

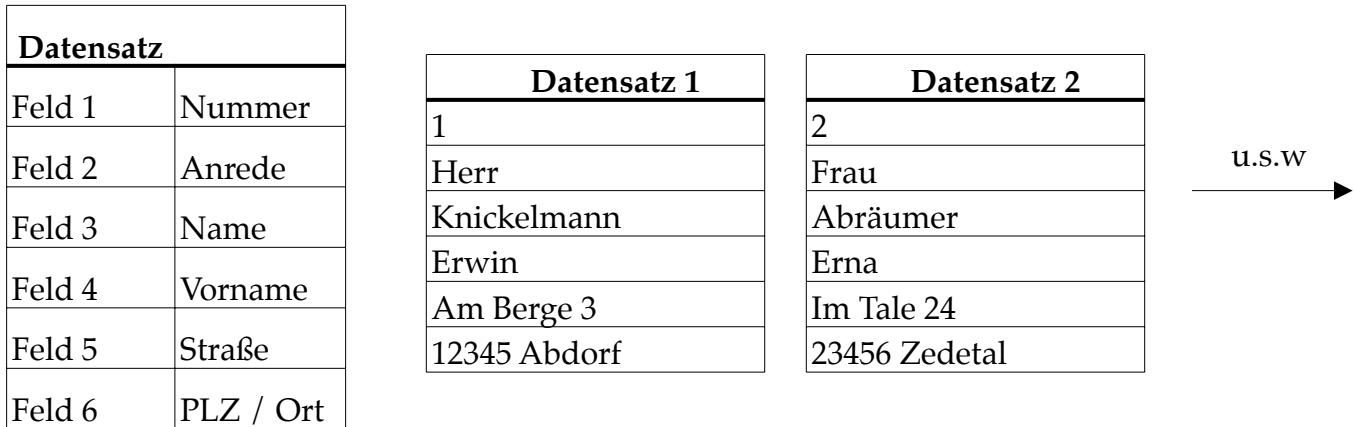
Relationale Datenbanken - Entity Relationship Modell - mySQL

Relationale Datenbanken - eine Einführung	2
Relationen, Attribute , Schlüssel.....	3
Beispiel "Schachturnier"	5
Normalformen nach Codd	6
Relationsalgebra.....	11
Join, Project und Select.....	12
Entity Relationship Modell.....	14
Einführung und Beispiel.....	15
Noch einmal "Schachturnier"	16
Beispiel "Schule"	17
Aufgaben	18
mySQL	21
MySql und ERM.....	21
Wichtige Hinweise zu Änderungen 2002 auf 2006	22
MySql installieren und starten.....	23
Die Teile von MySql	25
Datenbank anlegen.....	28
Tabellen anlegen.....	30
Daten aus Textdateien importieren	33
Daten anschauen.....	36
Auswählen mit Select.....	37
Übungen	39

Sie finden diese Inhalte in 2 PDF-Dateien (rbanken1.pdf und rbanken2.pdf). Zusätzlich finden sie noch eine Filemaker-Einführung, eine Einführung in PHP von Herrn Meyer und die Textdateien für den 2. Teil dieses Kapitels. Beispiele und Aufgaben für mySQL - Abfragen sind außerdem in der Datei rbanken3.pdf zu finden.

Datenbanken, mit denen wir es privat zu tun bekommen, sind in den meisten Fällen lineare Datenbanken, in denen pro Datensatz immer die gleichen Informationen abgespeichert sind.

Ein einfaches Beispiel dafür ist eine Adressendatei, in der jeder Datensatz aus 5 Feldern besteht.



Struktur des Datensatzes Alle Datensätze hintereinander (sequentiell) gespeichert.

In der Programmiersprache Pascal hätte man so einen Datensatz als RECORD definiert und dann eine Datei aus diesen Records gebildet. In Java bietet sich offensichtlich an, einen Datensatz als Instanz einer entsprechenden Klasse zu definieren, wobei sich möglicherweise zusätzlich einige Methoden zum Umgang mit diesen Instanzen in die Klassendefinition integrieren lassen. Diese programmiertechnischen Details stehen aber jetzt nicht zur Debatte.

Alle Datensätze lassen sich als Tabelle darstellen: An dieser Tabelle lassen sich einige wichtige Begriffe verdeutlichen:

Nummer	Anrede	Name	Vorname	Straße	PLZ/Ort
1	Herr	Knickelmann	Erwin	Am Berge 3	12345 Abdorf
2	Frau	Abräumer	Erna	Im Tale 24	23456 Zedetal
3	Herr	Meier	Helmut	Am Rande 34	34567 Ddorf
4	Frau	Müller	Elvira	Schildweg 2	45678 Fstadt
5	Herr	Knispel	Karl	Am Schloss 1	56789 Ettal
6	Frau	Schmidt	Karin	Talweg 9	67890 Kstadt

- Die Tabelle heißt **Relation** und bekommt einen Namen - hier z.B. den Namen "Adresse"
- Die Spaltenüberschriften heißen **Attribute** der Relation. Diese Attribute müssen eindeutig sein, d.h., es darf nicht zwei Attribute gleichen Namens geben.
- Jedes Attribut hat einen bestimmten Wertebereich z.B. Zahl, String, Datum etc.
- Jede Zelle in der Tabelle darf nur einen Wert des Attribut enthalten.
- Zwei Zeilen in der Relation unterscheiden sich mindestens in einem Attributwert.
- Alle Attributwerte in einer Zeile der Relation heißen **Tupel**. Im obigen Beispiel enthält die Relation also 6-Tupel. Ein Tupel entspricht einem Datensatz.

Eine Relation muss ein Attribut haben, mit dem ein Tupel eindeutig identifiziert werden kann. Dieses Attribut heißt **Schlüssel**. In unserem Beispiel könnte auch *PLZ/Ort* oder *Straße* als Schlüssel verwendet werden. Da aber in einer neuen Eintragung in der Datei möglicherweise eine schon vorhandene Anschrift oder ein schon vorhandener Wert für PLZ/Ort noch mal vorkommt, eignen sich diese Attribute nicht. Der Schlüssel, der zur Identifikation der Tupel verwendet wird heißt auch **Primärschlüssel**.

Nummer	Anrede	Name	Vorname	Straße	PLZ/Ort
1	Herr	Knickelmann	Erwin	Am Berge 3	12345 Abdorf
2	Frau	Abräumer	Erna	Im Tale 24	23456 Zedetal
3	Herr	Meier	Helmut	Am Rande 34	34567 Ddorf
4	Frau	Müller	Elvira	Schildweg 2	45678 Fstadt
5	Herr	Knispel	Karl	Am Schloss 1	56789 Ettal
6	Frau	Schmidt	Karin	Talweg 9	67890 Kstadt

Relation "Adresse"

Bis hierher ist nun noch nichts Besonderes an dieser Tabellendarstellung zu entdecken. Die Forderung nach eindeutigen Tupeln scheint sogar nachrangig zu sein, denn wen stört es schon, wenn eine Adresse doppelt oder dreifach in der Tabelle vorkommt, wichtig ist doch nur, dass man sie findet.

Das diese recht einfache Art der Darstellung von Daten schnell an ihre Grenzen stößt, machen einige kleine Beispiele deutlich:

Es gibt Menschen, die ihre Platten (oder CDs) in einer Datenbank verwalten. (Das ist zwar für den Normalmusikhörer großer Unfug, denn ein echter Musikfreund weiß auswendig, was auf seinen Platten ist und eine elektronische Verwaltung nützt nichts, wenn das Plattenregal hoffnungslos unsortiert ist.)

Dann stellt sich schnell das Problem, nach welchem Schlüssel die CDs erfaßt werden sollen. Soll der Schlüssel *Interpret /Band* sein und die Attribute sind die einzelnen *Stücke/Songs* mit ihrer Länge ? Was ist dann aber mit Samplern, auf denen diverse Interpreten versammelt sind ? Sollen möglicherweise alle *Stücke / Songs* auf allen Platten/CDs einzeln erfaßt werden ?

Dann müßte man als Schlüssel *Songtitel + Interpret* verwenden, da ja nicht nur Songs von Bob Dylan von vielen anderen Musikern auch veröffentlicht worden sind.

Wie bekommt man dann aber alle Stücke einer CD aus dieser Datenbank heraus ?
oder:

Die Bundesligatabelle aus der Sonntagszeitung ist eine der beliebtesten Lektüren des Fußballfans. Aus ihr sind aber die vergangenen Ergebnisse einzelner Spiele nicht mehr zu erkennen. Es ist nicht möglich, herauszulesen, wieviele Heimspiele Schalke in der Saison verloren hat. Es wäre auch gar nicht möglich, in dieser Tabelle einzelne Spiele darzustellen. Es wird aber sicher eine Datei geben, in der die einzelnen Spiele erfaßt sind. Wie muss diese Datei aussehen ? Wie kann man daraus eine Ergebnistabelle ermitteln ?

Damit die Problemstellungen nicht gleich zu komplex werden, zuerst ein noch übersichtliches Beispiel. Es stammt aus :

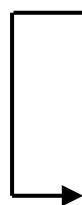
" *Eckart Modrow, Dateien - Datenbanken - Datenschutz, Dümmler - Verlag , 1996*" .

Es zeigt übersichtlich das Entstehen einer relationalen Datenbank, wobei Probleme, wie die oben geschilderten, zu lösen sind.

Dabei kann man auf zwei Wegen vorgehen. Es ist möglich zuerst eine Planung mit dem *Entity Relationship Modell* vorzuschicken und daraus dann Relationen abzuleiten. Eine andere Möglichkeit besteht darin, ausgehend von einer komplexen Anfangstabelle schrittweise geeignete Relationen zu entwickeln. Beide Wege werden hier kurz beschrieben. Die historisch ältere Vorgehensweise ist die zweite, mit der wir auch beginnen wollen.

Bei einem lokalen Schachturnier treten 4 Spieler aus unterschiedlichen Vereinen gegeneinander an. Jeweils unter der Leitung eines Schiedsrichters spielt jeder gegen jeden, was 6 Spiele bedeutet. Das ganze Turnier lässt sich als eine Tabelle darstellen:

Spiel_Nr.	Leiter	Leiter Verein	Leiter Vereinsort	Datum	Spieler 1	Spieler 1 Verein	Spieler 1 Vereinsort	Spieler 1 Punkte
1	Meier	1.SC Matt	Göttingen	07.02.2006	Ander s	wei ße Dame	Bonn	2
2	Müller	Glückauf	Bottr op	07.02.2006	Cr ay	1.SC Matt	Göttingen	1
3	Meier	1.SC Matt	Göttingen	09.02.2006	Ander s	wei ße Dame	Bonn	2
4	Müller	Glückauf	Bottr op	09.02.2006	Ber tr am	Bauer 09	Mai nz	0
5	Meier	1.SC Matt	Göttingen	12.02.2006	Ander s	wei ße Dame	Bonn	1
6	Paul s	Bauer 09	Mai nz	12.02.2006	Ber tr am	Bauer 09	Mai nz	2



Spieler 2	Spieler 2 Verein	Spieler 2 Vereinsort	Spieler 2 Punkte
Ber tr am	Bauer 09	Mai nz	0
Dör i ng	1.SC Matt	Göttingen	1
Cr ay	1.SC Matt	Göttingen	0
Dör i ng	1.SC Matt	Göttingen	2
Dör i ng	1.SC Matt	Göttingen	1
Cr ay	1.SC Matt	Göttingen	0

Auch wenn diese Tabelle alle Informationen enthält, ist sie für einige Abfragen ziemlich schlecht geeignet. Will man etwa wissen, welche Spieler unter der Leitung des Schiedsrichters aus Göttingen mehr als 1 Punkt erreicht haben, dann muss man schon sehr genau hinschauen, um diese Information zu ermitteln. Stellt man sich dann vor, dass eine komplette Tabelle z.B. aller Bundesligaspiele auch so aussieht, dann wird einzusehen sein, dass diese Tabelle bearbeitet werden muss.

Zuerst fällt auf, dass die Tabelle sehr viele redundante Informationen enthält. Allein 9 mal ist zu sehen, dass der 1.SC Matt aus Göttingen kommt (oben farbig / grau unterlegt).

Auf dem Weg zu einer relationalen Datenbank gibt es einen "vorgeschriebenen" Weg, der von E.F.Codd Anfang der 70er Jahre entwickelt wurde. Die große Tabelle wird dabei schrittweise in kleine Tabellen zerlegt und dabei werden Beziehungen zwischen diesen Teiltabellen festgelegt. Dieses Verfahren heißt *Normalisierung*. Dabei werden Tabellen in unterschiedliche *Normalformen* gebracht. Diese Normalformen entsprechen den einzelnen Schritten des Verfahrens.

Die **1.Normalform** hat eine Tabelle, wenn ihre Attributwerte **atomar** sind, das bedeutet, dass sie nicht weiter zerlegt werden können.

In unserem Beispiel ist das der Fall, da es z.B. vollkommen sinnlos wäre, etwa den Vereinsnamen noch weiter zu zerlegen. Auch bei Adressen müsste man sich sehr genau überlegen, ob Postleitzahl und Ort getrennt erfasst werden sollen oder ob PLZ/Ort schon als atomar angesehen wird.

Die **2.Normalform** nach Codd hat eine Tabelle, wenn sie in der 1.Normalform ist und zusätzlich alle Attribute, die nicht zum Schlüssel gehören voll funktional vom gesamten Schlüssel und nicht nur von einem Teil des Schlüssels abhängig sind.

Das ist in der Tabelle offensichtlich der Fall, da sie in der Anfangsform einen Schlüssel hat, die **Spiel Nr.**, der nur aus einem Attribut besteht.

In der Tabelle sieht man aber, dass z.B. *Leiter_Verein* und *Leiter_Vereinsort* oder *Spieler1_Verein* und *Spieler1_Vereinsort* von einander abhängig sind, obwohl keines dieser Attribute zum Schlüssel gehört. Das Attribut Vereinsort ist vom Verein abhängig.

Die **3.Normalform** nach Codd hat eine Tabelle, wenn sie in der 2.Normalform ist und zusätzlich keine transitiven Abhängigkeiten mehr bestehen. Das heißt, dass eine Attribut nicht über ein anderes Attribut vom Schlüssel abhängig ist, oder Attribute, die nicht zum Schlüssel gehören, voneinander abhängig sind.

<u>Leiter</u>	Verein	Ort
Meier	1.SC Matt	Göttlingen
Müller	Glückauf	Bottr op
Meier	1.SC Matt	Göttlingen
Müller	Glückauf	Bottr op
Meier	1.SC Matt	Göttlingen
Pauls	Bauer 09	Mainz

Relation LEITUNG

<u>Leiter</u>	Verein	Ort
Meier	1.SC Matt	Göttlingen
Müller	Glückauf	Bottr op
Pauls	Bauer 09	Mainz

Um die 3.Normalform zu erhalten, muss die Tabelle nun zerlegt werden. Zuerst ziehen wir die Attribute *Leiter* , *Leiter_Verein* und *Leiter_Vereinsort* aus der Tabelle heraus und machen sie zur Relation **Leitung** mit dem Primärschlüssel **Leiter** . Dabei können bereits einige Tupel heraus genommen werden, da sie doppelt vorkommen.

Diese Relation befindet sich dann noch nicht in der 3. Normalform, da das Attribut *Ort* über

Verein, (die beide nicht zum Schlüssel gehören) abhängig vom Schlüssel *Leiter* ist.

Wir werden diese Tabelle gleich noch weiter zerlegen müssen.

Zuerst aber ziehen wir die Attribute *Spiel_Nr.*, *Leiter* und *Datum* aus der Anfangstabelle heraus und nennen die neue Relation **Turnier** mit dem Primärschlüssel **Spiel_Nr.** und dem Fremdschlüssel **Leiter**.

Dabei bedeutet Fremdschlüssel, dass ein Attribut in einer anderen Relation Primärschlüssel ist (in unserem Falle die oben erzeugte Relation LEITUNG). Hier gibt es keine doppelten Tupel, es kann also auch nichts gestrichen werden. Diese Relation befindet sich schon in der 3. Normalform, da die Attribute *Leiter* und *Datum* nicht funktional voneinander abhängig sind und auch keine transitive Abhängigkeit zum Schlüssel besteht.

Aus der ursprüngliche Tabelle bleiben nun noch die Attribute übrig, die sich auf die

Relation TURNIER

<u>Spiel_Nr.</u>	<u>Leiter</u>	<u>Datum</u>
1	Meier	07.02.2006
2	Müller	07.02.2006
3	Meier	09.02.2006
4	Müller	09.02.2006
5	Meier	12.02.2006
6	Pauls	12.02.2006

Relation Ergebnisse

<u>Spieler</u>	<u>Spiel_Nr.</u>	<u>Verein</u>	<u>Ort</u>	<u>Punkte</u>
Bertram	1	Bauer 09	Mainz	0
Döring	2	1.SC Matt	Göttlingen	1
Cr ay	3	1.SC Matt	Göttlingen	0
Döring	4	1.SC Matt	Göttlingen	2
Döring	5	1.SC Matt	Göttlingen	1
Cr ay	6	1.SC Matt	Göttlingen	0
Ander s	1	wei ße Dame	Bonn	2
Cr ay	2	1.SC Matt	Göttlingen	1
Ander s	3	wei ße Dame	Bonn	2
Bertram	4	Bauer 09	Mainz	0
Ander s	5	wei ße Dame	Bonn	1
Bertram	6	Bauer 09	Mainz	2

Spieler und deren Ergebnisse beziehen. Wir machen daraus die Relation ERGEBNISSE

In dieser Relation sieht man, dass die *Spielnummer* als Primärschlüssel allein nicht ausreicht. Es müssen schon *Spielnummer* und *Spieler* zusammen als Schlüssel benutzt werden. Da nun der

Schlüssel aus zwei Attributen besteht, muss man prüfen, ob die Tabelle in der 2. Normalform ist.

Diese Relation befindet sich noch nicht in der 2. Normalform, da wieder *Verein* und *Ort*, die beide nicht zum Schlüssel gehören, funktional voneinander abhängig sind. Außerdem ist z.B. das Attribut **Verein** von einem Teil des Schlüssels (nämlich **Spieler**) abhängig, aber nicht vom anderen Teil (**Spiel Nr.**). Hier muss also weiter zerlegt werden. Wenn man die Relationen LEITUNG und ERGEBNISSE weiter in Tabellen zerlegt, bekommt man zum Schluss folgendes Bild:

Relation LEITUNG

<u>Leiter</u>	Verein
Meier	1.SC Matt
Müller	Glückauf
Pauls	Bauer 09

Relation TURNIER

<u>Spiel Nr.</u>	Leiter	Datum
1	Meier	07.02.2006
2	Müller	07.02.2006
3	Meier	09.02.2006
4	Müller	09.02.2006
5	Meier	12.02.2006
6	Pauls	12.02.2006

Relation VEREINE

<u>Verein</u>	Ort
1.SC Matt	Göttlingen
Glückauf	Bottr op
wei ße Dame	Bonn
Bauer 09	Mainz

Relation ERGEBNISSE

<u>Spieler</u>	<u>Spiel Nr.</u>	Punkte
Bertram	1	0
Ander s	1	2
Döring	2	1
Cray	2	1
Cray	3	0
Ander s	3	2
Döring	4	2
Bertram	4	0
Döring	5	1
Ander s	5	1
Cray	6	0
Bertram	6	2

Relation SPIELER

<u>Spieler</u>	Verein
Ander s	wei ße Dame
Cray	1.SC Matt
Döring	1.SC Matt
Bertram	Bauer 09

Dabei liegt es nahe, nun eine Relation `VEREINE` und eine Relation `ERGEBNISSE` zu erzeugen. Damit hat man dann insgesamt 5 Relationen, die nun noch einmal daraufhin geprüft werden, ob sie der 2.Normalform genügen.

Das ist offensichtlich der Fall.

Diese Relationen befinden sich sogar in der 3.Normalform.

Nun haben wir die Anfangstabelle soweit zerlegt, dass daraus eine relationale Datenbank geworden ist. Vergleicht man diese fünf Relationen mit der Anfangstabelle fallen Unterschiede sofort auf. In der Anfangstabelle gab es 6 Zeilen mit je 13 Spalten und damit 78 Eintragungen. Jetzt gibt es in den Relationen zusammen 74 Eintragungen. Das ist nicht der große Gewinn, den man erwarten konnte, wenn man den zweiten wichtigen Unterschied sieht: es gibt in den Tabellen keine redundanten Informationen mehr.

Aus den Relationen kann man die Zeilen der Anfangstabelle leicht rekonstruieren.

Will man z.B. wissen, wer unter wessen Leitung im dritten Spiel mit welchem Ergebnis angetreten ist, wird man mit dem Schlüssel SpielNr und in der Relation `TURNIER` beginnen und sich ohne Probleme über Schlüssel und Fremdschlüssel durch die Relationen hangeln.

In der Übersicht auf der nächsten Seite ist der Weg durch die Tabellen durch Pfeile gekennzeichnet. Die Reihenfolge der Abarbeitung ist durch eine passende Nummerierung zu erkennen.

Relation TURNIER

<u>Spiel Nr.</u>	Leiter	Datum
1	Meier	07.02.2006
2	Müller	07.02.2006
3	Meier	09.02.2006
4	Müller	09.02.2006
5	Meier	12.02.2006
6	Pauls	12.02.2006

Relation LEITUNG

<u>Leiter</u>	Verein
Meier	1.SC Matt
Müller	Glückauf
Pauls	Bauer 09

Relation VEREINE

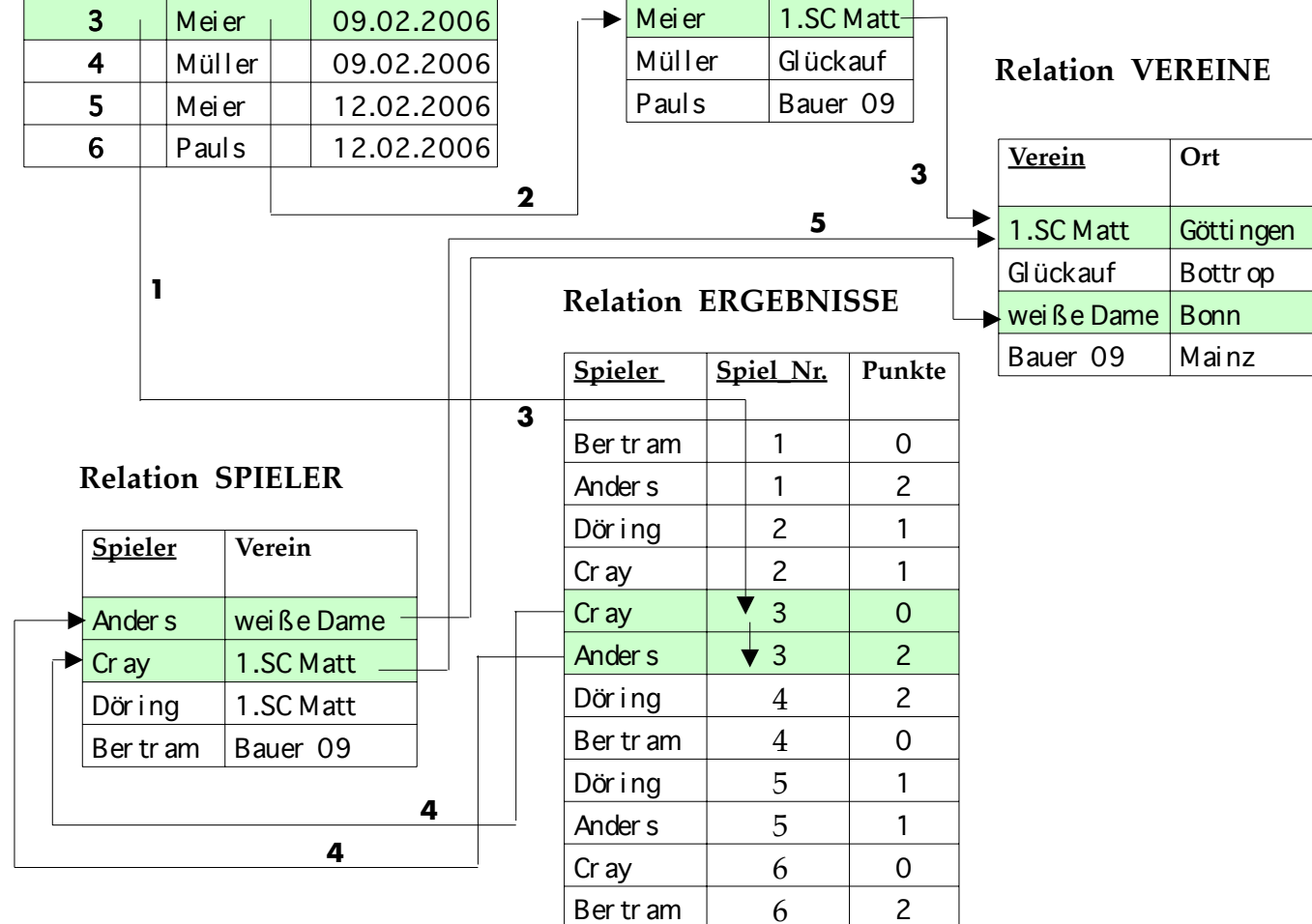
<u>Verein</u>	Ort
1.SC Matt	Göttingen
Glückauf	Bottr op
wei ße Dame	Bonn
Bauer 09	Mainz

Relation ERGEBNISSE

<u>Spieler</u>	<u>Spiel Nr.</u>	Punkte
Ber tr am	1	0
Ander s	1	2
Dör i ng	2	1
Cray	2	1
Cray	3	0
Ander s	3	2
Dör i ng	4	2
Ber tr am	4	0
Dör i ng	5	1
Ander s	5	1
Cray	6	0
Ber tr am	6	2

Relation SPIELER

<u>Spieler</u>	Verein
Ander s	wei ße Dame
Cray	1.SC Matt
Dör i ng	1.SC Matt
Ber tr am	Bauer 09



Das **3.** Spiel fand am **9.2.02** unter Leitung von **Meier** vom **1.SC Matt** aus **Göttingen** zwischen **Cray** vom **1. SC Matt** aus **Göttingen** und **Anders** vom Club **wei ße Dame** aus **Bonn** statt. Es endete mit **2 : 0** für Anders

Relationenalgebra

Nun wird es mathematisch. Man kann die Relationen als Mengen auffassen, deren Elemente die Tupel - also die Zeilen - sind. Zwischen Mengen kann man Operationen definieren und damit eine Relationenalgebra schaffen.

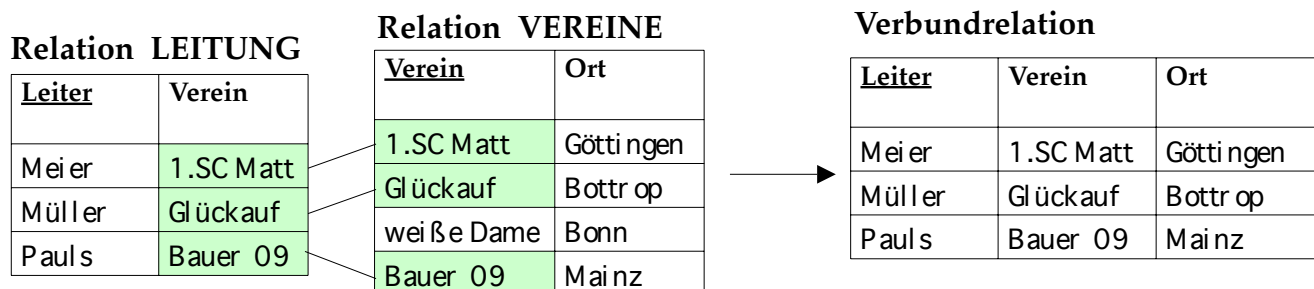
Die Operationen, die benutzt werden können sind: Vereinigung (\cup), Schnitt (\cap) und Differenz (\setminus). Die kann man natürlich nur benutzen, wenn die Relationen gleiche Attribute haben, denn sonst wären die Elemente (Tupel) mit diesen drei Operationen nicht "kompatibel".

Uns werden eher andere Operationen interessieren:

1. Der Verbund (JOIN), der zwei Relationen zu einer zusammenfügt, unter der Voraussetzung, dass die Relationen ein gemeinsames Attribut - den jeweiligen Primärschlüssel - haben. Dann werden die Tupel, die in diesem gemeinsamen Attribut übereinstimmen in der Verbundrelation zusammengefasst. Dazu ein Beispiel :

JOIN(LEITUNG,VEREINE,'Verein')

ergibt:



2. Die Projektion (PROJECT), die dazu dient, nicht benötigte Attribute aus einer Tabelle zu entfernen. Dazu werden alle die Attribute benannt, die erhalten bleiben sollen. Ein Beispiel:

PROJECT (TURNIER, 'SpielNr.', 'Leiter')

entfernt aus der Tabelle TURNIER das *Spieldatum*.

3. Die Auswahl (SELECT) dient dazu, aus einer Tabelle Tupel auszuwählen, die einer bestimmten Bedingung bezüglich eines Attributes genügen. Das ist sicher für Suchaktionen in großen Tabellen eine sehr wichtige Funktion. Dabei sind folgende Vergleichsoperatoren zulässig:

Operator	bedeutet
=	gleich
< >	ungleich
<	kleiner
>	größer
>=	größer oder gleich
<=	kleiner oder gleich

Die Wirkung der einzelnen Mengenoperationen wird erst deutlich, wenn das die Abfragen an die Tabellen etwas komplizierter werden. Im Folgenden Beispiel sollen die auftretenden Zwischenergebnisse - und das sind ja wieder Relationen - in einer flexiblen Tabelle, dem Arbeitsbereich vorgehalten werden. Dieser Bereich heißt - wie nicht an-

ders zu erwarten - **WORKSPACE (W)**.

Gesucht sind alle Spieler in unserem Schachturnier, die mindestens ein Spiel gewonnen haben mit Namen, Verein und Vereinsort.

Lösung:

1. Wir beginnen mit der Relation ERGEBNISSE und werden die Relation SPIELER damit verschmelzen, um auch die Vereine zu sehen.

W <----- JOIN(ERGEBNISSE,SPIELER,'Spieler')

W (Workspace)	<u>Spieler</u>	<u>Spiel_Nr.</u>	Punkte	Verein
	Bertram	1	0	Bauer 09
	Anders	1	2	weiße Dame
	Döring	2	1	1.SC Matt
	Cray	2	1	1.SC Matt
	Cray	3	0	1.SC Matt
	Anders	3	2	weiße Dame
	Döring	4	2	1.SC Matt
	Bertram	4	0	Bauer 09
	Döring	5	1	1.SC Matt
	Anders	5	1	weiße Dame
	Cray	6	0	1.SC Matt
	Bertram	6	2	Bauer 09

2. Der Vereinsort kann nun mit dem nächsten Verschmelzen hineingenommen werden:

W <----- JOIN(W,VEREINE,'Verein')

W (Workspace)	<u>Spieler</u>	<u>Spiel_Nr.</u>	Punkte	Verein	Ort
	Bertram	1	0	Bauer 09	Mainz
	Anders	1	2	weiße Dame	Bonn
	Döring	2	1	1.SC Matt	Göttingen
	Cray	2	1	1.SC Matt	Göttingen
	Cray	3	0	1.SC Matt	Göttingen
	Anders	3	2	weiße Dame	Bonn
	Döring	4	2	1.SC Matt	Göttingen
	Bertram	4	0	Bauer 09	Mainz
	Döring	5	1	1.SC Matt	Göttingen
	Anders	5	1	weiße Dame	Bonn
	Cray	6	0	1.SC Matt	Göttingen
	Bertram	6	2	Bauer 09	Mainz

3. Alle diejenigen, die mindestens ein Spiel gewonnen haben, müssen beim Attribut *Punkte* eine 2 stehen haben. Daher kann man nun die Tupel aus **W** herausziehen, die dieser Bedingung genügen:

$$W \leftarrow \text{SELECT}(W, 'Punkte' = 2)$$

Man erhält dann:

W (Workspace)	<u>Spieler</u>	<u>Spiel Nr.</u>	Punkte	Verein	Ort
	Ander s	1	2	weiße Dame	Bonn
	Ander s	3	2	weiße Dame	Bonn
	Döring	4	2	1.SC Matt	Göttingen
	Ber tram	6	2	Bauer 09	Ber tram

4. Nun brauchen wir aus dieser Tabelle aber das Attribut *Spiel Nr.* und das Attribut *Punkte* nicht. Wir holen daher alle anderen Attribute durch eine Projektion aus dem Workspace:

$$W \leftarrow \text{PROJECT}(W, 'Spieler', 'Verein', 'Ort')$$

Man erhält eine Relation, in der ein Tupel doppelt auftaucht. Es wird einmal automatisch gestrichen, so dass nur drei Tupel übrig bleiben.

W (Workspace)	<u>Spieler</u>	Verein	Ort
	Ander s	weiße Dame	Bonn
	Ander s	weiße Dame	Bonn
	Döring	1.SC Matt	Göttingen
	Ber tram	Bauer 09	Ber tram

Eine insgesamt recht mühsame Arbeit.

Ähnliche Arbeiten an relationalen Datenbanken lassen sich mit Datenbankabfragesprachen realisieren. Verbreitet ist SQL (**Structured Query Language**), mit der wir uns auch eingehender befassen wollen. SQL ermöglicht den Zugriff auf Datenstruktur und Daten in einer relationalen Datenbank. Man kann Tabellen einrichten, löschen, Relationen definieren und Datenabfragen durchführen.

Beispiele und Übungen dazu folgen später.

Zunächst aber - wie in der Einleitung angekündigt - eine andere Herangehensweise mit dem ER - Modell.

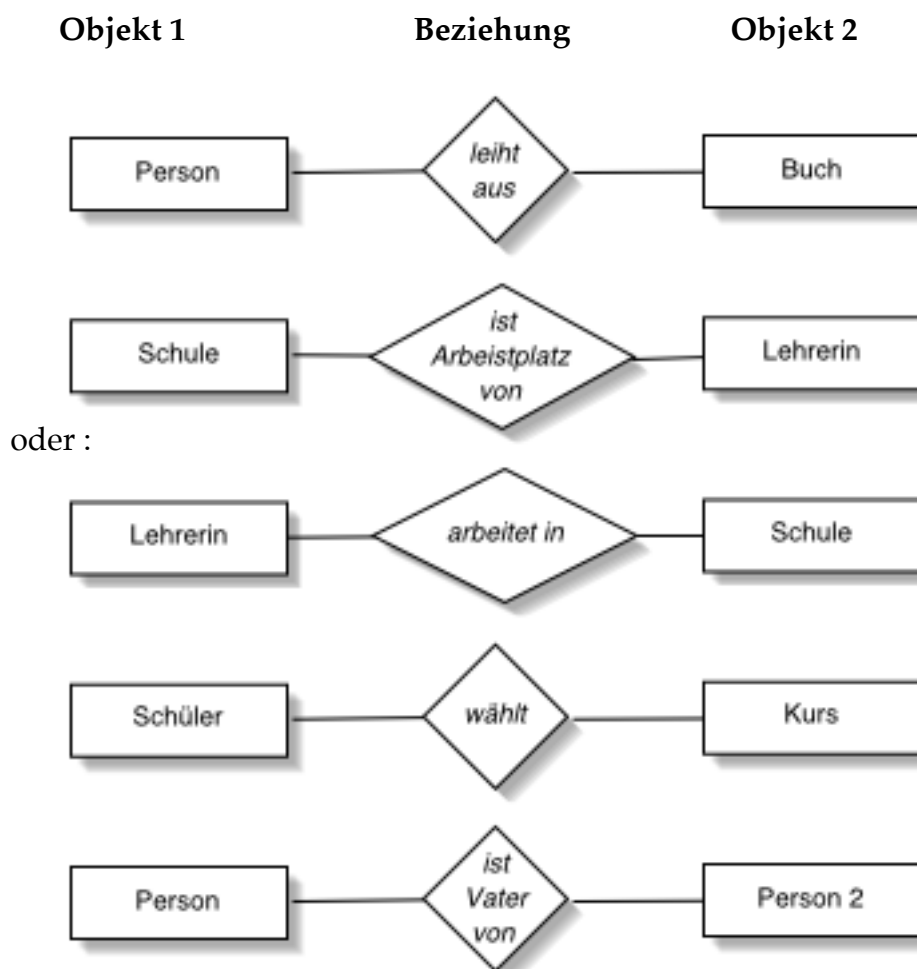
Das Entity - Relationship - Modell

Das oben erläuterte Verfahren zur Entwicklung einer relationalen Datenbank ging von einer schon vorhandenen (ungeeigneten) Tabelle aus. Sicher wäre es besser, schon vor Anlegen der ersten Tabelle einen Datenbankentwurf zu planen.

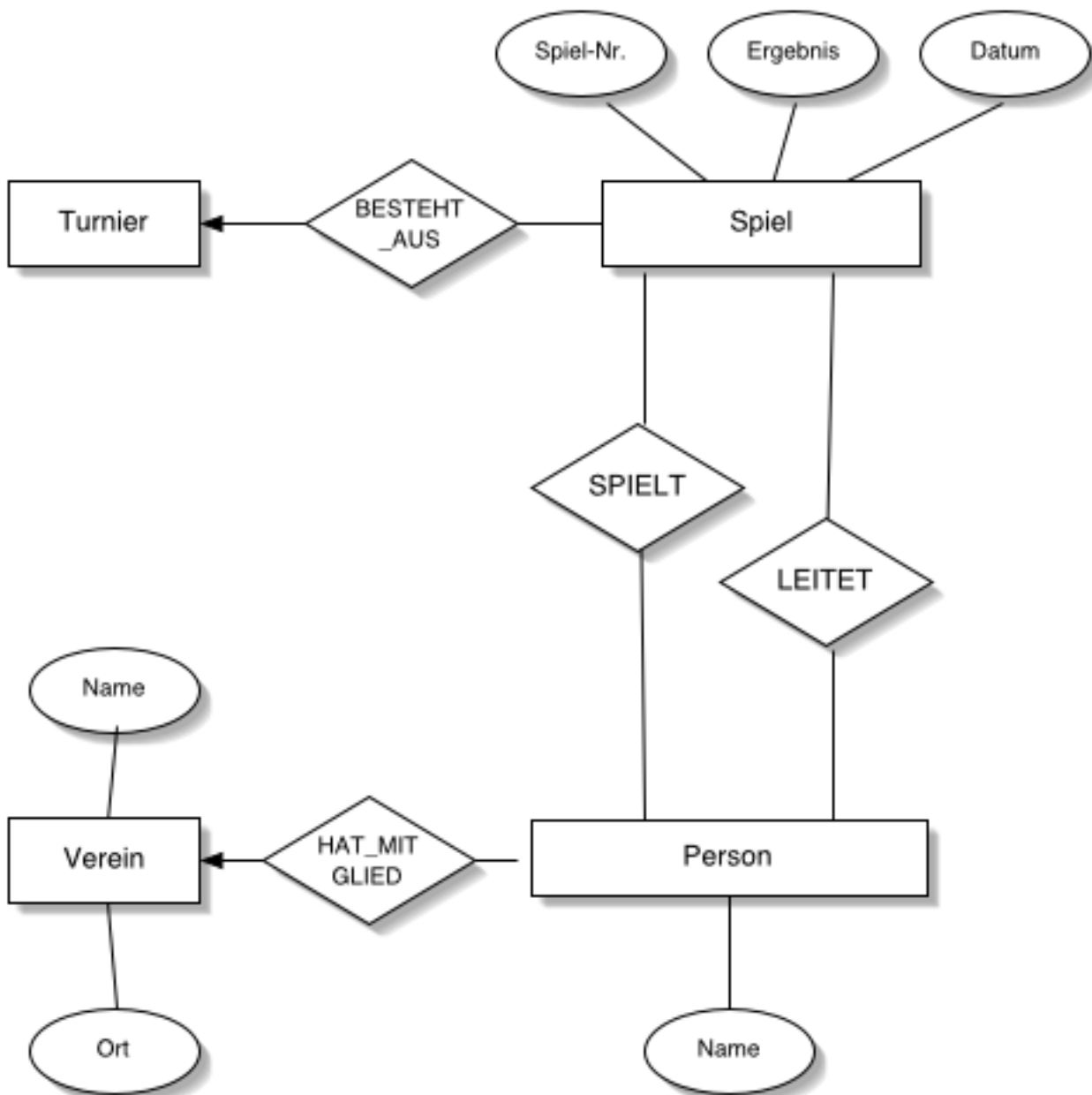
Bei der Planung geht man davon aus, dass es möglich ist, einen Ausschnitt aus der wirklichen Welt (Realwelt) in einem Datenmodell abzubilden. Dabei soll *Datenmodell* eine Sammlung aller Daten dieses Ausschnittes der Realwelt sein. Dabei ist nicht zwingend festgelegt, in welcher Form diese Daten vorliegen sollen.

Beschreibt man die Dinge der Realwelt als Objekte - auch **Entities** genannt - und Prozesse zwischen diesen Objekten als Beziehungen - **Relationen** genannt - , dann erhält man ein Netzwerk, in dem Objekte durch Beziehungen miteinander verknüpft sind.

Beispiele:



Das ERM ist nun das geeignete Hilfsmittel, um Datenmodelle zu entwickeln. Es ist eine Verfahren, den jeweiligen Zustand des Datenmodells bei seiner Entwicklung grafisch darzustellen. Die grafischen Symbole sind normiert. Reale oder abstrakte Objekte sind die **Entities** (Entity-Typen). Sie werden durch Rechtecke dargestellt. Die Eigenschaften der Objekte sind die **Attribute** und sie werden als Ovale dargestellt, die mit dem Objekt durch Linien verbunden sind. Die Beziehungen zwischen den Entities sind **Relationships** und werden durch Rauten dargestellt. Einfache Beispiele waren auf der vorangegangenen Seite zu sehen. Das Beispiel Schachturnier kann dann wie folgt aussehen:



Nun bleibt noch zu klären, wie man von dieser abstrakten grafischen Darstellung zu den Tabellen kommt, wie wir sie in der relationalen Datenbank kennengelernt haben.

Es liegt auf der Hand, dass jeder Entity - Typ einer Tabelle mit den entsprechenden Attributen entsprechen wird:

Relation SPIEL

<u>Spiel Nr.</u>	Datum	Ergebnis
1	07.02.2006	Ander s 2 / Ber tr am 0
2	07.02.2006	Cr ay 1 / Dör ing 1
3	09.02.2006	Ander s 2 / Cr ay 0
4	09.02.2006	Ber tr am 0 / Dör ing 2
5	12.02.2006	Ander s 1 / Dör ing 1
6	12.02.2006	Ber tr am 2 / Cr ay 0

Relation VEREIN

<u>Vereinsname</u>	Ort
1.SC Matt	Gött ingen
Glückauf	Bottr op
wei ße Dame	Bonn
Bauer 09	Mainz

Relation PERSON

<u>Name</u>
Ander s
Cr ay
Dör ing
Ber tr am
Mei er
Müller
Pauls

In einem weiteren Schritt werden jetzt auch die Beziehungen in Tabellen abgebildet:

Relation SPIELT

<u>Spiel Nr.</u>	Name
1	Ber tr am
1	Ander s
2	Dör ing
2	Cr ay
3	Cr ay
3	Ander s
4	Dör ing
4	Ber tr am
5	Dör ing
5	Ander s
6	Cr ay
6	Ber tr am

Relation LEITET

<u>Spiel Nr.</u>	Leiter
1	Mei er
2	Müller
3	Mei er
4	Müller
5	Mei er
6	Pauls

Relation HAT_MITGLIED

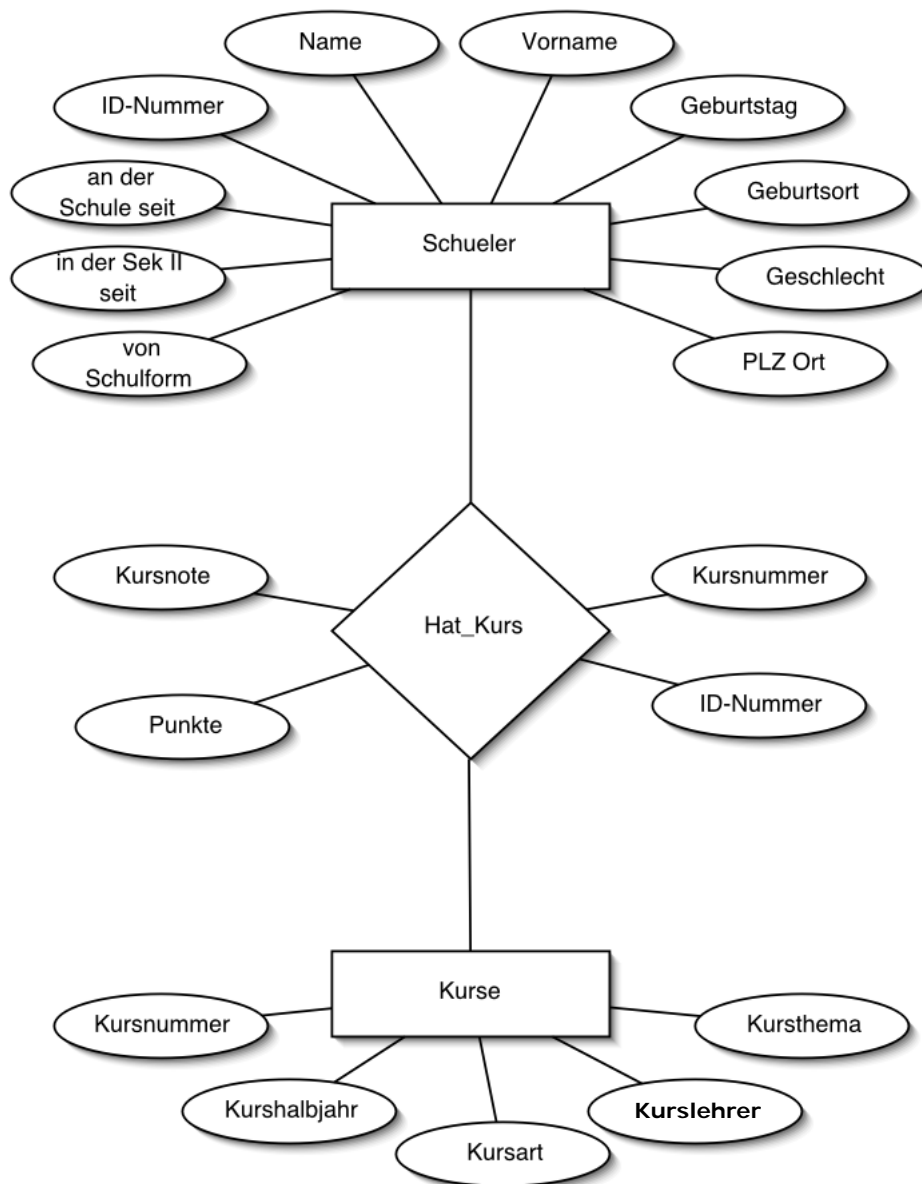
<u>Vereinsname</u>	<u>Name</u>
wei ße Dame	Ander s
1.SC Matt	Cr ay
1.SC Matt	Dör ing
Bauer 09	Ber tr am
1.SC Matt	Mei er
Glückauf	Müller
Bauer 09	Pauls

TURNIER und BESTEHT_AUS können wir hier weglassen , da es keine Attribute gibt, die wir

kennen müssen oder die für das Datenmodell von Interesse wären. Diese Tabellen müssen nun wieder - wie gehabt - normalisiert werden. Man sieht z.B., dass die Relation SPIEL im Attribut *Ergebnis* noch nicht atomar ist. Eine weitere Zerlegung der Tabellen würde zu einem ähnlichen Gesamtergebnis führen, wie wir es schon kennen.

Ein weitere Beispiel:

In der reformierten Oberstufe wird der Unterricht im 12. und 13. Jahrgang im Kurssystem organisiert. Schülerinnen und Schüler wählen Leistungs - und Grundkurse, bekommen in diesen



Kursen Noten und daraus wird dann das Studienbuch und am Ende das Abiturzeugnis.

Lehrerinnen und Lehrer unterrichten diese Kurse und bekommen Schülerlisten. Am Ende des Halbjahres geben sie diese Listen mit den Noten für jede Schülerin und jeden Schüler ab. Ihre Kurse haben eine Nummer oder Chiffre und ein Thema.

Man kann *Schueler* und *Kurse* als Objekt auffassen, wie sie in der obigen Grafik zu sehen sind. Dann kann man zwischen diesen Objekten verschiedene Beziehungen (oder auch nur eine) definieren.

Dabei handelt es sich bei der Beziehung *Hat_Kurs* um eine 1: n - Beziehung, da eine Schülerin / ein Schüler mehrere (n) Kurse besucht.

Relation SCHUELER

<u>ID Nummer</u>	Name	Vorname	akt. KHJ
1998- 001	Bei er	Er win	2
1998- 002	Kni ckel	Else	2
1998- 003	Mü ller	Kar l	2
1998- 004	Zeitler	Johanna	2



Relation HAT_KURS

<u>ID Nummer</u>	Kurs- nummer	Note
1998- 001	Dt 23	09
1998- 001	Ph 22	06
1998- 001	Ge 24	12
1998- 002	Dt 23	05
1998- 002	Ma 21	03
1998- 003	Ma 21	02
1998- 003	Ph 22	08
1998- 003	Ge 24	09
1998- 004	Dt 23	07



Betrachtet man die Beziehung von der Kursseite her, ist es genauso, da ein Kurs aus mehreren (m) Schülerinnen und Schülern besteht.

Ein wenig schwierig ist die Notengebung (das war ja schon immer so, warum sollte es im ERM besser sein).

Man kann die *Note* nicht als Attribut dem Kurs zuordnen, da es in einem Kurs ja für jede Teilnehmerin / jeden Teilnehmer eine andere Note geben kann. Man kann Note aber auch nicht als Attribut dem Objekt Schülerin / Schüler zuordnen, da es dafür ja unterschiedlich viele und verschiedene Noten geben wird. Da die Beziehung *Hat_Kurs* eine eindeutige Verbindung einer Schülerin/ eines Schülers zu einem Kurs darstellt, kann Note dort als Attribut untergebracht werden. Es wäre aber auch denkbar, eine weitere Beziehung zu erzeugen, die etwa BE-

KOMMT_NOTE heisst. Die könnte ebenfalls die Objekte *Schueler* und *Kurse* verbinden und dann als Attribut die Note enthalten. In dem obigen ER-Modell sind noch weitere Attribute angegeben. Die weiteren Tabellen könnten dann folgendermaßen aussehen:

Relation KURSE

<u>Kurs- nummer</u>	Lehrerin	Thema	Kursart
Dt 23	Kunkel	Literatur zwischen Wenn und Vielleicht	LK
Ma 21	Meier	Analysis ohne Gnade	GK
Ph 22	Schulz	Wellen	GK
Ge 24	Schmitt	Imperialismus	LK



Relation BESTEHT_AUS

<u>Kurs- nummer</u>	<u>ID Nummer</u>
Dt 23	1998 - 001
Dt 23	1998 - 002
Dt 23	1998 - 004
Dt 23	1998 - 007
....
Ma 21	1998 - 002
Ma 21	1998 - 003
Ma 21	1998 - 005
.....



Bei der genaueren Behandlung von SQL werden wir auf dieses Beispiel zurückkommen.

Aufgabe:

In vielen Radiosendern werden heute kaum noch Platten oder CDs auf - bzw. eingelegt. Die Songs kommen von der Festplatte. Sie sind dort in Audiodatenbanken abgelegt und stehen für einen sehr schnellen Zugriff bereit. Auch auf heimischen Computern (nicht nur von Schülerinnen und Schülern) kennt die Sammelwut für Musik im MP3 - Format keine Grenzen. Nun wird es sicher keine Notwendigkeit für Schülerinnen und Schüler geben , ihren Bestand an Songs in Form einer Datenbank selbst zu organisieren , es ist aber ein schönes Beispiel für die Komplexität eines Datenmodells.

Entwerfen sie mit ERM ein geeignetes Datenmodell. Die folgenden Informationen sollen sich verwalten lassen: Interpret / Band, Songtitel, Länge des Songs, Komponist, Erscheinungsjahr des Songs, Kennnummer der zug. CD, Titel der CD, die Songs auf einer CD.

Man soll herausfinden können, ob eine Band einen Song auf mehreren CDs veröffentlicht hat (das ist oft der Fall). Man soll herausfinden können, ob ein Song von mehreren Interpreten veröffentlicht wurde (das ist nicht nur bei Bob Dylan der Fall). Man soll alle Songs einer CD finden können auch wenn die CD ein Sampler mit verschiedenen Interpreten ist.