Nr*. Wird das UF zusammen mit dem HF erstellt, erledigt der Assistent diese Verknüpfung automatisch. Ebenso wenn das UF mit dem entsprechenden Symbol aus der Toolbox im HF eingefügt wird. In allen anderen Fällen muss die Verknüpfung des HF mit dem UF im Eigenschaftsfenster des UF selbsttätig durchgeführt werden (s. untenstehende Skizze). In unserem Beispiel muss in der Zeile „Verknüpfen von“ der Feldname *M-Nr* und in der Zeile „Verknüpfen nach“ der Feldname *M-ID* eingetragen werden. Dabei erscheint jeweils unten in der Statusleiste ein Hinweis, wenn sich der Cursor in der betreffenden Zeile befindet:
In der Zeile „Verknüpfen von“: Feldname(n) in untergeordnetem Objekt.
In der Zeile „Verknüpfen nach“: Feldname(n) in Hauptformular oder Bericht.

Hinweis: das nachträgliche Einfügen von Unterformularen in bestehende Formulare ist in zwei Fällen wichtig und notwendig, da in diesen Fällen der Assistent keine Unterstützung bietet:

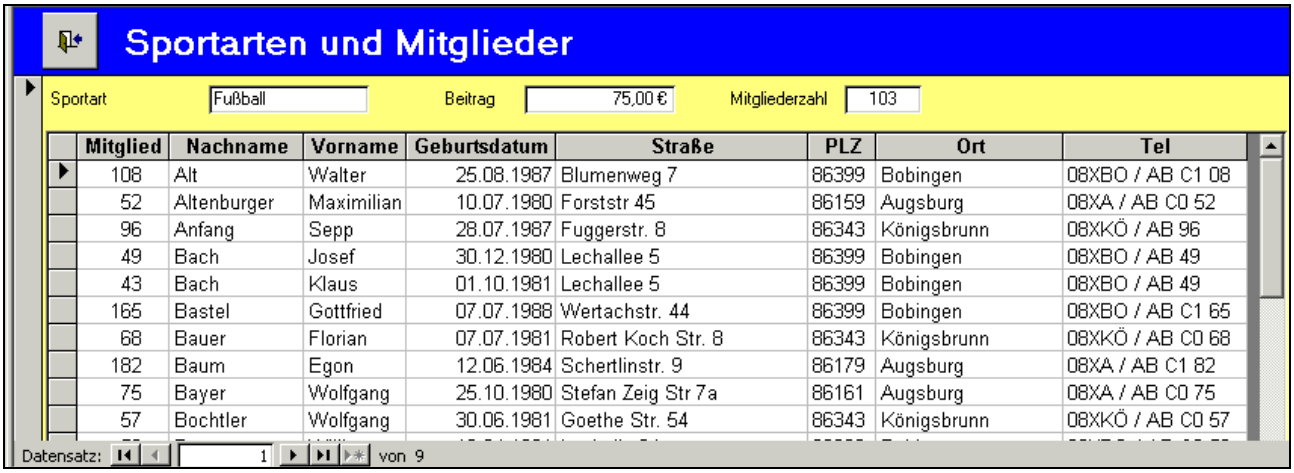
1. Das Hauptformular enthält mehr als ein Unterformular. HF enthält z. B. Mannschaften mit bestimmten Eigenschaften, wie zugehörige Spieler bzw. Trainer und Betreuer, die jeweils in eigenen Unterformularen verwaltet werden (vgl. Verein6).
2. Das Hauptformular enthält ein Unterformular, das wiederum ein Unterformular enthält. HF enthält z. B. Kunden, die über Aufträge Artikel kaufen oder leihen. Das 1.te UF enthält die Aufträge (ein Kunde kann mehrere Aufträge stellen). Das 2.te UF enthält die zugehörigen Auftragspositionen des jeweiligen Auftrags (ein Auftrag kann mehrere Positionen beinhalten).

Folgerung: die Abfrage *Daten für HF* ist nach dem Erstellen des Formulars *HF Mitglieder und Sportarten* überflüssig und kann wieder gelöscht werden.

Merke: Fremdschlüssel dienen zur Abspeicherung und können ihren Wert ändern. Primärschlüssel ändern ihren Wert nicht; sie dienen zur eindeutigen Identifizierung.

5.3.5 Formular Sportarten und Mitglieder

Wie in Kapitel 5.3.2 kann das Formular der Mitglieder und Sportarten aber auch von der Sportartenseite her betrachtet werden, d.h. die Sportarten stehen im HF und die zugehörigen Mitglieder im UF. Dann sieht das entsprechende Formular z. B. folgendermaßen aus:



5.4 Bericht

Mit einem Bericht kann man den Inhalt der Datenbank nach frei wählbaren Kriterien zusammenfassen. Die Art und Weise, wie der Ausdruck erfolgen soll, legt man im Berichtsentwurf fest. Eine Möglichkeit, die Daten wie bei einem Formular zu bearbeiten, gibt es nicht. Einen Bericht kann man folglich nur in der Entwurfs- oder Seitenansicht betrachten. Grundlage für einen Bericht ist eine Tabelle oder eine Abfrage, wenn man den Ausdruck auf bestimmte Datensätze beschränken möchte. Ein Bericht eignet sich also nicht als Eingabemaske für bestimmte Daten, ist aber i. a. wesentlich besser für die Ausgabedarstellung geeignet als ein Formular, wenn es sich nicht um eine Diagrammdarstellung handelt.

Beispiel: Daten werden häufig in Kategorien eingeteilt und dann ausgedruckt. Mit einem Bericht kann man beispielsweise die Summe jeder Kategorie errechnen und sie untereinander bzw. mit dem Gesamtbetrag vergleichen.

Einen Bericht erstellt man am einfachsten mit dem Berichtsassistenten (Analogon zum Formularassistenten). Mit ihm kommt man schnell zu einer brauchbaren Lösung, die man, wie schon beim Formular, entweder sofort verwenden oder noch weiter verfeinern kann. Das trifft insbesondere für Gruppenberichte zu, da der Assistent grundsätzlich den Inhalt aller numerischen Felder addiert. Das führt natürlich auch zu unsinnigen Lösungen (z. B. die Addition aller Angestelltennummern). MS-ACCESS stellt vier verschiedene Berichtsassistenten zur Verfügung, die die am häufigsten benötigten Lösungen abdecken:

- Die einfachsten Formen sind **AutoBericht: Einspaltig** und **AutoBericht: Tabellarisch.** Die Daten werden zeilenweise untereinander angeordnet. Die einzige Auswertungsmöglichkeit besteht darin, bestimmte Felder zu selektieren und eine Sortierfolge festzulegen.
- Der **Diagramm-Assistent** unterstützt beim Erstellen von Diagrammen.
- Mit dem **Etiketten-Assistent** kann man Karteikarten und Etikettenformulare bedrucken. Der Vorteil ist, dass man aus einer Vielzahl gängiger Formulargrößen auswählen kann und sich nicht um Einstellarbeiten kümmern muss.

Die Struktur eines Berichts kann maximal 7 Bereiche umfassen:

- Berichtskopf: Die erste Seite beginnt mit einer Überschrift.
- Seitenkopf und Seitenfuß: Jede Seite erhält eine Kopf- und Fußzeile.
- Detailbereich: Enthält die ausgewählten Datenfelder.
- Berichtsfuß: Auf der letzten Seite kann eine Summierung bestimmter Feldinhalte oder eine andere Berechnung ausgeführt werden.

Wenn man Datensätze gruppenweise auswertet, kommen zwei weitere Bereiche hinzu:

- Gruppenkopf: Eine Überschrift, die bei einem Gruppenwechsel gedruckt wird.
- Gruppenfuß: Eine mathematische Auswertung der Datensätze, die nur die Datengruppe betrifft.

Wie schon erwähnt, ist die Grundlage eines Berichts eine Tabelle oder eine Abfrage; auf dieser wird dann am einfachsten mit dem betreffenden Berichtassistenten das gewünschte Grundgerüst für den Bericht erstellt. Dabei ist die Vorgehensweise nahezu dieselbe wie bei der Erstellung eines Formulars. Dies betrifft sowohl die Toolbox, die die gleiche ist wie bei Formularen, als auch die Steuerelemente, die dieselbe Bedeutung haben wie bei Formularen.

Zusammenfassend kann man also einen Bericht von einem Formular folgendermaßen unterscheiden:

- **Formular:**
  Ein Objekt, das man zum Eingeben, Ändern und Einsehen von Datensätzen benutzen kann. Man kann ein Formular dazu verwenden, einzelne Datensätze oder grafische Auswertungen am Bildschirm oder in gedruckter Form anzuzeigen.
- **Bericht:**
  Ein Objekt, das man zum Ausdrucken von Datensätzen in einem benutzerdefinierten Layout verwenden kann. Berichte dienen darüber hinaus dem Zusammenstellen von Datensätzen und dem Berechnen von Gruppierungsfunktionen für einzelne Datensatzgruppen und beispielsweise einer Gesamtsumme für den ganzen Bericht.

Erstellen eines Berichts in der Datenbank *Verein3*:

Zunächst wird die Abfrage *Sportabteilungen und sämtliche Mitgliederdaten* erstellt, die sämtliche Daten enthält, die in dem zu erstellenden Bericht benötigt werden; dazu gehören auch solche Datenfelder, die nicht zur Anzeige, sondern nur zum Gruppieren oder Sortieren benutzt werden.

Nun wird im Datenbankfenster die Registerkarte *Bericht* ausgewählt und dann die Schaltfläche *Neu* geklickt. Man verwendet den Berichtsassistenten und die Abfrage *Sportabteilungen und sämtliche Mitgliederdaten,* aus der sämtliche Felder übernommen werden.

Die Datenanzeige „*nach Sportarten*“ wird gewählt und das Fenster *Gruppierungsebene* mit *weiter* übergangen. Sortiert werden soll nach Nachname und Vorname. Im nächsten Bild kann der Berichtsstil bestimmt werden *(Links ausrichten 2*) und nach *Fertigstellen* ist der Bericht als Seitenansicht auf dem Bildschirm zu sehen. Er entspricht der Ausgabe über den Drucker. Zieht man den Cursor über den Bericht, so wirkt er wie eine Lupe. Durch Klicken mit der Lupe wird zwischen einer Groß- bzw. Kleindarstellung hin und her geschaltet. Nun ist der Bericht in der Entwurfsansicht so nachzubearbeiten, bis er in die unten gezeigte Form gebracht ist.

Der Bericht wird unter dem Namen *Abteilungen und Mitglieder* abgespeichert.

TSV Irgendhausen

Stand: 6.08.2002

86123 Irgendhausen, Sepp Herberger Str. 54

Tel.: 08123 / 4567

Fax.: 08123 / 5678

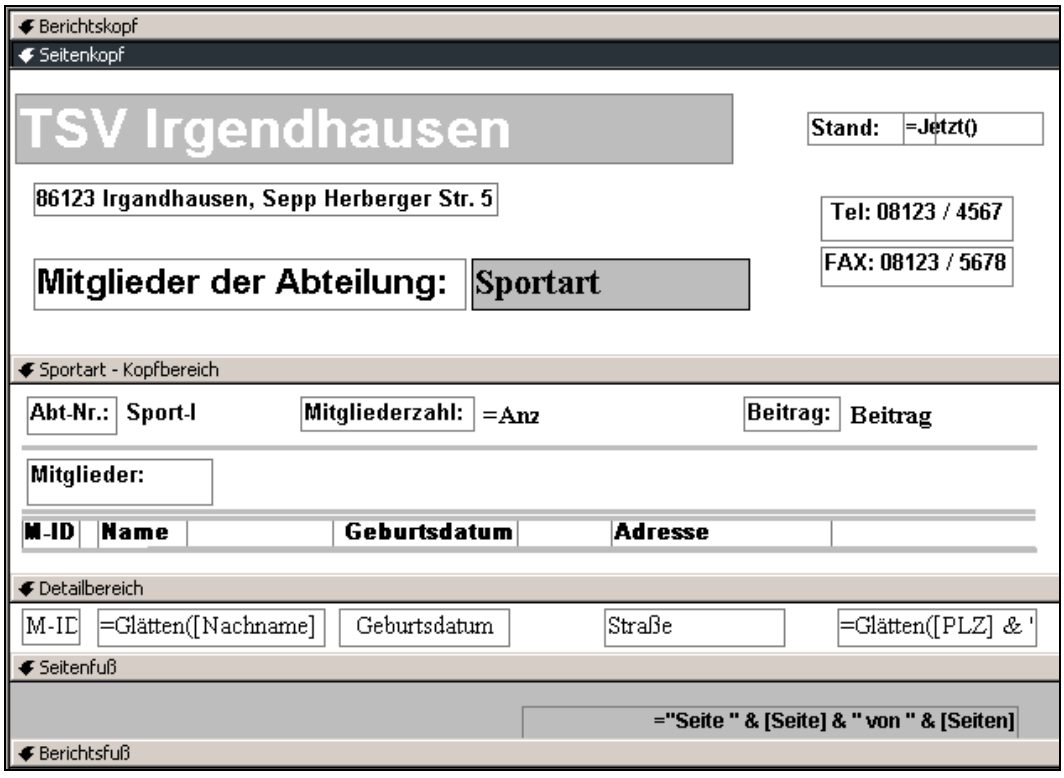
Mitglieder der Abteilung: **Aerobic**

Abt-Nr.: 11 **Mitgliederzahl: 10** **Beitrag: 100,00 €**

Mitglieder:

M-Nr	Name	Geb.datum	Adresse	
61	Amann, Ulrike	22.09.1980	Wertachstr. 10	86399 Bobingen
158	Börse, Hanna	07.08.1963	Illerstr. 3	86399 Bobingen
162	Hüpf, Johanna	05.09.1968	Postweg 5	86159 Augsburg
24	Kirchner, Martina	14.07.1965	Dieselstraße 27	86150 Augsburg
28	Niederlechner, Erika	12.11.1956	Leipzigerstr. 93	86343 Königsbrunn

Datenblattansicht des Berichts *Abteilungen und Mitglieder*

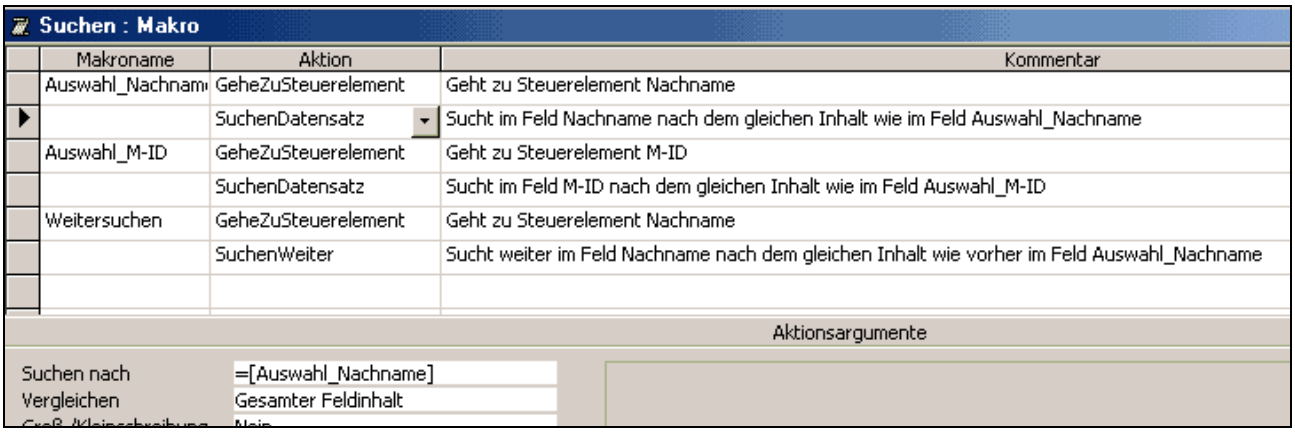


Entwurfsansicht des Berichts *Abteilungen und Mitglieder*

5.5 Makros

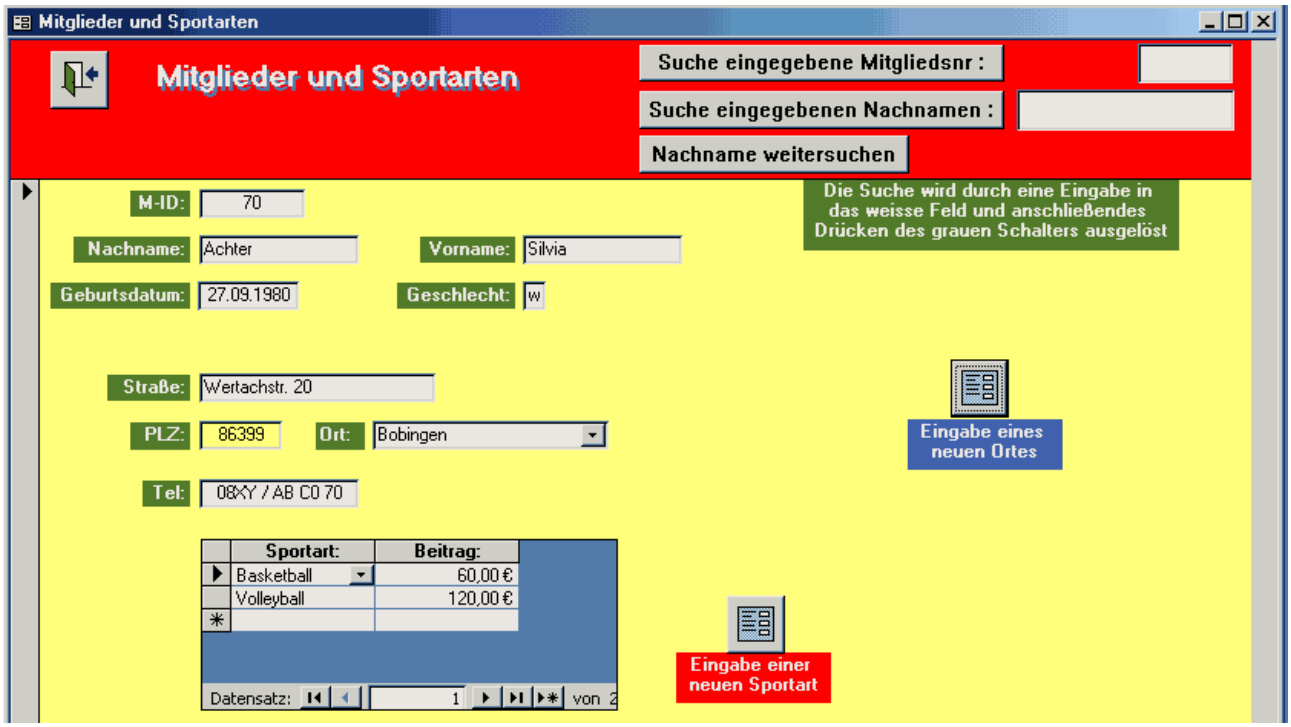
5.5.1 Grundlagen

Mit Makros kann man häufig wiederkehrende Routineaufgaben automatisieren. Mit jedem Makro wird mindestens eine Aufgabe oder Aktion ausgeführt, es können aber auch mehrere sein. Die Makroaktion wird unter einem Makronamen zusammengefasst und kann von einem Menüpunkt oder einer Schaltfläche aus gestartet werden. Ein Makro wird in einem Makrodokument erstellt. Häufig fasst man in einem Makrodokument mehrere Makroaktionen zusammen; man spricht dann von einer Makrogruppe (z. B. Makrogruppe *Suchen* in den Beispieldatenbanken *Verein4a* bzw. *Verein6a*).



5.5.2 Anwendungen in den Beispieldatenbanken Verein4a bzw. Verein6a

Das DBMS *Access* stellt eine Auswahl an vordefinierten Aktionen zur Verfügung, die durch eigene Angaben ergänzt werden können. So kann man schnell und nahezu problemlos eigene Makros zusammenstellen.



Die Erstellung eines Makros soll an einem Beispiel durchgeführt werden:

Aufgabenstellung: Im Formular für die Eingabe und Pflege der Mitgliederdaten soll nach einem bestimmten Mitgliedsnamen gesucht werden, so dass das Datenblatt dieser Person angezeigt wird.

Lösung: Zunächst wird im Formular (*Verein4: Mitglieder und Sportarten HF; Verein6: F Mitglieder und Abteilungen HF*) ein ungebundenes Steuerfeld (ab| aus der Toolbox) mit dem Namen *Auswahl_Nachname* erstellt. In dieses Steuerfeld kann später der zu suchende Nachnamen eingetragen werden.

Anschließend wählt man im Datenbankfenster durch Mausklick das Symbol für Makro aus und nimmt die Option *Neu*. Nun erscheint das Makroblatt in Entwurfsansicht. In dieses Makroblatt können dann mehrere Makros, eine sogenannte Makrogruppe, eingetragen werden. Wenn in einem Makroblatt nur ein Makro steht und außerdem kein Makroname eingegeben wurde, so wird das Makro mit dem Namen des Makroblattes gestartet. Man wählt als Makronamen die Bezeichnung *Auswahl_Nachname*. Wird die Spalte *Makroname*

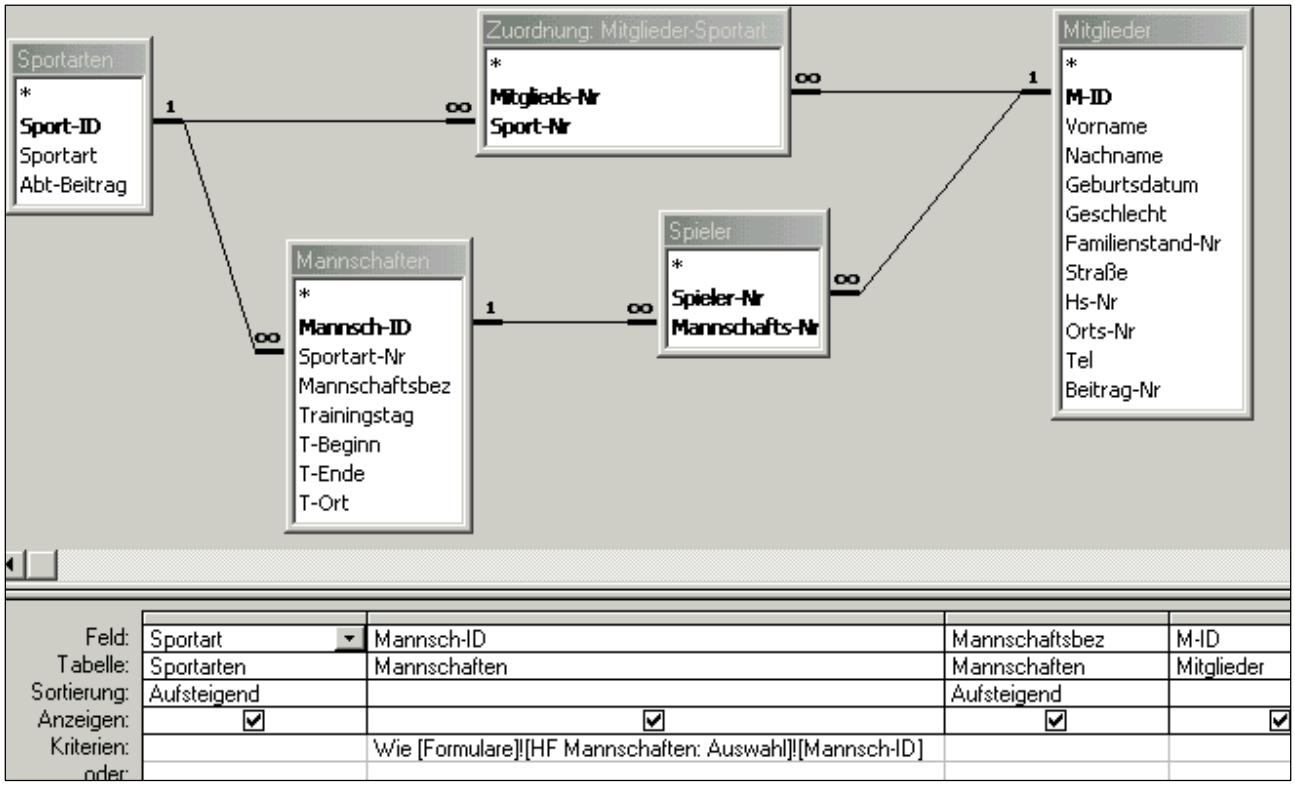
nicht angezeigt, so schaltet man sie durch den Menüpunkt *Ansicht* ein. Als erste Aktion ist *GeheZuSteuerelement* auszuwählen und im unten angezeigten Feld *Steuerelementname* die Bezeichnung *Nachname* einzugeben. Diese Aktion bewirkt, dass nach Ausführen des Makros der Cursor zuerst in das Steuerelement mit der Bezeichnung *Nachname* gesetzt wird. Als zweite Aktion wird der Befehl *SuchenDatensatz* aufgerufen. Die Abbildung der Makrogruppe zeigt im unteren Teil des Bildschirms die notwendigen Aktionsargumente. Im Feld *Suchen nach* muss eingeben werden, in welchem Feld der zu suchende Begriff steht. Hier wird die im Formular für das ungebundene Feld gewählte Bezeichnung =[*Auswahl_Nachname*] eingetragen (in diesem Fall identisch mit Makronamen). Die restlichen Eintragungen können übernommen werden; befindet man sich in der entsprechenden Zeile, wird ihre Bedeutung im Kommentarfeld angezeigt.

Nun erstellt man im entsprechenden Formular mit Hilfe der Toolbox eine ungebundene Befehlsschaltfläche an der gewünschten Position. Dieser Schaltfläche wird durch Anklicken in der Zeile *beim Klicken* im zugehörigen Eigenschaftsfenster folgende Aktion zugeordnet: *Suchen.Auswahl_Nachname*. (geschieht durch Mausklick und Auswahl aus der Liste der angezeigten möglichen Aktionen).

Ähnlich verfährt man bei der Erstellung der Makros *Suchen.Auswahl_M-ID* bzw. *Suchen.Weitersuchen*.

5.5.3 Weitere Hinweise

1. In der Datenbank *Verein6a* wird im Formular *F Mannschaften: Auswahl HF* mit einem kleinen „Trick“ gearbeitet: Damit die beiden verwendeten Unterformulare mit dem Hauptformular verknüpft werden können, braucht man im Hauptformular das Verknüpfungsfeld *Mannsch-ID*. Dieses muss aber nicht sichtbar sein, da die Auswahl der Mannschaftsbezeichnung und der damit verbundenen *Mannsch-ID* über das ungebundene Kombinationsfeld mit Namen *Auswahl_Mannsch-ID* erfolgt und angezeigt wird. Also kann man das Feld *Mannsch-ID* hinter dem Unterformular für die Trainer verstecken. Der Inhalt des Feldes *Mannsch-ID* wird mit Hilfe des Makros *Suchen.Mannsch-ID* aus dem Kombinationsfeld entnommen und in das Steuerelement *Mannsch-ID* übertragen. Dieses Makro wird beim Aktualisieren des Kombinationsfeldes aktiviert.
2. Ein anderer „Trick“ wird bei der für den Bericht *Mannschaften: Auswahl bestimmter Mannschaft* zugrundeliegenden Abfrage *Mannschaften: Auswahl für Bericht* benutzt. Die für den Bericht benötigte Mannschaftsbezeichnung wird im Formular *F Mannschaften: Auswahl HF* bestimmt, anschließend kann der Bericht geöffnet werden. In der Abfrage *Mannschaften: Auswahl für Bericht* wird als Kriterium für die Mannsch-ID „Wie [Formulare]![F Mannschaften: Auswahl HF]![Mannsch-ID]“ eingetragen. Dies bewirkt, dass bei Öffnen des Berichts in der zugrundeliegenden Abfrage die aus dem Formular gewählte *Mannsch-ID* als Kriterium festgelegt wird, und somit nur diese Mannschaft im Bericht angezeigt wird.



Damit das *Startmenü* beim Öffnen der Datenbank immer angezeigt wird, erstellt man ein Makro mit dem Namen *Autoexec*; dieses Makro enthält als Aktion *ÖffnenFormular* (Formularname ist *Startmenü*). Das Startmenü für die Datenbank *Verein6* ist unten angezeigt; das Menü für *Verein4* ist entsprechend.