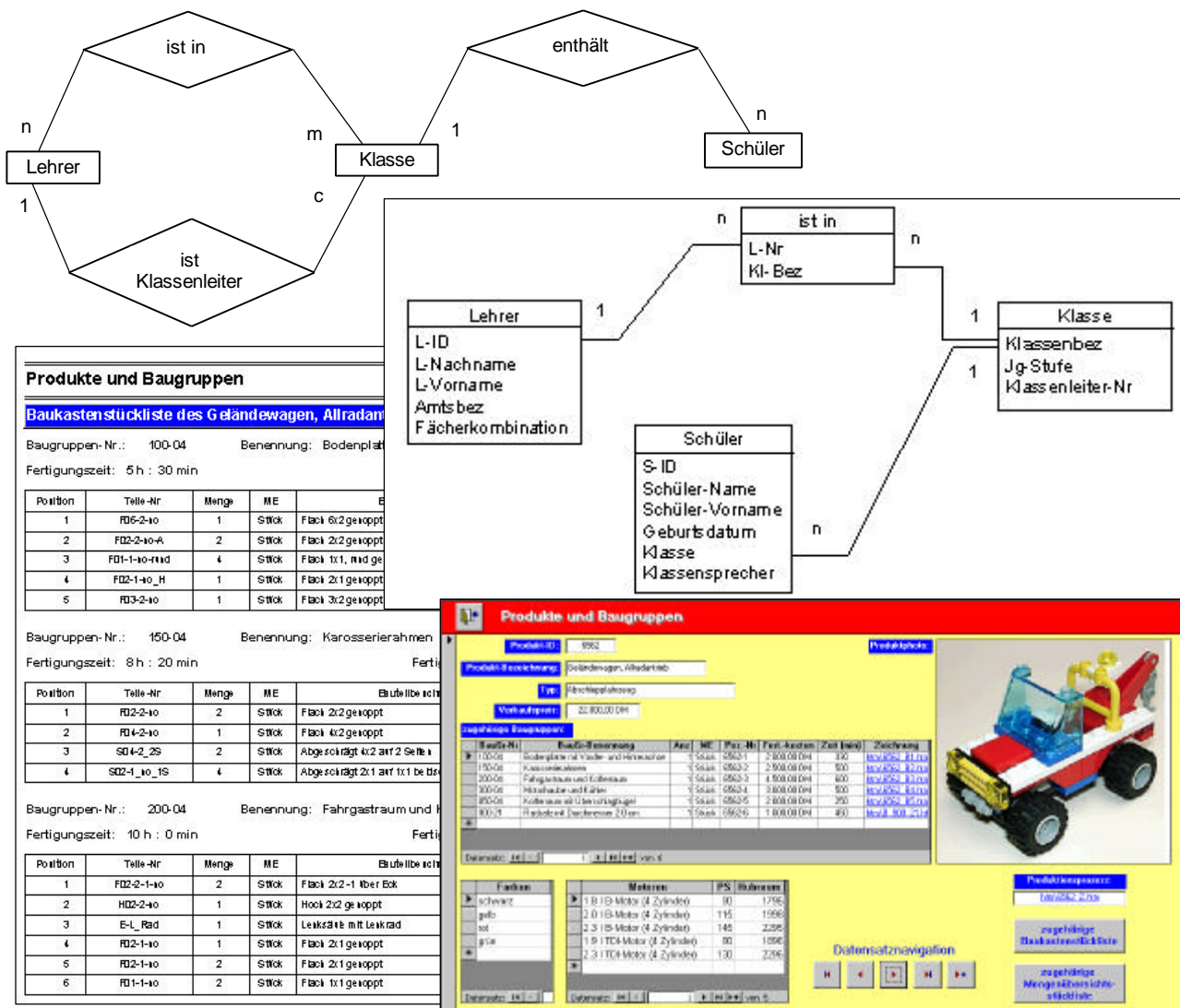


Arbeitskreis  
„Datenbanken im Unterricht“

# Datenbank



Zentralstelle für Computer im Unterricht  
Augsburg 2000



Arbeitskreis  
„Datenbanken im Unterricht“

# **Datenbank**

---

Zentralstelle für Computer im Unterricht  
Augsburg 2000

Erarbeitet im Auftrag  
des Bayerischen Staatsministeriums  
für Unterricht und Kultus

im Rahmen des Arbeitskreises  
„Datenbanken im Unterricht“  
(JP 1997 1.20)  
2. überarbeitete Auflage

*Leiter des Arbeitskreises:*

Hans-Wolfgang Matzke, Zentralstelle für Computer im Unterricht Augsburg

*Mitglieder des Arbeitskreises:*

Heinz Bayerlein,	Staatl. Fachoberschule Nürnberg
Konrad Kölbel,	Staatl. Fachoberschule Bayreuth
Dr. Werner Lorbeer,	Holbein-Gymnasium Augsburg
Albert Wiedemann,	Gisela-Gymnasium München
Paul Weishaupt,	Zentralstelle für Computer im Unterricht Augsburg

*Redaktion:*

Hans-Wolfgang Matzke, Zentralstelle für Computer im Unterricht Augsburg  
Jutta Niedermaier, Zentralstelle für Computer im Unterricht Augsburg

© Nachdruck - auch auszugsweise - nur mit Genehmigung der Zentralstelle für Computer im Unterricht Augsburg

---

Zentralstelle für Computer im Unterricht, Schertlinstr. 9, 86159 Augsburg  
Telefon (08 21) 2 59 19-0, Telefax (08 21) 2 59 19-19  
Internet <http://www.zs-augsburg.de>

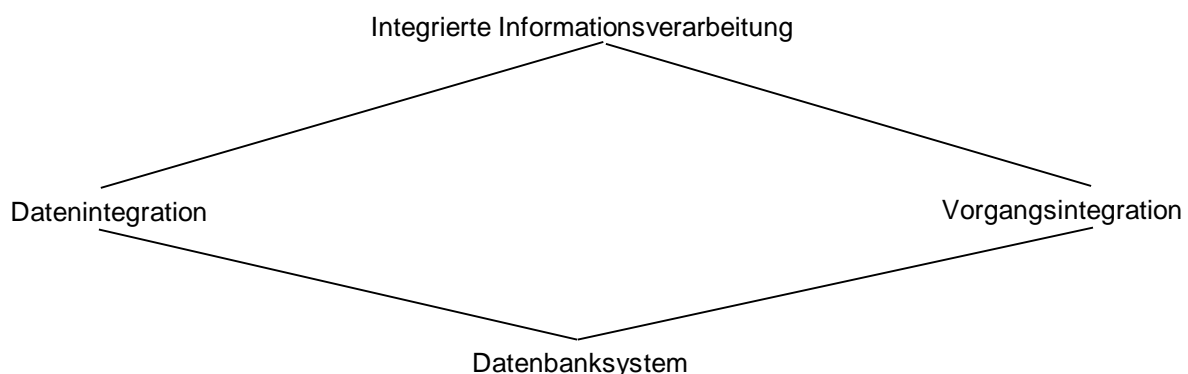
## Vorwort

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).

Eine wichtige Voraussetzung für die Realisation der integrierten Informationsverarbeitung sind geeignete *Datenbankverwaltungssysteme*, denen bestimmte logische Datenmodelle zugrunde liegen. Erst das *relationale Datenmodell* schaffte einen technischen Durchbruch. Im Mittelpunkt dieser Arbeit stehen *relationale Datenbanksysteme*, da Standards für objektorientierte und objektrelationale Datenbankmodelle noch nicht existieren.

Zu Beginn des Arbeitskreises (Herbst 1995) wurden 4 verschiedene Windows-Datenbanken, von denen Schullizenzen erhältlich sind, getestet. Der Arbeitskreis entschied sich aus mehreren Gründen einstimmig dafür, die erstellten Beispiele mit dem Programm MS-Access zu entwickeln. Die Konzepte sind jedoch auch mit anderen Programmen realisierbar.

Mit dem im Unterricht verwendeten Produkt

- soll das relationale Modell umgesetzt werden können,
- soll Multiuserbetrieb im Schulnetz realisiert werden können,
- soll SQL basierter Zugriff auf die Datenbank möglich sein.

Viele der auf dem Markt angebotenen Datenbankprogramme erfüllen diese Kriterien nicht.

Die Produktauswahl erfordert im Gegensatz zu Textverarbeitung und Tabellenkalkulation wesentlich größere Sorgfalt und Sachkenntnis, weil die oben genannten Kriterien nur bei detaillierter Betrachtung verifiziert werden können.

Von der Programmernomonomie sind klar gegliederte Bedienoberflächen und kurze Einarbeitungszeiten zwingend erforderlich, damit sich der Schüler auf die wesentlichen Inhalte konzentrieren kann.

Für das Einführungsbeispiel im Lehrerskript wurde ein vereinfacht dargestelltes Thema (*Lehrer und Klassen*) gewählt, das die folgenden Gesichtspunkte berücksichtigt:

- die Struktur ist jedem Lehrer der betroffenen Schulart (Realschule, Gymnasium, Berufliche Schule) vertraut,
- die Struktur beinhaltet alle für eine datenbanktechnische Betrachtung notwendigen Beziehungen (1-1, 1-n und n-m),
- das Thema ist unabhängig von technischen oder wirtschaftlichen Betrachtungsweisen.

Obwohl das gewählte Beispiel (*Lehrer und Klassen*) alle diese Bedingungen erfüllt, bleibt es doch überschaubar (4 Tabellen) und somit als Einführungsbeispiel geeignet. Dies gilt nicht mehr für einige der auf der CD mitgelieferten Beispiele. Deshalb sollte der Lehrer für den Unterricht ein Einführungsbeispiel wählen, das keinesfalls komplexer als das Beispiel *Lehrer und Klassen* ist. Für den Einstieg genügt es, ein Thema zu wählen, dessen Struktur keine n-m-Beziehung aufweist.

Nichtsdestoweniger muss der Lehrer aber über die gesamte Problematik der komplexen Beziehungen Bescheid wissen, um dieses Problem sinnvoll in den Unterricht einzubauen oder zu vermeiden. Auch aus diesem Grund wird diese Problematik im Einführungsskriptum für den Lehrer ausführlich besprochen.

Auf der mitgelieferten CD befindet sich eine Menge an Beispielen aus den unterschiedlichsten Wissensgebieten (Technik, Wirtschaft, Hobby, Multimedia). Diese Beispiele sind für den Einsatz im Unterricht und als Anregung für die Entwicklung weiterer Datenbanken gedacht.

Mit Hilfe einer Datenbank können nicht nur sachbezogene Daten, sondern auch personenbezogene Daten verwaltet werden. Infolgedessen ist noch ein wesentlicher Gesichtspunkt bei der Arbeit mit Datenbanken zu beachten: der Datenschutz. Sobald in einem Rechner personenbezogene Daten verarbeitet werden, müssen die Vorschriften der Datenschutzgesetze Beachtung finden, z. B. Bayerisches Datenschutzgesetz, Bundesdatenschutzgesetz sowie BayEUG. Aus diesem Grund gehört zur Datenbankentwicklung auch ein Datenschutzkonzept.

Vielen Kollegen ist die grundsätzliche Problematik des Datenschutzes bewusst. Hingegen sind einige wesentliche Einzelheiten, die im Umgang mit personenbezogenen Daten beachtet werden sollten, oft nicht in vollem Umfang vertraut. Deshalb soll das gewählte Einführungsbeispiel auch dazu dienen, auf die für diesen Fall wichtigsten Richtlinien des Datenschutzes hinzuweisen. Sollten, wie im Einführungsbeispiel, die Daten von Lehrern und Schülern bearbeitet werden, müssen darüber hinaus die Richtlinien der Bekanntmachung des bayerischen Staatsministeriums für Unterricht, Kultus, Wissenschaft und Kunst vom 19. März 1996 Nr. III/8-III/4-L0572-1/41785 (KWMBI. I Nr. 9/1996) eingehalten werden. Diese Bekanntmachung enthält erläuternde Hinweise für die Schulen zum Vollzug des Bayerischen Datenschutzgesetzes. Der Datensatz des Einführungsbeispiels enthält daher bewusst keine sensiblen Daten, sondern nur allgemein öffentlich zugängliche Angaben. Eine Erweiterung des in dem Beispiel verwendeten Datensatzes ist nur im Hinblick auf die Einhaltung der obengenannten Vorschriften und Richtlinien zulässig. Deshalb empfiehlt sich generell bei der Verwendung personenbezogener Daten im Unterricht der Einsatz fiktiver Daten (wie in diesem Beispiel).

Bei der Verarbeitung der Lehrerdaten ist ggf. zusätzlich eine Abstimmung mit der Personalvertretung erforderlich.

Augsburg, im Juli 1997

Die Mitglieder des Arbeitskreises

## Vorwort zur 2. Auflage

Seit Erscheinen der 1. Auflage sind nun 3 Jahre vergangen und das Thema „Verwalten von großen Datenmengen mit Hilfe von Datenbanken“ ist aktueller als je zuvor. Hingewiesen sei an dieser Stelle u. a. auf die große Anzahl von Suchmaschinen, die die Informationsrecherche im Internet erleichtern sollen. Darüber hinaus bieten große Softwarehersteller inzwischen auch relationale Datenbankprogramme an, mit deren Hilfe Daten im Internet zur Verfügung gestellt werden können, so dass jeder Anwender von seinem Rechner aus Abfragen an die Datenbank richten kann, um die von ihm gewünschten Informationen zu erhalten.

Damit diese Informationsrecherchen für den Benutzer auch möglichst erfolgreich verlaufen, sind Kenntnisse über die Struktur und den Aufbau einer Datenbank sowie über die Arbeit mit einer Datenbank sehr hilfreich.

Diese grundlegenden Erkenntnisse haben sich in der Zwischenzeit in verschiedenen Schularten durchgesetzt und sind auch schon teilweise in den entsprechenden Lehrplänen verankert, zumal die Thematik schon in der 1. Auflage didaktisch so aufbereitet war, dass sie lehr- und lernbar wurde.

Nachdem die 1. Auflage sehr schnell vergriffen war, die Nachfrage sowohl nach der Theorie als auch nach weiteren Beispielen aber immer noch sehr groß ist, soll diesem Wunsch in einer 2. überarbeiteten Auflage entsprochen werden.

Dabei wurden folgende wesentliche Überarbeitungen bzw. Änderungen vorgenommen:

- Der Theorieteil bleibt bis auf eine Stelle nahezu unverändert; lediglich die Klassenleiter-Beziehung zwischen den Tabellen *Lehrer* und *Klassen* wurde aus bestimmten Gründen (vgl. Kapitel 3.3.2, Seite 30) im globalen Modell weggelassen.
- Die Struktur der Tabellen *Klassen* und *Schüler* wurde verändert. Die Schülertabelle wurde um das Feld *Geburtsdatum* ergänzt, damit Berechnungen durchgeführt werden können, außerdem wurde das Feld *Klassensprecher* als Ja/Nein-Feld in diese Tabelle aufgenommen, dadurch entfällt dieses Feld in der Tabelle *Klassen*. Das Feld *Ausbildungsrichtung* der Tabelle *Klassen* wurde durch das Feld *Jg-Stufe* ersetzt.
- In den Kapiteln 4.3.2.1 *Elementare Anfragen an die Datenbank* und 4.3.2.2 *Komplexe Anfragen an die Datenbank* wurden zusätzliche Beispiele eingefügt.
- Das Kapitel 4.4 *Formulare* wurde um den Abschnitt 4.4.5 *Beispiel für die Erstellung eines Formulars mit Unterformular* erweitert. Dies trägt dem Teil der benutzerfreundlichen Anwendungserstellung Rechnung und zeigt die Bedeutung des relationalen Modells auch bei der Erstellung von praktischen Anwendungen auf.
- Das Kapitel 4.5 *Berichte* wurde um den Abschnitt 4.5.1 *Beispiel für die Erstellung eines Berichts* erweitert.



- Die Beispiele auf der beigelegten CD wurden aktualisiert und erweitert. Insbesondere wurde der Modellierungsphase dadurch noch mehr Rechnung getragen, dass die Übungsaufgaben zur Erstellung von semantischen und logischen Modellen aus vorgegebenen Informationsstrukturen im Ordner *Uebungen* um zahlreiche Beispiele aus den verschiedensten Bereichen ergänzt wurden.
- Die Bezeichnung für nummerierte Primärschlüsselfelder haben aus methodischen Gründen die Endung „ID“ erhalten (z. B. *L-ID*), das zugehörige Fremdschlüsselfeld in der entsprechenden Tabelle hat dann die Endung „Nr“ (z. B. *L-Nr*). Dies ist nicht erforderlich, hat sich aber im Unterricht zur besseren sprachlichen Unterscheidung von Primär- und Fremdschlüssel bewährt.

Grundsätzlich bleibt noch anzumerken, dass diese Handreichung für den unterrichtenden Lehrer gedacht ist, und nicht für den Schüler. Dies betrifft insbesondere die Aufgaben- und Übungsteile auf der CD sowie einige Theorieteile, die ein tieferes Verständnis vermitteln, als für den normalen Unterricht vorgesehen ist.

Wie man selbst aus der eigenen Unterrichtserfahrung weiß, genügt es nicht, den Unterricht mit einem einzigen Beispiel zu bestreiten. Der unterrichtende Lehrer benötigt einen gewissen Erfahrungsvorsprung, u. a. auch für Übungsbeispiele und Leistungsnachweise.

Eine Datenbank ist in der Regel ein komplexes Gebilde und ein neues für den Unterricht geeignetes Beispiel mit allen Dokumentationen entsteht nicht an einem Tag. Aus diesem Grund sollten nicht alle Beispiele und Übungen, die sich auf der CD befinden, den Schülern frei zugänglich gemacht werden. Diese Beispiele dienen in erster Linie dem Lehrer zur Anregung und Entwicklung weiterer Modelle.

Die Beispiele sind auch für unterschiedliche Einsatzzwecke im Unterricht gedacht. Einige eignen sich zur Einführung in die Problematik, andere zur Anwendung (z. B. Vertiefung der Abfragetechnik), weitere sind als Ausblick auf mögliche Einsatzgebiete bzw. für das Verständnis komplexer Zusammenhänge konzipiert.

Augsburg, September 2000

H.-W. Matzke

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>11</b>
1.1	Ziele der Datenorganisation	11
1.2	Dateisysteme und Datenbanksysteme	12
1.3	Das Konzept des Datenbanksystems	14
<b>2</b>	<b>Aufbau und Arbeitsweise von Datenbanksystemen</b>	<b>17</b>
2.1	Das ANSI-Architekturmodell (3-Schichten-Architektur)	17
2.1.1	Konzeptionelles Schema	19
2.1.2	Internes Schema	19
2.1.3	Externes Schema	19
2.2	Arbeitsweise	20
<b>3</b>	<b>Datenmodellierung</b>	<b>21</b>
3.1	Informationsstruktur ermitteln	21
3.1.1	Ermittlung aufgrund von Realitätsbeobachtungen	22
3.1.2	Ermittlung aufgrund von Benutzersichtanalysen	23
3.1.3	Ermittlung aufgrund von Datenbestandsanalysen	23
3.2	Datenstruktur modellieren	23
3.2.1	Grundsätzliche Methoden	23
3.2.2	Elemente von Datenmodellen	24
3.2.2.1	Entitäten	24
3.2.2.2	Beziehungen zwischen Entitäten	25
3.2.2.3	Attribute	26
3.2.3	Umsetzung der Informationsstruktur in ein Entity-Relationship-Modell	28
3.3	Umsetzung des semantischen Datenmodells in ein logisches Datenbankmodell	29
3.3.1	Identifikationsschlüssel	30
3.3.2	Entwicklung des relationalen Datenbankschemas mit Hilfe von Abbildungsregeln	30
3.3.3	Entwicklung des relationalen Datenbankschemas mit Hilfe der Normalformen	37
3.4	Vertiefung des relationalen Datenbankmodells	43
3.4.1	Grundlegende Begriffe	43
3.4.2	Abhängigkeiten und Normalformen	44
3.4.2.1	Sinn und Zweck von Normalformen	44
3.4.2.2	Funktionale Abhängigkeiten	45
3.4.2.3	Transitive Abhängigkeit	48
3.4.3	Strukturelle Integritätsbedingungen	49
3.5	Zusammenfassung	51

<b>4</b>	<b>Realisierung des logischen Modells mit einem Softwareprodukt</b>	<b>53</b>
4.1	Tabellen	54
4.1.1	Planen von Tabellen	54
4.1.2	Datendefinition mit dem gewählten DBMS: Tabellen anlegen	55
4.1.3	Funktion eines Index	56
4.1.4	Verknüpfung von Tabellen	57
4.2	Datenmanipulation	58
4.3	Datenabfrage	59
4.3.1	SQL	60
4.3.2	QBE (Query By Example)	62
4.3.2.1	Elementare Anfragen an die Datenbank	63
4.3.2.2	Komplexe Anfragen an die Datenbank	64
4.3.3	Verschiedene Abfragearten in MS-ACCESS	67
4.4	Formulare	69
4.4.1	Ein Formular erstellen	70
4.4.2	Die Gestaltungsbereiche	71
4.4.3	Steuerelemente und deren Eigenschaften	71
4.4.4	Weitere Hinweise	72
4.4.5	Beispiel für die Erstellung eines Formulars mit Unterformular	73
4.4.5.1	Erstellung des Hauptformulars	74
4.4.5.2	Befehlsschaltflächen / Kombinationsfelder	75
4.4.5.3	Konstruktion und Einbau des Unterformulars	76
4.5	Berichte	79
4.5.1	Beispiel für die Erstellung eines Berichts	80
4.5.2	Zusammenfassung	83
4.6	Datenschutz und Datensicherheit	83
<b>5</b>	<b>Literaturverzeichnis</b>	<b>85</b>
<b>6</b>	<b>Anhang</b>	<b>87</b>
6.1	Datenschutz	87
6.2	Index	91



# 1 Einführung

## 1.1 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?

Siehe hierzu auch Vorwort und Anhang des Skriptums sowie Kapitel 3.2.3, Festlegung des Datenrahmens.

## 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**, vergleiche hierzu Bild 3-30). Daten, die redundant gespeichert sind, müssen dasselbe aussagen (Datenkonsistenz). Die Forderung nach Datensicherheit wird gelegentlich in die Datenintegrität einbezogen.

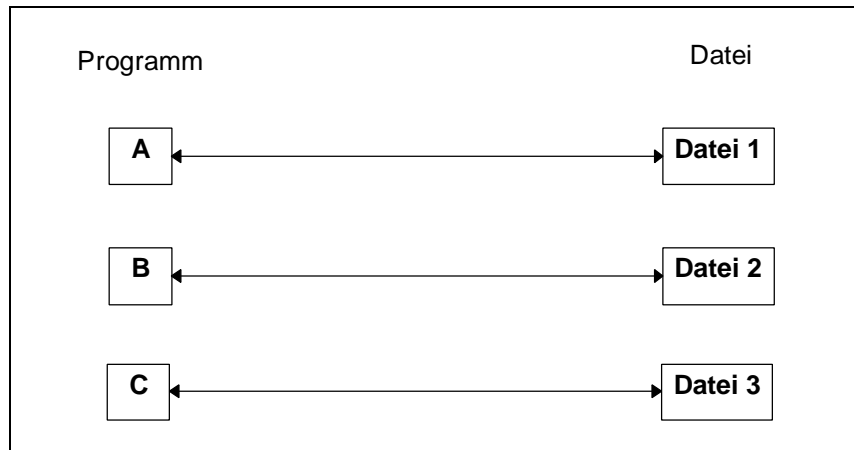
## 9) Redundanzfreiheit

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. Weniger Redundanz wird z. B. mit geringerer Flexibilität und Effizienz erkaufte.

# 1.2 Dateisysteme und Datenbanksysteme

Bei den weiteren Überlegungen sollen ausschließlich solche Datenbanksysteme betrachtet werden, mit denen die in Kapitel 1.1 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.

## 1.3 Das Konzept des Datenbanksystems

Der Versuch, den Begriff eines Datenbanksystems umfassend und exakt zu definieren, würde zu einer endlosen und fruchtlosen Diskussion führen - von zu vielen Seiten ist der Begriff mit Bedeutung belegt. Es gibt jedoch eine Reihe von grundlegenden Konzepten, die ein Datenbanksystem auszeichnen. Im folgenden sollen diese Konzepte herausgestellt werden.

Sehr allgemein gesprochen stellt man sich unter einem **Datenbanksystem** 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 20 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 (die 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 1.3-1:

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

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

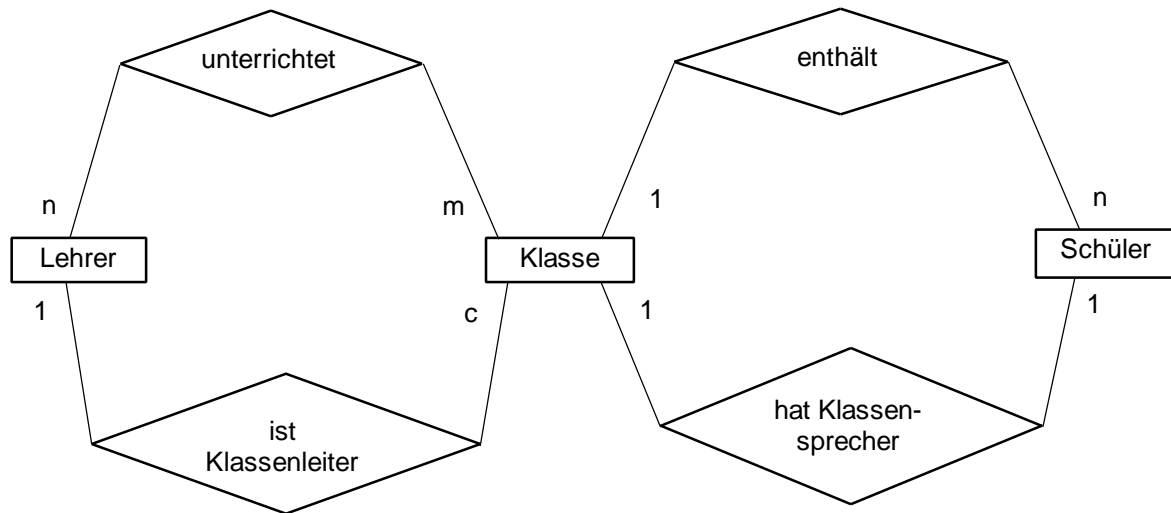
S-Nr	Schüler-Name	Vorname	Klasse	Fach	Fachlehrer	Note
427	Abel	Franz	12TA	E	Spieking	3
428	Bond	Regine	12TA	E	Spieking	1
429	...	...	...	...	...	...
430	...	...	...	...	...	...

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
...	...	...	...	...

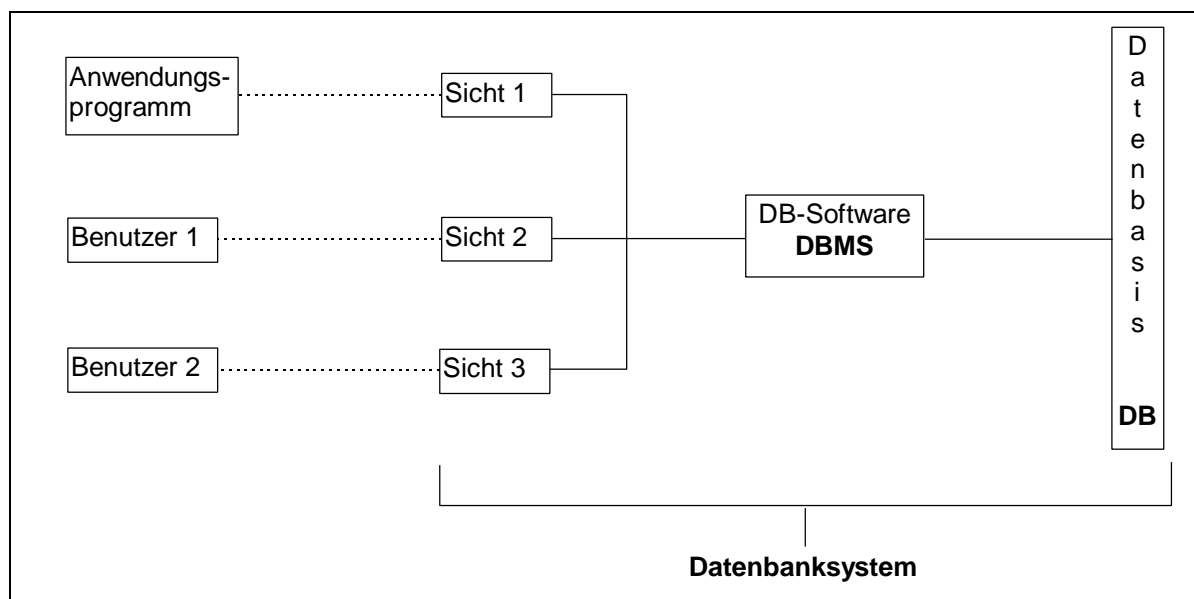


Ein Programmierer möchte die Daten vielleicht so sehen:



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

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



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

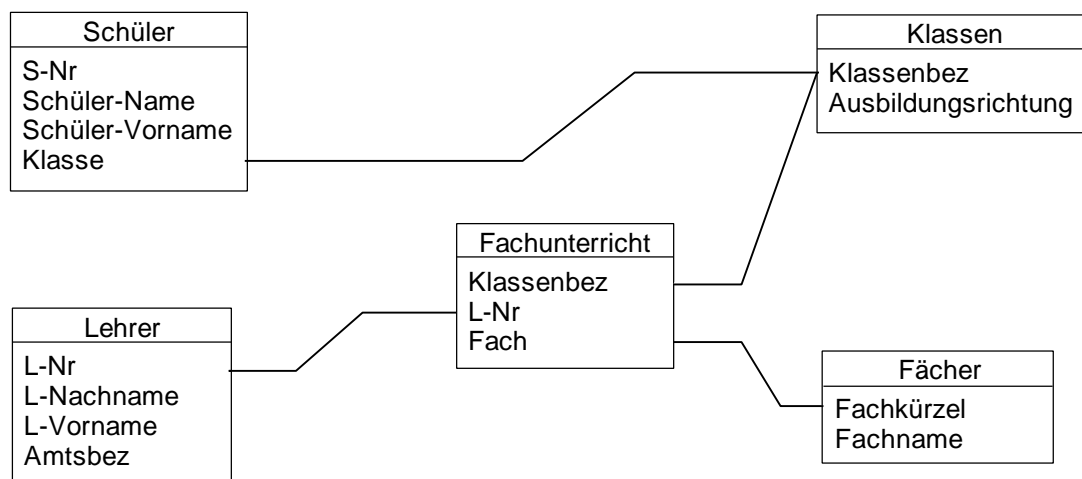
Der Begriff Datenbank wird als Bezeichnung für die Datenbasis eines Datenbanksystems verwendet. In der Literatur wird der Begriff auch als Kurzwort für das gesamte Datenbanksystem benutzt.

Die Komponenten eines Datenbanksystems sind die Datenbank (Datenbasis) und das Datenbankmanagementsystem. Eine Datenbank kann durch folgende Stichworte charakterisiert 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 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** (Vielfachzugriff), um **Daten aus den unterschiedlichen Dateien miteinander zu verknüpfen** (Flexibilität), und auf diese Weise zusätzliche Informationen aus den bereits bestehenden Dateien zu gewinnen (vgl. hierzu Bild 1-3). Hinzu kommt die Möglichkeit der komfortablen Beantwortung von Anfragen des Benutzers, mit Hilfe einer integrierten Abfragesprache.

„Benutzer“ ist hier sehr allgemein zu verstehen: Gemeint ist sowohl das von einem Anwendungsprogrammierer erstellte Anwendungsprogramm als auch der Benutzer, der - nicht durch ein Anwendungsprogramm geführt - im Dialog auf die Datenbank zugreift (vgl. hierzu Bild 1-2).



**Bild 1-3: Beziehungen zwischen verschiedenen Tabellen**

Die Sicht des Englischlehrers Spieking aus Beispiel 1.3-1 stellt eine Projektion der entsprechenden Datenfelder aus den verschiedenen Tabellen dar.

## 2 Aufbau und Arbeitsweise von Datenbanksystemen

In diesem Kapitel soll die Grundstruktur eines Datenbanksystems beschrieben werden.

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).

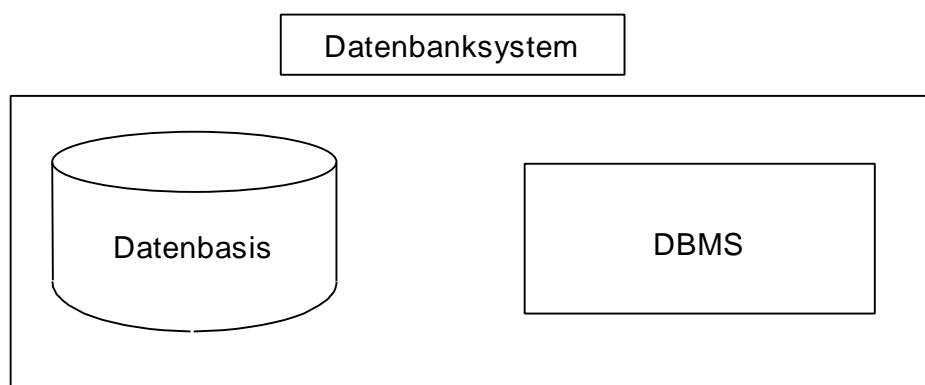


Bild 2-1: Komponenten eines Datenbanksystems

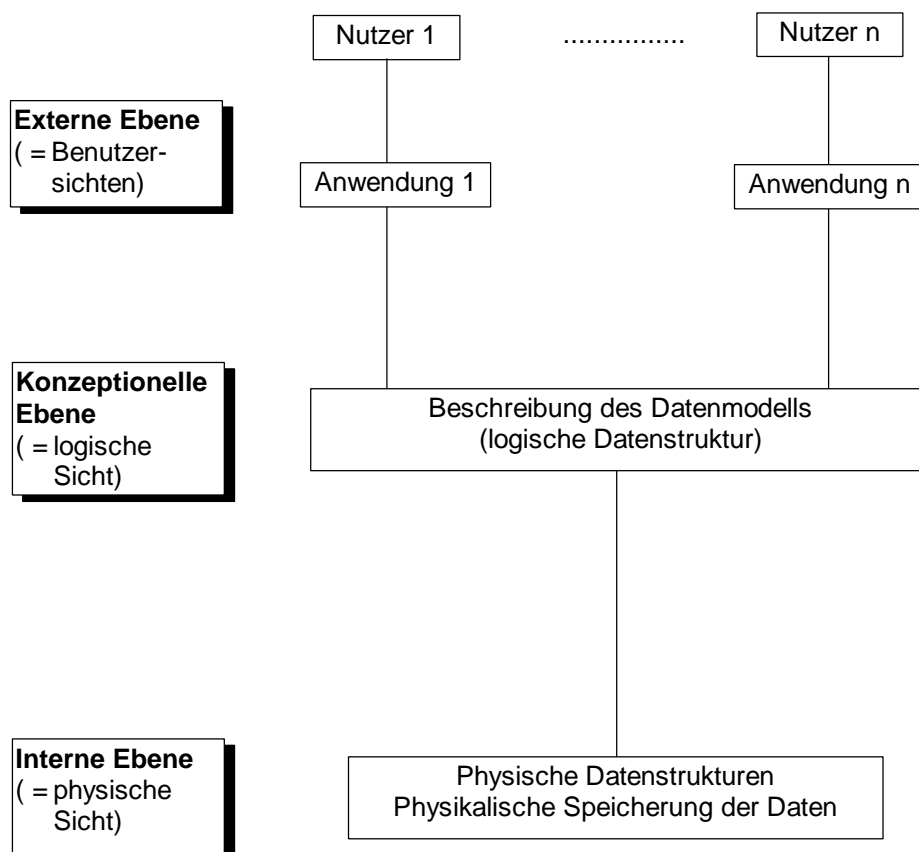
### 2.1 Das ANSI-Architekturmodell (3-Schichten-Architektur)

Urheber des ANSI-Architekturmodells ist das American National Standards Institute, 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 2-2: Datenbankebenen im ANSI-Architekturmodell**

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

### 2.1.1 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. Angaben dieser Art bleiben dem internen Schema vorbehalten.

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.

### 2.1.2 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. Da die Speicherungsform und die Zugriffspfade unmittelbar die Effizienz der Verarbeitung beeinflussen, sollte dem Entwurf eines internen Schemas eine sorgfältige Analyse der Benutzeraufgaben vorausgehen. So sollte vor allem die Zugriffshäufigkeit zu den Daten und das Zeitverhalten der Anwendungen ermittelt werden, da sie Rückschlüsse auf geeignete Speicherungsformen und Zugriffspfade zulassen.

### 2.1.3 Externes Schema

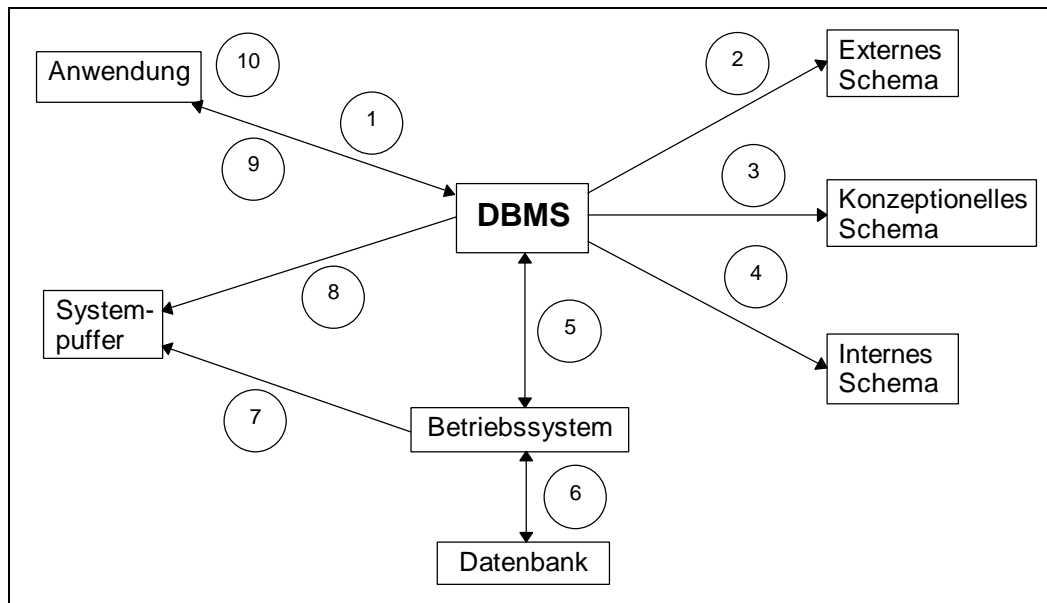
Ein externes Schema beschreibt einen Ausschnitt aus dem konzeptionellen Schema eines Unternehmens, der auf die spezielle Datensicht einer bestimmten Benutzergruppe zugeschnitten ist. Da ein externes Schema nur einen Teil der konzeptionellen Gesamtsicht wiedergibt, bezeichnet man es auch als Subschema. 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 1.3-1).

## 2.2 Arbeitsweise

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

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



**Bild 2-3: Abarbeitung einer Anfrage**

1. Das DBMS empfängt den Befehl des Anwendungsprogrammes, ein bestimmtes Objekt zu lesen.
2. Das DBMS holt sich die benötigten Definitionen des entsprechenden Objekttyps aus dem zugehörigen externen Schema.
3. Das DBMS stellt fest, auf welche konzeptionellen Objekte sich die Anfrage bezieht.
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.
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.

### 3 Datenmodellierung

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

<b>Externe Phase</b> – Ermittlung des Informationsbedarfs der Benutzer – Strukturierung dieser Informationen	<b>Informationsstruktur</b>	DBMS  unabhängig
<b>Konzeptionelle Phase</b> – formale und strukturierte Beschreibung aller relevanten Objekte und deren Beziehungen untereinander	<b>semantisches Modell</b>	
<b>Logische Phase</b> – Umsetzung des semantischen Datenmodells in ein relationales Datenbankmodell	<b>logisches / relationales Modell</b>	
<b>Physische Phase</b> – Modellierung der Datenbankstruktur mit einem relationalen Datenbankmanagementsystem (z. B. MS-Access)	<b>Implementierung mit Software</b>	DBMS abhängig

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 (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.

- **Bottom-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

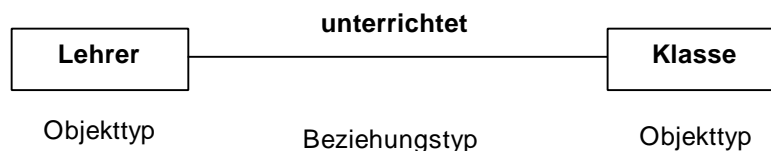
Dieses Verfahren eignet sich zur Konstruktion eines globalen wie auch eines anwendungsorientierten Grobmodells, kann aber auch bei der Verfeinerung eingesetzt werden.

Betrachtet man die Miniwelt „Schule“, erkennt man gewisse Objekte 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.

Um das Beispiel nicht zu komplex zu gestalten bzw. die Richtlinien des Datenschutzes einzuhalten (siehe hierzu auch 3.2.3, Festlegung des Datenrahmens), soll eine Beschränkung auf die Objekte **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<sup>1</sup>. 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*.

<sup>1</sup> Im objektorientierten Ansatz „besitzt“ ein Objekt neben Eigenschaften auch Methoden (Operationen). Eine Klasse definiert die Eigenschaften und Methoden gleichartiger Objekte.



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.

### 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 kann ein Grobmodell verfeinert und ein detailliertes, anwendungsorientiertes Datenmodell erstellt werden.

### 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.

### 3.2.1 Grundsätzliche Methoden

Es lassen sich grundsätzlich zwei Methoden unterscheiden:

#### 1. Top-down-Methode (vom Groben zum Detail)

- Entitätsklassen (Objekttypen) und deren Beziehungen festlegen (z. B. Entity-Relationship-Modell, Entität-Beziehungs-Modell).
- In der logischen Phase werden Attribute und Relationen (Tabellen) für die Entitäts- und Beziehungsmengen definiert, die den relationalen Regeln entsprechen (normalisierte Relationen).

#### 2. Bottom-up-Methode (vom Detail zum Ganzen)

- Die relevanten Attribute auswählen.
- In der logischen Phase müssen diese Attribute nach den relationalen Regeln kombiniert werden, so dass normalisierte Relationen entstehen.

Die Bottom-up-Methode wird aus folgenden Gründen nicht weiterverfolgt:

- Die im Rahmen der Informationsstruktur ermittelten Objekt- und Beziehungsklassen werden nicht genutzt (Bruch in der Systementwicklung).
- Bei der Transformation in ein relationales Modell müssen die fünf Normalformen beherrscht werden (hoher Lernaufwand).
- Die Methode ist sehr zeitaufwendig und unwirtschaftlich.

### 3.2.2 Elemente von Datenmodellen

Hier soll in der konzeptionellen Phase das Entity-Relationship-Modell (E-R-Modell) verwendet werden, weil es in der Praxis weit verbreitet ist und sich für die Entwicklung relationaler Datenbankmodelle bewährt hat. Die Elemente des E-R-Modells sind **Entitäten**, **Beziehungen zwischen Entitäten** und **Attribute** zur Charakterisierung von Entitäten. Im folgenden werden diese Elemente besprochen.

#### 3.2.2.1 Entitäten

Eine Entität kann sein:

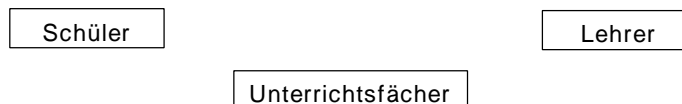
- ein Individuum (z. B. der Schüler *Meier*)
- ein reales Objekt (z. B. der Raum mit der Nr *112*)
- ein abstraktes Konzept (z. B. der Kurs *Informatik*)
- ein Ereignis (z. B. eine mündliche Prüfung)

Entitäten benötigt man für die datenmäßige Darstellung von Phänomenen eines Realitätsausschnitts. Sie sind gleichsam Abbilder dieser Phänomene in der Datenwelt. Die abgebildeten Phänomene können realer Natur, z. B. *Schüler*, *Lehrer*, *Klassen*, *Unterrichtsfächer* usw., oder nicht-gegenständlicher Natur, z. B. *Noten*, *Preise*, *Termine* usw. sein. Für den Entitätsbegriff folgt damit:

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

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 ihrem Wert durch verschiedene Attribute bzw. Attributkombinationen (vgl. Bild 3-7).

### 3.2.2.2 Beziehungen zwischen Entitäten

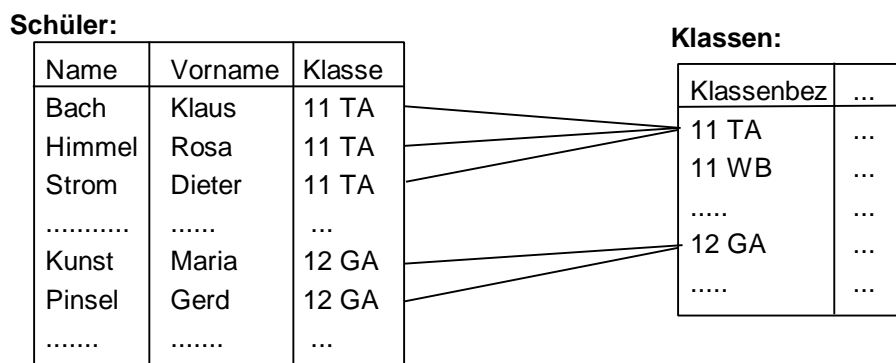
Zwischen Entitäten können **Beziehungen** bestehen.

**Beispiel:** *Lehrer unterrichtet Klasse*: *unterrichtet* ist eine Beziehung zwischen einem bestimmten Lehrer und einer bestimmten Klasse.

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, wieviel Entitäten der Entitätsmenge  $E2$  einer beliebigen Entität der Entitätsmenge  $E1$  zugeordnet sein können. Die Kardinalität spezifiziert also die Anzahl der an der Assoziation möglicherweise beteiligten Entitäten (Objekte) zu jedem beliebigen Zeitpunkt.

Eine **Beziehung** zwischen zwei Entitätsmengen  $E1$  und  $E2$  besteht aus der Assoziation  $a(E1, E2)$  und der dieser Assoziation entgegengerichteten Assoziation  $a^*(E2, E1)$ .

**Beispiel:** Eine Klasse kann von mehreren Schülern besucht werden ( $a(\text{Klasse}, \text{Schüler})$ ). Ein Schüler besucht genau eine Klasse ( $a^*(\text{Schüler}, \text{Klasse})$ ).



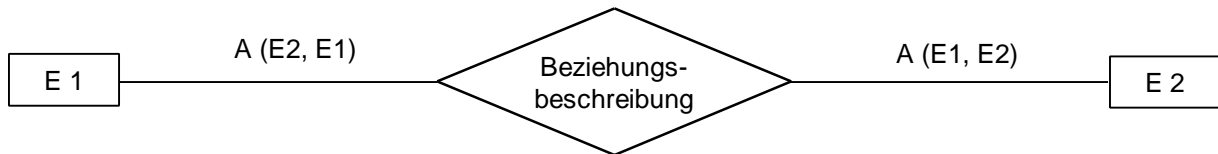
**Bild 3-3: Beziehung zwischen den Entitäten *Schüler* und *Klassen***

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

Anzahl der Entitäten $E2$ , die einer Entität $E1$ zugeordnet werden können ( $a(E1, E2)$ )	Symbol	Bezeichnung des Kardinalitätstyps der Assoziation
genau eine	1	<i>einfach</i>
keine oder eine	c oder 0, 1	<i>einfach-bedingt (konditionell)</i>
mindestens eine oder mehrere ( $m \geq 1$ )	m oder 1, m	<i>mehrfach (viele, komplex)</i>
keine, eine oder mehrere ( $m \geq 0$ )	mc oder 0, m	<i>mehrfach-bedingt</i>

**Tabelle 3-1: Kardinalitätstypen von Assoziationen**

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:

- einfach-einfache Beziehungen, d. h. die beiden Kardinalitätstypen sind einfach oder konditionell. Hierzu gehören: die Beziehungstypen (1, 1), (1, c) und (c, c).
- einfach-komplexe Beziehungen, d. h. einer der beiden Kardinalitätstypen ist einfach und der andere komplex. Hierzu gehören: die Kardinalitätstypen (1, m), (1, mc), (c, m) und (c, mc).
- komplex-komplexe Beziehungen, d. h. beide Kardinalitätstypen sind komplex. Hierzu gehören: die Kardinalitätstypen (m, m), (m, mc) und (mc, mc).

Im folgenden Bild 3-5 sind einige Beispiele für Beziehungen zusammengestellt:

Beziehungstyp	Beispiel
1-1	<pre> graph LR     K[Klasse] --- 1 --- H{hat}     H --- 1 --- KL[Klassenleiter]           </pre>
1-n	<pre> graph LR     K[Klasse] --- 1 --- E{enthält}     E --- n --- S[Schüler]           </pre>
n-m	<pre> graph LR     L[Lehrer] --- n --- U{unterrichtet}     U --- m --- K[Klasse]           </pre>

**Bild 3-5: Beispiele für Beziehungen**

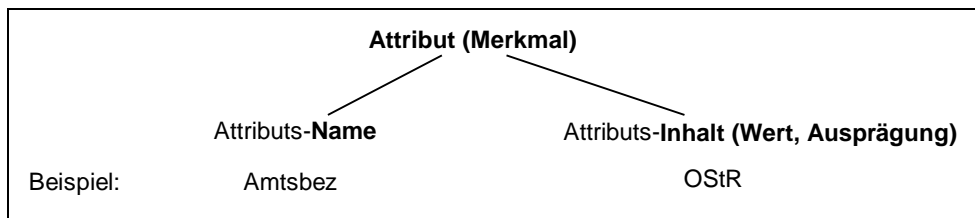
### 3.2.2.3 Attribute

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.

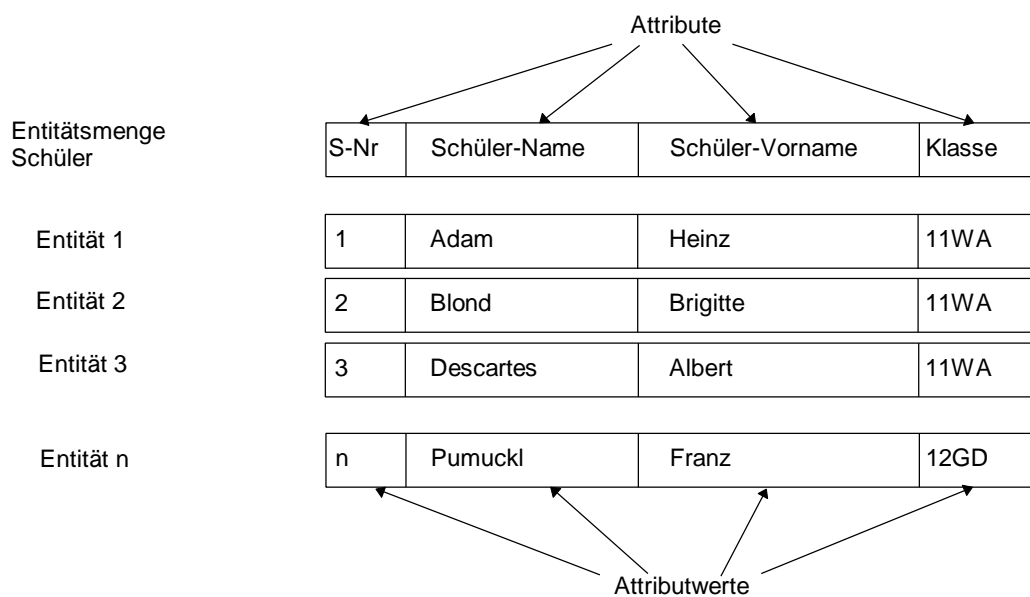
Ein **Attribut** (engl. attribute) beschreibt eine bestimmte Eigenschaft, die sämtliche Entitäten einer Entitätsmenge oder sämtliche Einzelbeziehungen einer Beziehung aufweisen.

Relevante Attribute der Entitätsmenge *Schüler* sind z. B. *Schülernummer*, *Name*, *Adresse*, *Klasse* usw. Die Beziehung *unterrichtet* wird inhaltlich durch die Attribute *L-Nr*, *Kl-bez*, *Fach* charakterisiert.

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**

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.

### 3.2.3 Umsetzung der Informationsstruktur in ein Entity-Relationship-Modell

Zur grafischen Darstellung eines E-R-Modells als sogenanntes Entity-Relationship-Diagramm (E-R-Diagramm) verwendet man die bereits bekannten Symbole:

- Rechtecke für Entitätsmengen und
- Rauten für Beziehungsmengen (Darstellung ohne Rauten ist auch üblich)

**Fallbeispiel:** Der folgende Ausschnitt aus einer Klassenverwaltung soll modelliert werden (die Beziehungen sollen in Kapitel 3.3.2 ermittelt werden):

**Bedingung 1:** Eine Klasse enthält Schüler.

**Bedingung 2:** Ein Lehrer ist in mehreren Klassen eingesetzt.

Bei der Festlegung des Datenrahmens für die Schüler und Lehrer müssen die Richtlinien der Datenschutzgesetze (siehe Vorwort bzw. Anhang) unbedingt beachtet werden. Darüber hinaus ist bei der Bearbeitung von Schüler- und Lehrerdaten die Bekanntmachung des bayerischen Staatsministeriums für Unterricht, Kultus, Wissenschaft und Kunst vom 19. März 1996 Nr. III/8-III/4-L0572-1/41785 (KWMBL. I Nr. 9/1996) zu berücksichtigen. Bei der Verarbeitung der Lehrerdaten ist ggf. zusätzlich eine Abstimmung mit der Personalvertretung erforderlich. Aus diesen Gründen wird der mögliche Datenrahmen auf den unten angegebenen eingeschränkt. Eine Erweiterung des in dem Beispiel verwendeten Datenrahmens ist nur im Hinblick auf die Einhaltung der oben genannten Vorschriften und Richtlinien zulässig. Deshalb empfiehlt sich generell bei der Verwendung personenbezogener Daten im Unterricht der Einsatz fiktiver Daten (wie in diesem Beispiel).

Folgende Daten werden von einer **Klasse** gespeichert:

- Klassenbezeichnung (z. B. *11 A*)
- Eine Jahrgangsstufe (z. B. *11*)
- einen Lehrer als Klassenleiter

Folgende Daten werden von einem

<b>Schüler</b>	bzw. von einem	<b>Lehrer</b> gespeichert:
- eine Schülernummer		- eine Lehrernummer
- der Familiennamen		- der Familiennamen
- der Vornamen		- der Vornamen
- das Geburtsdatum		- die Amtsbezeichnung
- die besuchte Klasse		- die Fächerkombination
- Eigenschaft Klassensprecher		

Die Unterrichtsfächer sollen in dieser ersten Version des Modells aus methodischen Gründen (liefert eine n-m-Beziehung zwischen den Lehrern und den Fächern und eine 1-n-Beziehung zwischen den Fächern und der Unterrichtsverteilung) nicht als Entität erfasst werden.

Bedingung 1 ergibt das folgende E-R-Diagramm (*Klasse* und *Schüler* sind Entitäten, *enthält* und *hat Klassensprecher* sind Beziehungen):

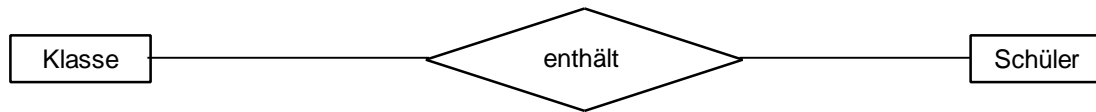


Bild 3-8: Das E-R-Diagramm der Bedingung 1

Die Erweiterung des Modells auf die Bedingung 2 ergibt nur eine neue Entität (*Lehrer*) und zwei neue Beziehungen (*ist in* und *ist Klassenleiter*).

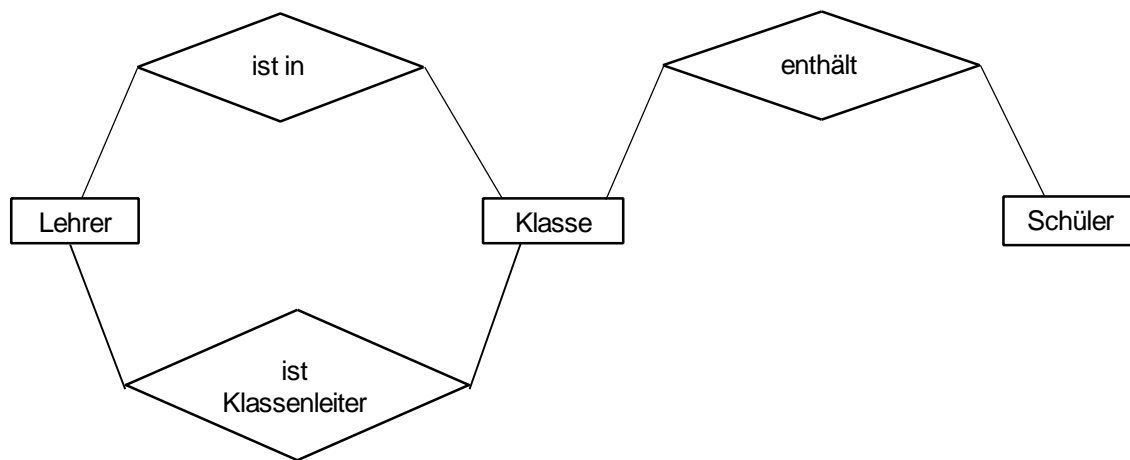


Bild 3-9: Das vollständige E-R-Diagramm des Modells *Klassenverwaltung* (Bedingung 2)

### 3.3 Umsetzung des semantischen Datenmodells in ein logisches Datenbankmodell

Es kommen folgende logische Datenbankmodelle in Betracht:

- das hierarchische Modell
- das Netzwerkmodell
- das Relationenmodell
- das objektorientierte Modell

Wir wollen hier nur das *Relationenmodell* behandeln, da die meisten PC-Datenbankmanagementsysteme darauf zugeschnitten sind, und jedes relationale Modell in ein hierarchisches Modell oder Netzwerkmodell umgesetzt werden kann. Das relationale Datenbankmodell wurde 1970 von dem Amerikaner Codd entwickelt.

### 3.3.1 Identifikationsschlüssel

Jede Entität einer Entitätsmenge ist ein individuelles Exemplar, das sich von den übrigen Entitäten der Menge unterscheidet. Der Unterschied zwischen zwei Entitäten einer Entitätsmenge drückt sich in unterschiedlichen Attributwerten aus. Zumindest für ein Attribut müssen zwei Entitäten unterschiedliche Werte annehmen, damit sie als individuelle Exemplare unterscheidbar sind. Jede Entität einer Entitätsmenge wird dann eindeutig durch die Werte sämtlicher Attribute identifiziert. Hier stellt sich die Frage, ob nicht wenige Attribute genügen oder sogar nur ein Attribut genügt, um die Entitäten einer Menge eindeutig zu identifizieren. Die Beantwortung dieser Frage führt zu dem Begriff des Identifikationsschlüssels.

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 3-9 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** 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. Beispielsweise könnte das Attribut *Klasse* einer Entitätsmenge *Schüler* als Sekundärschlüssel Verwendung finden.

### 3.3.2 Entwicklung des relationalen Datenbankschemas mit Hilfe von Abbildungsregeln

Das relationale Modell kann 1-1- und 1-n-Beziehungen unmittelbar abbilden, aber keine n-m-Beziehungen.

Der Übergang vom semantischen Datenmodell zum relationalen Datenmodell ist nicht ohne weiteres möglich, da das relationale Modell **keine komplex-komplexen Beziehungen** (n-m-Beziehungen) direkt abbilden kann. 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.



Im einzelnen gelten folgende **Abbildungsregeln**:

Entitätsmenge, Beziehung		Abbildungsregel für das relationale Modell
Entitätsmenge	<b>Typ 1</b>	Jede Entitätsmenge muss als eigenständige Relation definiert werden.
1:1-Beziehung	<b>Typ 2</b>	Es genügt eine Relation. Aus Datenschutzgründen oder für selten benötigte Informationen kann eine eigene Relation definiert werden.
c:1-Beziehung	<b>Typ 3</b>	Primärschlüssel der 1-Relation muss als Fremdschlüssel in der c-Relation auftreten. Eine Beziehungsmengen-Relation ist nicht notwendig.
1:m-Beziehung	<b>Typ 4</b>	Primärschlüssel der 1-Relation ist Fremdschlüssel in der m-Relation. Keine eigene Beziehungsmengen-Relation notwendig.
Alle komplex-komplexen Beziehungen (n:m-, n:mc-, mc:n-, nc:mc-Beziehungen)	<b>Typ 5</b>	Jede komplex-komplexe Beziehungsmenge muss als eigenständige Relation definiert werden. Die Primärschlüssel der zugehörigen Entitätsmengen treten als Fremdschlüssel in der Beziehungsmengen-Relation auf.

**Tabelle 3-2: Abbildungsregeln**

Für die Klassenverwaltung gilt:

Die Entitätsmengen *Schüler*, *Klassen* und *Lehrer* müssen als eigenständige Relationen (Tabellen) definiert werden (Typ 1).

Zwischen den Entitätsmengen *Klassen* und *Schüler* liegt eine Beziehung vor:

– *Klasse enthält Schüler*

Im nächsten Schritt sind die sich aus der Informationsstruktur (siehe Fallbeispiel 3.2.3) ergebenden Kardinalitätstypen der Assoziationen zu ermitteln (vgl. hierzu Bild 3-8 und Bild 3-9):

**Beziehung *Klasse enthält Schüler*:**

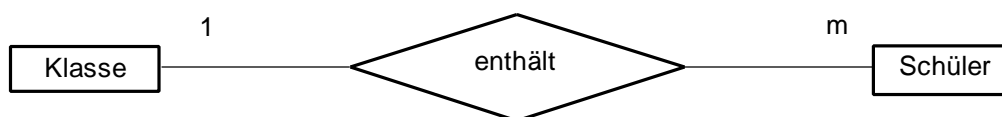
$a(Klasse, Schüler)$

1	?	Wieviele Schüler kann eine Klasse enthalten?
	m	Eine Klasse kann mehrere Schüler enthalten.
		Kardinalität der Assoziation: m

$a^*(Schüler, Klasse)$

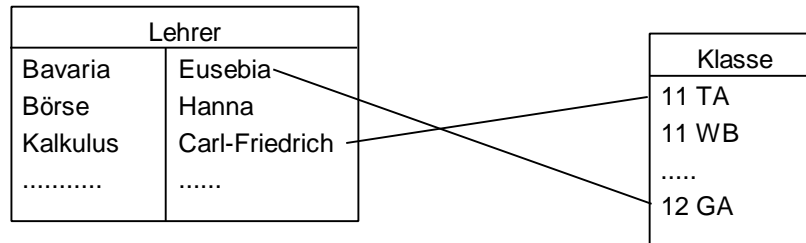
1	?	Wieviele Klassen kann ein Schüler besuchen?
	1	Ein Schüler kann in einem Halbjahr nur eine Klasse besuchen.
		Kardinalität der Assoziation: 1

**Ergebnis 1:** Die Beziehung *Klasse enthält Schüler* ist eine 1-m-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Typ 4).



**Bild 3-10: Beziehung *Klasse enthält Schüler***



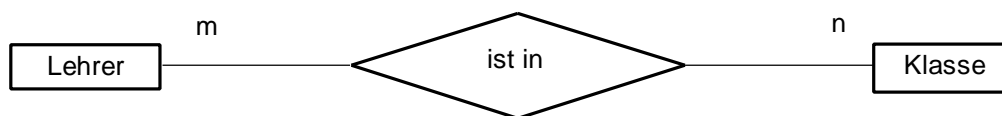
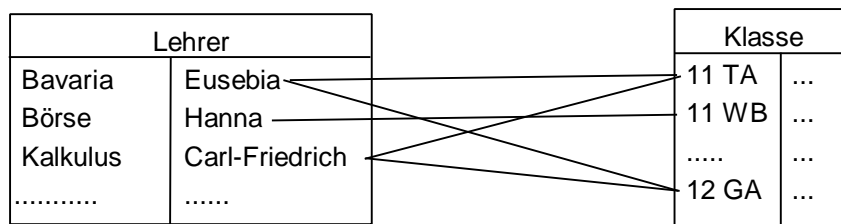
Bild 3-13: Beispieldaten für die Beziehung *Lehrer ist Klassenleiter***Beziehung *Lehrer ist in Klasse*:** $a(\text{Lehrer}, \text{Klasse})$ 

- 1      ?      In wie vielen Klassen kann ein Lehrer eingesetzt sein?  
 n      Ein Lehrer kann in einer oder mehreren Klassen eingesetzt sein.  
 Kardinalität der Assoziation: n

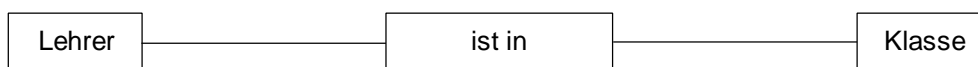
 $a^*(\text{Klasse}, \text{Lehrer})$ 

- 1      ?      Wie viele Lehrer können in einer Klasse eingesetzt sein?  
 m      In einer Klasse können mehrere Lehrer eingesetzt sein.  
 Kardinalität der Assoziation: m

**Ergebnis 3:** Die Beziehung *Lehrer ist in Klasse* ist eine n-m-Beziehung; es ist eine eigene Beziehungsmengen-Relation notwendig (Typ 5).

Bild 3-14: Beziehung *Lehrer ist in Klasse*Bild 3-15: Beispieldaten für die Beziehung *Lehrer ist in Klasse*

Es liegt also eine komplex-komplexe Beziehung (Typ 5, Tabelle 3-2) vor. Nach den Abbildungsregeln muss folglich für die Beziehungsmenge *ist in* eine eigene Relation *ist in* konstruiert werden.

Bild 3-16: Relationen *Lehrer, ist in, Klasse*

Im nächsten Schritt sind die sich aus der Informationsstruktur (siehe Fallbeispiel 3.2.3) ergebenden Kardinalitätstypen der Assoziationen zu ermitteln:

**Beziehung *Lehrer - ist in*:** Wie in 3.2.3 erwähnt, sollen die Unterrichtsfächer in diesem Modell aus Komplexitätsgründen nicht als Entität erfasst werden. Deshalb wird zwischen dem Lehrer, der nur in einem Fach in einer Klasse eingesetzt ist, und dem Lehrer, der in mehreren Fächern in einer Klasse eingesetzt ist, nicht unterschieden.

$a(\text{Lehrer}, \text{ist in})$

- 1     ?    In wie vielen Klassen kann ein bestimmter Lehrer eingesetzt sein?
- n    In einer oder mehreren Klassen
- Kardinalität der Assoziation: n

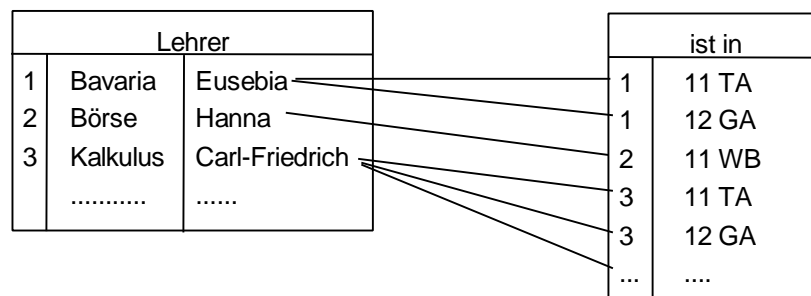
$a^*(\text{ist in}, \text{Lehrer})$

- 1     ?    Wie oft kann ein bestimmter Lehrer in einer bestimmten Klasse eingesetzt sein?
- 1    Einmal oder keinmal.
- Kardinalität der Assoziation: 1

**Ergebnis 4:** Die Beziehung *Lehrer - ist in* ist eine 1-n-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Typ 4).



**Bild 3-17: Beziehung *Lehrer - ist in***



**Bild 3-18: Beispieldaten für die Beziehung *Lehrer - ist in***

**Beziehung *ist in - Klasse*:**

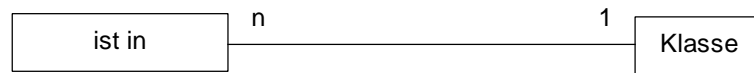
$a(\text{ist in}, \text{Klasse})$

- 1     ?    Wie oft kann ein Lehrer einer bestimmten Klasse in dieser Klasse eingesetzt sein?
- 1    Ein Lehrer kann in einer bestimmten Klasse einmal eingesetzt sein.
- Kardinalität der Assoziation: 1

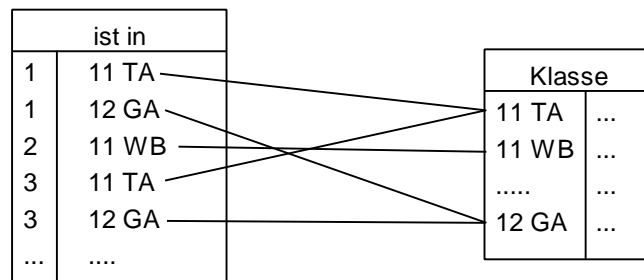
$a^*(\text{Klasse}, \text{ist in})$

- 1     ?    In wie vielen Klassen kann ein bestimmter Lehrer eingesetzt sein?
- n    Ein Lehrer kann in einer oder mehreren Klasse eingesetzt sein.
- Kardinalität der Assoziation: n

**Ergebnis 5:** Die Beziehung *ist in*- *Klasse* ist eine n-1-Beziehung; es ist keine eigene Beziehungsmengen-Relation notwendig (Typ 4).

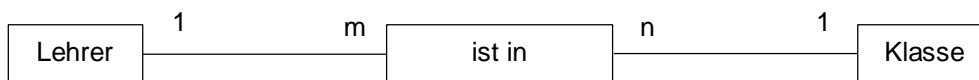


**Bild 3-19:** Beziehung *ist in* - *Klasse*

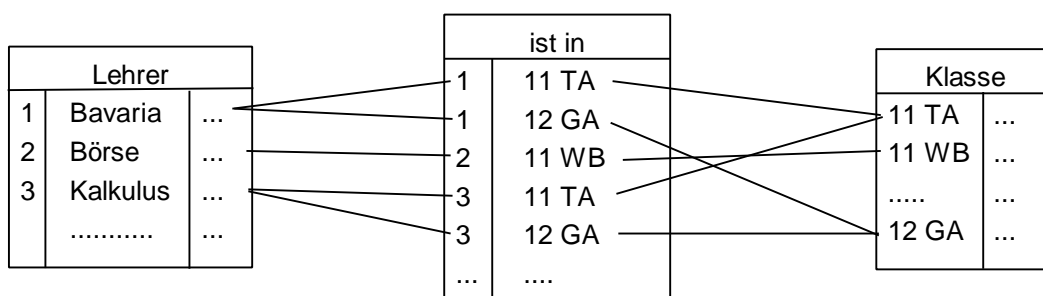


**Bild 3-20:** Beispieldaten für die Beziehung *ist in* - *Klasse*

**Zusammenfassung:** Insgesamt wurde die n-m-Beziehung *Lehrer ist in Klasse* durch die Konstruktion der Beziehungsmengen-Relation *ist in* in die zwei Beziehungen *Lehrer - ist in* (1-m-Beziehung) und *ist in - Klasse* (n-1-Beziehung) aufgelöst. Die Beziehungsmengen-Relation *ist in* muss auf der logischen Ebene als eigenständige Tabelle dargestellt werden.

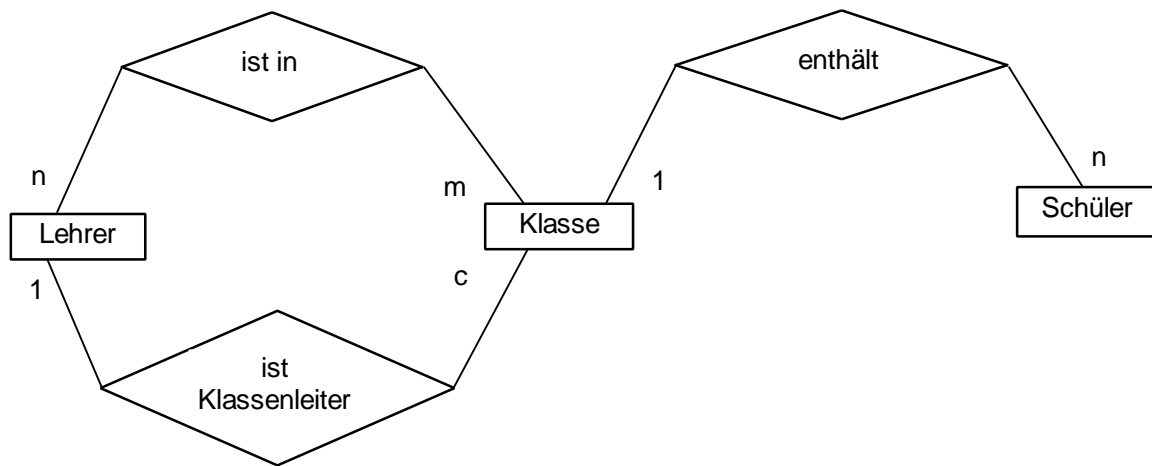


**Bild 3-21:** Beziehungen mit der Beziehungsmengen-Relation *ist in*



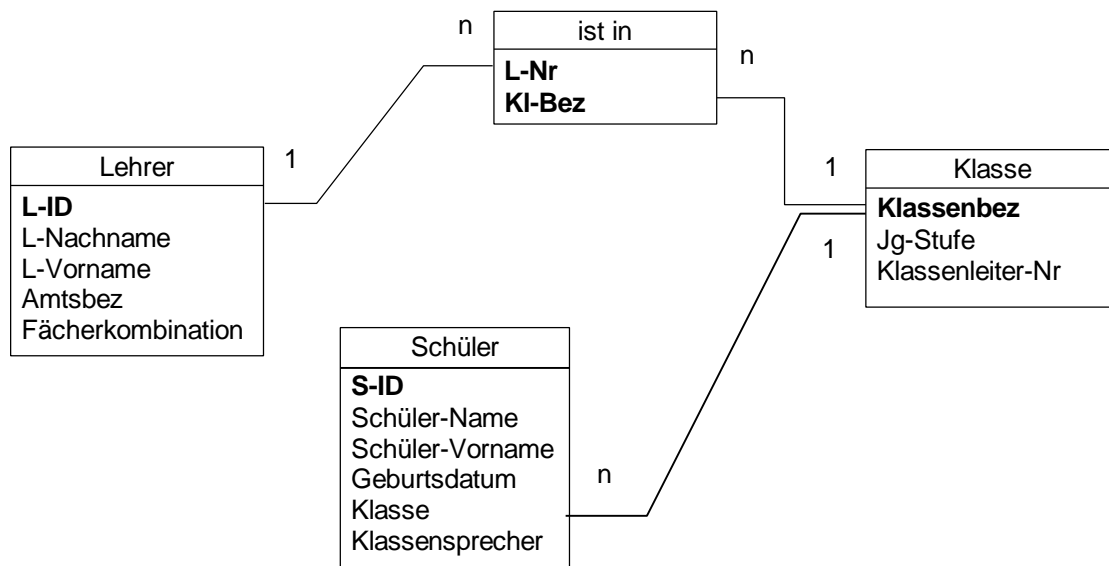
**Bild 3-22:** Beispieldaten für die Beziehungen *Lehrer - ist in - Klasse*

Das in Bild 3-23 dargestellte semantische Modell wurde somit in das in Bild 3-24 gezeigte logische Modell umgesetzt.



**Bild 3-23: E-R-Diagramm mit Beziehungen des Modells Klassenverwaltung**

Daraus ergibt sich die folgende Struktur des relationalen Modells der Klassenverwaltung:



**Bild 3-24: Struktur des relationalen Modells der Klassenverwaltung**

**Hinweise:** Primärschlüsselfelder sind fett gedruckt. Bei 1-n-Beziehungen enthält die 1-Seite in der Regel die Endung „-ID“ und die n-Seite die Endung „-Nr“; dies ist nicht zwingend erforderlich, hat sich aber aus methodischen Gründen im Unterricht bewährt. Außerdem wird die Klassenleiter-Beziehung zwischen den Tabellen *Lehrer* und *Klassen* nicht im globalen logischen Modell realisiert, sondern nur lokal in denjenigen Abfragen, in denen die „Klassenleiter-Beziehung“ benötigt wird. Andernfalls müsste für diejenigen Abfragen, in denen diese Beziehung nicht benötigt wird, die entsprechende Verknüpfung gelöscht werden, da sonst nur diejenigen Lehrer einer Klasse angezeigt würden, die sowohl unterrichten als auch Klassenleiter sind.

### 3.3.3 Entwicklung des relationalen Datenbankschemas mit Hilfe der 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 den Normalformen. 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 folgenden Regeln beachtet werden:

#### 1. Ist jeder Datensatz verschieden von den anderen Datensätzen?

Tabelle: Lehrer

<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>
Bavaria	Eusebia	StRin
Börse	Hanna	StRin
Bavaria	Eusebia	StRin

Tabelle 3-3: Unterscheidung von Datensätzen

#### Fehler in der Tabelle *Lehrer*:

Die erste und die dritte Zeile der Tabelle *Lehrer* sind identisch. Die Datensätze sind nicht eindeutig identifizierbar. Dadurch ist es möglich, dass Datensätze logisch widersprüchlich (inkonsistent) sind, d. h., die **Integrität der Entität** verletzt ist.

#### Lösung:

Es muss ein Attribut (Feld) oder eine Kombination von Attributen gefunden werden, die jeden Datensatz eindeutig identifiziert. Dieses Attribut bzw. diese Attributkombination wird **Primärschlüssel** genannt. Die Integrität der Entität ist gewahrt, wenn der Primärschlüssel **eindeutig und nicht leer** ist. Grundsätzlich gibt es zwei Möglichkeiten:

- Eine Kombination aus mehreren Feldern kann zum natürlichen Primärschlüssel erklärt werden (z. B. Name, Vorname, Amtsbez, Fächerkombination).
- Es kann ein künstlicher Primärschlüssel erzeugt werden, in dem man ein Feld einfügt, das die Datensätze der Reihe nach durchnummeriert (hier: *Lehrernummer*).

Ein Mehrfelder-Primärschlüssel muss eine **minimale** Kombination von Merkmalen sein, d. h. kein Merkmal der Kombination kann gestrichen werden, ohne dass die Eindeutigkeit der Identifikation verloren geht.

Mit einem Mehrfelder-Primärschlüssel sind meist folgende Nachteile verbunden:

- Man muss genau überlegen, ob die gleiche Attributwert-Kombination (Mehrfelder-Primärschlüssel) in der Tabelle nicht doppelt vorkommen kann.
- Je länger der Primärschlüssel, desto länger die Zugriffszeit. Da der Primärschlüssel für Verknüpfungen benötigt wird, sollte er nicht zu lang sein.
- Bei der Planung der Tabellen ist besonders darauf zu achten, dass jedes Attribut, das nicht zum Primärschlüssel gehört, **alle** Attribute des Primärschlüssels zur eindeutigen Identifikation benötigt (*volle funktionale Abhängigkeit, 2. Normalform*).
- Personenbezogene Bestandteile des Schlüssels (z. B. Geburtsdatum) können zu Datenschutzproblemen führen.

Außerdem lassen nicht alle Datenbankmanagementsysteme die Definition von Mehrfelder-Primärschlüsseln zu.

Tabelle: Lehrer

<b><i>L-ID</i></b>	<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>
<b>1</b>	Bavaria	Eusebia	StRin
<b>2</b>	Börse	Hanna	StRin
<b>3</b>	Kalkulus	Carl-Friedrich	StD

Tabelle 3-4: Einführung eines Primärschlüsselfeldes (fett gedruckt)

In der Regel ist also ein Einfeld-Primärschlüssel einem Mehrfelder-Primärschlüssel vorzuziehen. Bei der Realisierung von n-m-Beziehungen sind jedoch Mehrfelder-Primärschlüssel angebracht.

**Regel Nr. 1:** Jeder Datensatz (Tupel, Entität) einer Tabelle muss durch einen Primärschlüssel eindeutig identifizierbar sein.

## 2. Sind alle Einzelinformationen in eigenen Feldern gespeichert?

Tabelle: Lehrer und Klassen

<i>L-Name</i>	<i>Amtsbez</i>	<i>Klassen</i>	<i>Fächerkombination</i>
Bavaria Eusebia	StRin	11 TA, 11 TB, 12 WA, 12 GB, 12 SC	D/G/Sk
Börse Hanna	StRin	11 WA, 11 WB, 12 WA, 12 WD, 12 WF	W/Sw

Tabelle 3-5: Zusammengesetzte Attributinhalt

### Fehler in der Tabelle *Lehrer und Klassen*:

1. Das Feld *Name* enthält sowohl den Nachnamen als auch den Vornamen des Lehrers. So ist es z. B. nicht möglich, ein persönliches Anschreiben an den Lehrer richtig zu verfassen ("Sehr geehrte Frau Börse Hanna ..." ist nicht üblich.).
2. Der Attributwert des Attributs *Klassen* setzt sich aus allen Klassen zusammen, in denen der betreffende Lehrer eingesetzt ist. Die einzelnen Klassen sollten besser in getrennten Feldern stehen, um maximale Flexibilität beim Umgang mit den Daten zu gewährleisten. Ein Lehrer



kann in verschiedenen Klassen eingesetzt sein, in einer Klasse allerdings höchstens einmal; es sind aber andere Lehrer in derselben Klasse eingesetzt. Zwischen den Relationen *Klassen* und *Lehrer* besteht also eine n-m-Beziehung.

Auf den ersten Blick sieht es so aus, als ob das Feld *Fächerkombination* ebenfalls mehrere Werte enthält (nämlich die einzelnen Unterrichtsfächer des betreffenden Lehrers). Dies ist aber nicht der Fall. Bei der Angabe der Fächerkombination in der obigen Form handelt es sich um eine Information, die genau in dieser Form (ein einziger String) für die weitere Bearbeitung benötigt wird. Bei der Umsetzung in ein logisches Datenbankmodell müssen dann die einzelnen Unterrichtsfächer aus verschiedenen Gründen in einer eigenen Entität erfasst werden:

- Einige Unterrichtsfächer haben andere Bezeichnungen wie die Fächer aus der Fächerkombination der betreffenden Fachlehrer (z. B.: Fachlehrer mit Fächerkombination W kann die Fächer ODV, W und RW unterrichten).
- Nicht jeder Lehrer unterrichtet in jedem Schuljahr alle Fächer seiner Fakultas (aus seiner Fächerkombination). In diesem Fall ginge Information verloren (in welchen Fächern kann der betreffende Kollege überhaupt Unterricht erteilen). Bei der Verwendung dieser Information mit realen Daten sind aber unbedingt die Vorschriften des Datenschutzes zu beachten.

Das Feld *Fächerkombination* wird in diesem Fall als ein String aufgefasst, der die Information über die gesamte Fakultas des betreffenden Kollegen beinhaltet. Diese Information wird nicht beim Unterrichtseinsatz (Fachunterricht in den jeweiligen Klassen) verwendet, sondern dient nur als zusätzliche Informationsangabe über den einzelnen Kollegen.

Eine Tabelle befindet sich in der **1. Normalform (1NF)**, wenn sie nur **elementare Attribute** enthält.

**Regel Nr. 2:** Jede Teilinformation sollte in einem eigenen Feld gespeichert werden.

### 3. Kommen *Aufzählfelder* vor?

Tabelle: Lehrer unterrichten Klassen

<i>L-Nr</i>	<i>Klasse 1</i>	<i>Klasse 2</i>	<i>Klasse 3</i>
1	12TA	12GB	12GB
2	11WA	11WB	12WC
3	12TA	12SA	

**Tabelle 3-6: Aufzählfelder**

#### **Fehler in der Tabelle *Lehrer unterrichten Klassen*:**

Die Felder *Klasse 1*, *Klasse 2* und *Klasse 3* enthalten jeweils gleichartige Informationen, nämlich die Klassenbezeichnungen der von einem Lehrer unterrichteten Klassen. Probleme ergeben sich, wenn z. B. ein Lehrer in vier oder mehr Klassen Unterricht erteilt. Sieht man aber für jeden Lehrer die maximale Anzahl von möglichen Klasseneinsätzen vor, besitzen viele Datensätze keine Attributwerte. Dies ist ein Schönheitsfehler und muss bei Abfragen evtl. berücksichtigt werden; außerdem wirkt sich dies auf den Speicherbedarf aus, da auch ein leeres Feld Speicherplatz benötigt.

**Lösung:** In diesem Fall sollten die Informationen über die Klasse, in der ein Lehrer Unterricht erteilt, als eigener Datensatz gespeichert werden.

**Regel Nr. 3:** Für aufzählende Felder sollte eine eigene Tabelle angelegt werden.

Tabelle: Lehrer in Klasse

<i>L-Nr</i>	<i>Kl-Bez</i>	<i>Fach</i>
1	12TA	D
1	12GB	Ges
1	12GB	Soz
2	11WA	Wi
2	11WB	Wi
...	...	...

Tabelle 3-7: Korrekte Tabelle der Lehrer-Klassen-Zuordnung

#### 4. Enthält jedes Feld in jedem Datensatz Attributwerte?

##### Fehler in der Beispieldabelle *Lehrer unterrichten Klassen* (Tabelle 3-6):

Der Datensatz Nr. 3 enthält in den Feldern *Klasse 3* keine Attributwerte. Dies ist ein Schönheitsfehler, der sich auch auf den Speicherbedarf auswirken kann, da auch ein leeres Feld Speicherplatz benötigt (siehe unter 3.).

##### Lösung:

Auch in diesem Fall sollte die Teilinformation in einer eigenen Tabelle (vgl. Tabelle 3-7) gespeichert werden. Beispiel: Siehe unter 3.!

**Regel Nr. 4:** Felder, die nicht für jeden Datensatz ausgefüllt werden können, sollten in einer eigenen Tabelle gespeichert werden.

#### 5. Werden für jedes Attribut, das nicht zum Primärschlüssel gehört, alle Attribute des Primärschlüssels zur eindeutigen Identifikation benötigt?

Tabelle: Lehrer in Klasse

<i>L-Nr</i>	<i>Kl-Bez</i>	<i>Jahrgangsstufe</i>
1	12TA	12

Tabelle 3-8: Nicht volle funktionale Abhängigkeit vom Primärschlüssel

##### Fehler in der Tabelle *Lehrer in Klasse*:

Die Jahrgangsstufe ist allein durch das Attribut *Kl-Bez* eindeutig identifizierbar, das Attribut *L-Nr* des Primärschlüssels wird dafür nicht benötigt. Das Attribut *Jahrgangsstufe* gehört deshalb nicht zur Relation *Lehrer in Klasse*, sondern zur Tabelle *Klassen*.

**Bemerkung:** In Schulen, die aus verschiedenen Ausbildungsrichtungen zusammengesetzte Klassen enthalten, gehört die Ausbildungsrichtung eigentlich zur Relation *Schüler*. Denn die Ausbildungsrichtung ist in Mischklassen nicht von der Klassenbezeichnung, sondern vom Schüler abhängig.

Tabelle: Klassen

<i>Klassenbez</i>	<i>Jahrgangsstufe</i>	<i>Klassenleiter L-Nr</i>
12SC	12	12

Tabelle 3-9: Nicht erfüllte Minimalitätsbedingung

**Fehler in der Tabelle *Klassen*:**

Die *Klassenleiter L-Nr* ist schon allein durch das Attribut *Klassenbez* des Primärschlüssels eindeutig identifizierbar, das Attribut *Jahrgangsstufe* des Primärschlüssels wird dafür nicht benötigt. In diesem Fall ist es falsch, das Attribut *Jahrgangsstufe* als Teil des Primärschlüssels zu deklarieren, da das Attribut *Klassenbez* genügt, jeden Datensatz der Tabelle eindeutig zu identifizieren (**Minimalitätsbedingung**).

Eine Relation der 1. Normalform befindet sich in der **2. Normalform (2NF)**, wenn zur Beschreibung der Abhängigkeit vom Primärschlüssel für jedes Attribut, das nicht zum Primärschlüssel gehört (mehrere Attribute zusammen können einen Primärschlüssel bilden), **alle** Attribute des Primärschlüssels benötigt werden (volle funktionale Abhängigkeit). Wenn der Primärschlüssel nur aus **einem** Attribut besteht, ist jede Relation (Tabelle), die sich in der 1. Normalform befindet, zwangsläufig auch in der 2. Normalform.

**Regel Nr. 5:** Jedes Attribut, das nicht zum Primärschlüssel gehört, muss alle Attribute des Primärschlüssels zur eindeutigen Identifikation benötigen.

**6. Stehen alle Felder in direktem Bezug zum Primärschlüssel?**

Tabelle: Lehrerdaten

<i>L-Nr</i>	<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>	<i>Besoldungsgruppe</i>
1	Bavaria	Eusebia	StRin	A13
2	Börse	Hanna	StRin	A13
3	Kalkulus	Carl-Friedrich	StD	A15

Tabelle 3-10: Indirekte Abhängigkeit vom Primärschlüssel

**Fehler in der Tabelle *Lehrerdaten*:**

Die *Besoldungsgruppe* ist eigentlich von der *Amtsbez* abhängig. Sie steht also zur *L-Nr* in keinem direkten Bezug. Wenn sich die Besoldungsgruppe einer Amtsbezeichnung ändert, muss die Änderung in jedem Datensatz der Tabelle *Lehrerdaten* vorgenommen werden, der diese Besoldungsgruppe ausweist.

**Lösung:** Eine eigene Tabelle für die Besoldungsgruppen anlegen.

Tabelle: Besoldungsgruppen

<i>Amtsbezeichnung</i>	<i>Amtsbez Kürzel</i>	<i>Besoldungsgruppe</i>
...	...	...
Studienrat	StR	A13
Studienrätin	StRin	A13
Oberstudienrat	OStR	A14
...	...	...

**Tabelle 3-11: Tabelle zur Auflösung indirekter Abhängigkeit**

Eine Relation befindet sich in der **3. Normalform (3NF)**, wenn sie die zweite Normalform erfüllt und alle Attribute, die nicht zum Primärschlüssel gehören, direkt von diesem abhängen. Alle Nicht-Primärschlüssel-Attribute müssen also voneinander unabhängig sein. Es ist nicht erlaubt, dass ein Attribut, das nicht zum Primärschlüssel gehört, nur indirekt (transitiv) von diesem abhängt.

**Regel Nr. 6:** Felder, die in keinem direkten Bezug zum Primärschlüssel der Tabelle stehen, sollten in einer eigenen Tabelle gespeichert werden.

## 7. Enthalten Felder berechenbare Daten (Prozessdaten)?

Tabelle: Schüler

<i>S-Nr</i>	<i>Schüler-Name</i>	<i>Schüler-Vorname</i>	<i>Klasse</i>	<i>Geb-Datum</i>	<i>Alter</i>
345	Mayr	Hans-Josef	12TP	15.09.81	17

**Tabelle 3-12: Prozessdaten**

### Fehler in der Tabelle *Schüler*:

Das Alter kann aus dem Geburtsdatum (*Geb-Datum*) berechnet werden. Es ist deshalb überflüssig das Alter zu speichern. Dies kann sogar zu Fehlern (Inkonsistenzen) führen (spätestens nach einem Jahr).

**Regel Nr. 7:** Felder sollten keine berechenbaren Daten enthalten.

Bei Feldern mit berechenbaren Daten wird in der Regel gegen die dritte Normalform verstoßen, die verlangt, dass alle Attribute, die nicht zum Primärschlüssel gehören, voneinander unabhängig sein müssen. Die Dauer in Tagen hängt nämlich von den Attributen *Eintritt* und *Austritt* ab und nicht direkt vom Primärschlüssel. Die Regel 7 ist also in der Regel 6 enthalten.

Komplexe Berechnungen sind zeitaufwendig. Wegen der Performance (schnellen Verfügbarkeit der Informationen) wird deshalb oft die Regel Nr. 7 nicht eingehalten. Grundsätzlich sollte jedoch erst einmal vollständig normalisiert werden. Anschließend kann zur Verbesserung des Laufzeitverhaltens wieder teilweise denormalisiert werden.

Bei Anwendung der Regeln ergibt sich ebenfalls die in Bild 3-24 gezeigte Struktur des relationalen Datenmodells.

## 3.4 Vertiefung des relationalen Datenbankmodells

### 3.4.1 Grundlegende Begriffe

Gegenüberstellung der Grundbegriffe aus den verschiedenen Entwurfsebenen

Externe Phase	Konzeptionelle Phase	Logische Phase	Physikalische Phase
Informationsstruktur	E-R-Modell	Relationenmodell	RDBS
Objektstruktur	E-R-Diagramm	Relationenstruktur	Datenbank
Objekt- /Beziehungsmenge	Entitäts- / Bezie- hungsmenge	Relation	Tabelle / Datendatei
Objekt / Beziehung	Entität / Beziehung	Tupel	Datensatz
Attribut / Eigenschaft	Attribut / Merkmal	Attribut	Feld

Tabelle 3-13: Grundbegriffe der Entwurfsebenen

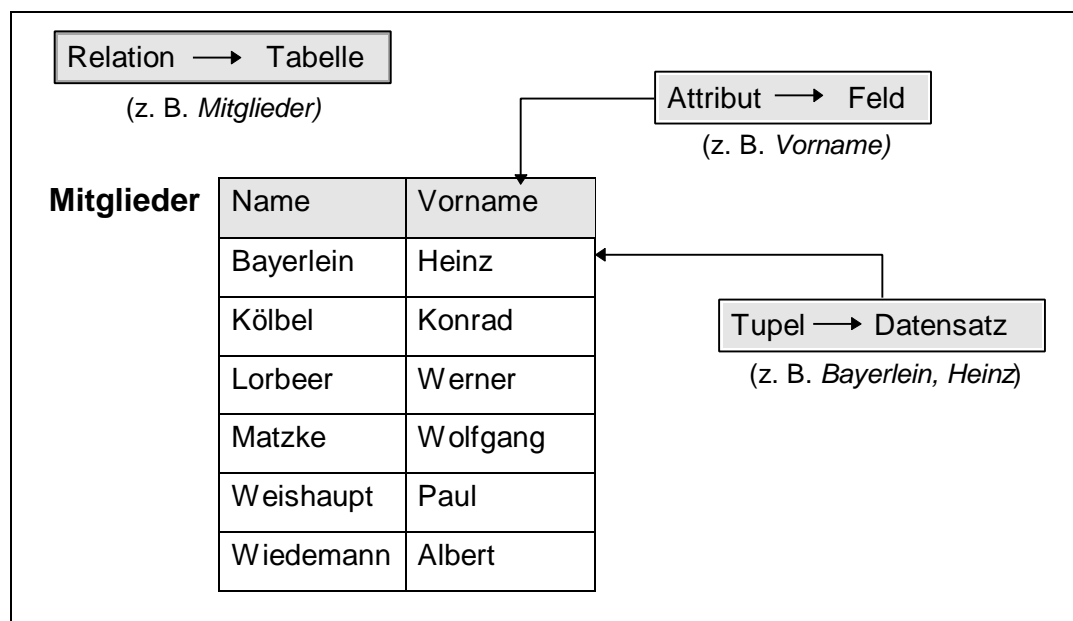


Bild 3-25: Begriffe im Zusammenhang mit Datenbanktabellen

Das relationale Modell besteht aus der Definition von *Objekten* (z. B. Relation, Tupel, Domain), *Operationen*, mit denen die Objekte bearbeitet werden können (z. B. einfügen, ändern, löschen, selektieren, projizieren, verknüpfen) und *Regeln*, die sicher stellen sollen, dass keine widersprüchlichen Daten (Anomalien, Inkonsistenzen) und Redundanzen (Mehrfachspeicherung des gleichen Sachverhalts) in der Datenbasis gespeichert sind.

### 3.4.2 Abhängigkeiten und Normalformen

#### 3.4.2.1 Sinn und Zweck von Normalformen

Die Untersuchung des Relationenmodells hat eine eigentliche Datenbanktheorie hervorgebracht, bei der die formalen Aspekte, teils losgelöst von realen Gegebenheiten, präzise beschrieben werden. Ein bedeutendes Teilgebiet dieser Datenbanktheorie bilden die sogenannten Normalformen (engl. normal forms). Mit diesen werden innerhalb von Tabellen Abhängigkeiten studiert und aufgezeigt; oft zur Vermeidung redundanter Information und von damit zusammenhängenden Anomalien.

Zur **Redundanz** eines Merkmals:

Ein Merkmal einer Tabelle ist **redundant**, wenn einzelne Werte dieses Merkmals innerhalb der Tabelle ohne Informationsverlust weggelassen werden können.

Tabelle: Lehrerdaten

<i>L-Nr</i>	<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>	<i>Besoldungsgruppe</i>
1	Bavaria	Eusebia	StRin	A13
2	Börse	Hanna	StRin	A13
3	Kalkulus	Carl-Friedrich	StD	A15
4	Spieking	Josef	OStR	A14

Tabelle 3-14: Redundante und anomalienträchtige Tabelle

Betrachtet man dazu als Beispiel eine Tabelle *Lehrerdaten*, die neben *L-Nr*, *L-Nachname*, *L-Vorname* und *Amtsbez* für jeden Lehrer noch seine *Besoldungsgruppe* enthält (vgl. Tabelle 3-14).

Aus Tabelle 3-14 ist ersichtlich, dass für jeden Lehrer neben der Amtsbezeichnung auch die zugehörige Besoldungsgruppe aufgelistet ist. Das Merkmal *Besoldungsgruppe* ist also redundant, da mehrmals ein und dieselbe Besoldungsgruppe in der Tabelle vorkommt. Es würde genügen, sich die Amtsbezeichnungen mit ihren zugehörigen Besoldungsgruppen in einer separaten Tabelle zu merken, anstatt diesen Sachverhalt für jeden Mitarbeiter redundant mitzuführen.

Bei Tabellen mit redundanter Information können sogenannte Anomalien auftreten. Möchte man z. B. aus organisatorischen Überlegungen in der Tabelle *Lehrerdaten* (Tabelle 3-14) eine neue *Besoldungsgruppe* (z. B. *A12* bzw. *A16*) definieren, so kann man diesen Sachverhalt nicht ohne weiteres in der Tabelle festhalten. Es liegt eine **Einfügeanomalie** vor, weil man keine neue Tabellenzeile ohne eine eindeutige Lehrernummer (*L-Nr*) einbringen kann.

Von einer **Löschanomalie** spricht man, wenn ein Sachverhalt ungewollt verloren geht. Falls man in der Tabelle *Lehrerdaten* sämtliche Mitarbeiter löscht, verliert man gleichzeitig die Amtsbezeichnungen mit ihren Besoldungsgruppen.

Schließlich gibt es auch **Änderungsanomalien**: Soll die Besoldungsgruppe der Amtsbezeichnung *StRin* bzw. *StR* von *A13* auf *A13L* abgeändert werden, so muss bei sämtlichen Mitarbeitern mit der entsprechenden Amtsbezeichnung diese Änderung vollzogen werden. Anders ausgedrückt: Obwohl nur ein einziger Sachverhalt eine Veränderung erfährt, muss die Tabelle *Lehrerdaten* an verschiedenen Stellen angepasst werden - dieser Nachteil wird als Änderungsanomalie bezeichnet.

In den folgenden Ausführungen werden Normalformen diskutiert, die Anomalien und damit auch Redundanzen vermeiden helfen. Die Tabelle 3-15 gibt eine Übersicht über die Normalformen und ihre Bedeutungen. Sie illustriert insbesondere, dass mit Hilfe der Normalformen verschiedene Abhängigkeiten zu Konfliktsituationen führen können.

Die Tabelle 3-15 zeigt, dass die Normalformen die Menge der zugelassenen Tabellen mehr und mehr einschränken. So muss beispielsweise eine Tabelle oder ein ganzes Datenbankschema in dritter Normalform die erste und die zweite Normalform erfüllen, und zusätzlich dürfen keine sogenannten transitiven Abhängigkeiten unter den Nichtschlüsselmerkmalen auftreten.

Beim Studium der Normalformen ist wesentlich, dass nicht alle Normalformen dieselbe Bedeutung haben. **Meistens beschränkt man sich in der Praxis auf die ersten drei Normalformen**, da Mehrwert- oder Verbundabhängigkeiten kaum in Erscheinung treten. Die vierte und die fünfte Normalform kommen somit nur in seltenen Fällen zum Zug. Sie werden aus diesem Grund nicht weiter betrachtet; es sei hierzu auf die Fachliteratur verwiesen.

Normalform	Eigenschaft
Unnormalisiert	Tabelle in beliebiger Form
Erste Normalform (1NF)	Alle Merkmalswerte sind atomar. (keine Wiederholungsgruppen zugelassen)
Zweite Normalform (2NF)	Nichtschlüsselmerkmale sind voll vom Schlüssel abhängig.
Dritte Normalform (3NF)	Es bestehen keine transitiven Abhängigkeiten.
Boyce-Codd Normalform (BCNF)	Nur Abhängigkeiten vom Schlüssel zugelassen.
Vierte Normalform (4NF)	Keine Mehrwertabhängigkeiten
Fünfte Normalform (5NF)	Nur triviale Verbundabhängigkeit

**Tabelle 3-15: Übersicht über die Normalformen und ihre Charakterisierung**

Das Verständnis für die Normalformen hilft, die im Abschnitt 3.3.2 (Tabelle 3-2) erläuterten Abbildungsregeln zu untermauern. Bei einer sauberen Definition eines Entity-Relationship-Modells und einer konsequenten Anwendung der Abbildungsregeln bleiben die Normalformen in der Regel erfüllt. Dies ist jedoch nicht automatisch der Fall. Deshalb ist es zweckmäßig, zu überprüfen, ob die Relationen den Normalformen genügen.

### 3.4.2.2 Funktionale Abhängigkeiten

Die erste Normalform bildet die Ausgangsbasis für die übrigen Normalformen und lautet:

#### Erste Normalform (1NF)

Eine Tabelle ist in erster Normalform, falls die **Wertebereiche der Merkmale atomar** sind. Die erste Normalform verlangt, dass jedes Merkmal Werte aus einem unstrukturierten Wertebereich bezieht. Somit dürfen keine Mengen, Aufzählungstypen oder Wiederholungsgruppen in den einzelnen Merkmalen vorkommen.

**Lehrer (unnormalisiert)**

<i>L Nr</i>	<i>L-Name</i>	<i>L-Vorname</i>	<i>Amtsbez</i>	<i>Klassen</i>
1	Bavaria	Eusebia	StRin	11WA, 11WB, 12TD, 12SC
3	Kalkulus	Carl-Friedrich	StD	11TC, 11TD, 12GB

**Lehrer 1NF (in erster Normalform)**

<i>L Nr</i>	<i>L-Name</i>	<i>L-Vorname</i>	<i>Amtsbez</i>	<i>Klasse</i>
1	Bavaria	Eusebia	StRin	11WA
1	Bavaria	Eusebia	StRin	11WB
1	Bavaria	Eusebia	StRin	12TD
1	Bavaria	Eusebia	StRin	12SC
3	Kalkulus	Carl-Friedrich	StD	11TC
3	Kalkulus	Carl-Friedrich	StD	11TD
3	Kalkulus	Carl-Friedrich	StD	12GB

**Lehrer (2NF)**

<i>L-Nr</i>	<i>L-Name</i>	<i>L-Vorname</i>	<i>Amtsbez</i>
1	Bavaria	Eusebia	StRin
3	Kalkulus	Carl-Friedrich	StD

**Unterricht (2NF)**

<i>L-Nr</i>	<i>Klasse</i>
1	11WA
1	11WB
1	12TD
1	12SC
3	11TC
3	11TD
3	12GB

**Bild 3-26: Tabellen in erster und zweiter Normalform**

Die Tabelle Lehrer aus Bild 3-26 ist vorerst nicht normalisiert, enthält sie doch pro Lehrtupel mehrere Klassenbezeichnungen, in welchen vom jeweiligen Lehrer zu unterrichten ist. Die unnormierte Tabelle lässt sich auf einfache Art in die erste Normalform bringen, indem für jede Klassenbelegung ein eigenes Tupel erzeugt wird. Dabei fällt auf, dass der Schlüssel der Tabelle Lehrer beim Überführen in die erste Normalform erweitert werden muss, da man für das eindeutige Identifizieren der Tupel sowohl die Lehrer- als auch die Klassenbezeichnung benötigt. Es ist üblich (aber nicht zwingend), bei zusammengesetzten Schlüsseln beide Schlüsselteile am Tabellenanfang direkt hintereinander zu zeigen (vgl. Bild 3-26). Sollte ein Lehrer aber in einer Klasse mehr als ein Fach unterrichten (gilt nur unter Berücksichtigung der Fächer als Entität), muss der Primärschlüssel aber um das Feld *Fach* erweitert werden.

Paradoxerweise erhält man bei der Anwendung der ersten Normalform eine Tabelle mit Redundanz. In Bild 3-26 sind sowohl Namen wie Vornamen der Lehrer redundant, da sie bei jeder Klassenbelegung wiederholt werden. Die zweite Normalform schafft hier Abhilfe:



## Zweite Normalform (2NF)

Eine Tabelle ist in zweiter Normalform, wenn sie in erster Normalform ist und wenn jedes Nichtschlüsselmerkmal von jedem Schlüssel **voll funktional abhängig** ist.

Ein Merkmal bzw. eine Merkmalskombination B ist **funktional abhängig** vom Merkmal bzw. der Merkmalskombination A, falls zu einem beliebigen Wert von A in der Relation höchstens ein Wert von B existiert. (Schreibweise:  $A \rightarrow B$ ).

Die funktionale Abhängigkeit B von A verlangt somit, dass jeder Wert von A auf eindeutige Weise einen Wert von B bestimmt. Alle Identifikationsschlüssel haben die Eigenschaft, dass die Nichtschlüsselmerkmale eindeutig vom Schlüssel abhängig sind. Es gilt also allgemein für einen Identifikationsschlüssel S und für ein beliebiges Merkmal B einer bestimmten Tabelle die funktionale Abhängigkeit  $S \rightarrow B$ .

Bei zusammengesetzten Schlüsseln muss man den Begriff der funktionalen Abhängigkeit zum Begriff der vollen funktionalen Abhängigkeit wie folgt erweitern:

Ein Merkmal bzw. eine Merkmalskombination B einer Relation R ist **voll funktional abhängig** vom Merkmal bzw. der Merkmalskombination A derselben Relation R, falls B funktional abhängig von A, nicht jedoch schon von einem Teil von A funktional abhängig ist. (Schreibweise:  $A \Rightarrow B$ ).

Volle funktionale Abhängigkeit besteht also dann, wenn der zusammengesetzte Schlüssel allein in seiner Gesamtheit die übrigen Nichtschlüsselattribute eindeutig bestimmt. Die volle funktionale Abhängigkeit vom Schlüssel verbietet also, dass ein Merkmal von einem Teil des Schlüssels funktional abhängig ist.

Betrachtet man die in die erste Normalform überführte Tabelle *Lehrer* (1NF) in Bild 3-26. Sie enthält den zusammengesetzten Schlüssel (L-Nr, Klasse) und muss somit auf ihre volle funktionale Abhängigkeit hin überprüft werden. Für den *L-Namen*, den *L-Vornamen* sowie die *Amtsbez* der Lehrer gelten die funktionalen Abhängigkeiten  $(L-Nr, Klasse) \rightarrow L-Name$ ,  $(L-Nr, Klasse) \rightarrow L-Vorname$  sowie  $(L-Nr, Klasse) \rightarrow Amtsbez$ . Jede Kombination der *L-Nr* mit der *Klasse* bestimmt nämlich eindeutig einen *L-Namen*, *L-Vornamen* oder die *Amtsbez*. Hingegen weiß man, dass sowohl der *Name* wie der *Vorname* bzw. die *Amtsbez* des Lehrers nichts mit den Klassen zu tun haben. Man stellt also fest, dass die drei Merkmale *L-Name*, *L-Vorname* sowie *Amtsbez* funktional abhängig sind von einem Teil des Schlüssels (d. h. es gilt  $L-Nr \rightarrow L-Name$ ,  $L-Nr \rightarrow L-Vorname$  und  $L-Nr \rightarrow Amtsbez$ ). Dies ist ein Widerspruch zur Definition der vollen funktionalen Abhängigkeit. Somit ist die Tabelle *Lehrer* 1NF nicht in zweiter Normalform.

Ist eine Tabelle mit einem zusammengesetzten Schlüssel nicht in zweiter Normalform, so muss sie in Teiltabellen zerlegt werden. Dabei fasst man die Merkmale, die von einem Teilschlüssel abhängig sind, und diesen Teilschlüssel in einer eigenständigen Tabelle zusammen. Die Resttabelle mit dem zusammengesetzten Schlüssel und evtl. weiteren Beziehungsattributen belässt man als weitere Tabelle.

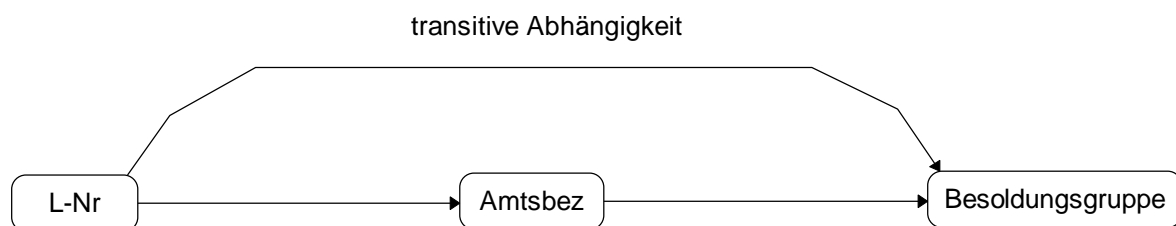
Im Beispiel aus Bild 3-26 erhält man die beiden Tabellen *Lehrer 2NF* und *Unterricht 2NF*. Beide Tabellen sind in erster und in zweiter Normalform. In der Tabelle *Lehrer 2NF* tritt kein zusammengesetzter Schlüssel mehr auf, und die zweite Normalform ist offensichtlich erfüllt. Die Tabelle *Unterricht 2NF* hat keine Nichtschlüsselattribute, womit sich eine Überprüfung der zweiten Normalform auch hier erübrigt.

### 3.4.2.3 Transitive Abhängigkeit

In Bild 3-27 ist die anfangs diskutierte Tabelle *Lehrerdaten* (2NF) gezeigt, welche neben den Lehrerdaten auch die Besoldungsgruppen enthält. Man erkennt sofort, dass sich die Tabelle in erster und zweiter Normalform befindet. Da kein zusammengesetzter Schlüssel auftritt, muss man die Eigenschaft der vollen funktionalen Abhängigkeit nicht überprüfen. Doch wie man sieht, tritt das Merkmal *Besoldungsgruppe* redundant auf. Dieser Missstand ist mit der dritten Normalform zu beheben, die in diesem Abschnitt näher erläutert werden soll.

**Lehrerdaten (2NF)**

<i>L-Nr</i>	<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>	<i>Besoldungsgruppe</i>
1	Bavaria	Eusebia	StRin	A13
2	Börse	Hanna	StRin	A13
3	Kalkulus	Carl-Friedrich	StD	A15
4	Spieking	Josef	OStR	A14



**Lehrerdaten (3NF)**

<i>L-Nr</i>	<i>L-Nachname</i>	<i>L-Vorname</i>	<i>Amtsbez</i>
1	Bavaria	Eusebia	StRin
2	Börse	Hanna	StRin
3	Kalkulus	Carl-Friedrich	StD
4	Spieking	Josef	OStR

**Amtlich (3NF)**

<i>Amtsbez k</i>	<i>Amtsbez L</i>	<i>Bes-gruppe</i>
StRin	Studienrätin	A13
StR	Studienrat	A13
OStRin	...	A14
OStR	...	A14
...	...	...

**Bild 3-27: Transitive Abhängigkeit und dritte Normalform**

### Dritte Normalform (3NF)

Eine Tabelle ist in dritter Normalform, wenn sie sich in zweiter Normalform befindet und wenn kein Nichtschlüsselattribut **transitiv abhängig** von irgendeinem Schlüssel ist.

Seien A, B und C Attribute bzw. Attributkombinationen einer Relation R, dann heißt C **transitiv abhängig** von A, wenn gilt:

$A \rightarrow B$ , d. h. B ist funktional abhängig von A, und

$B \rightarrow C$ , d. h. C ist funktional abhängig von B.

C ist also mittelbar, über B, von A abhängig.

Beispielsweise ist das Merkmal *Besoldungsgruppe* in Bild 3-27 über den Umweg *Amtsbez* von der *L-Nr* funktional abhängig. Man erkennt zwischen der *L-Nr* und der *Amtsbez* eine funktionale Abhängigkeit, sowie eine weitere zwischen *Amtsbez* und *Besoldungsgruppe*. Die beiden funktionalen Abhängigkeiten  $L-Nr \rightarrow Amtsbez$  und  $Amtsbez \rightarrow Besoldungsgruppe$  lassen sich also über die *Amtsbez* zur transitiven Abhängigkeit  $L-Nr \rightarrow Besoldungsgruppe$  zusammenfügen.

Aus dem Beispiel *Lehrerdaten* (2NF) in Bild 3-27 geht hervor, dass das Merkmal *Besoldungsgruppe* transitiv vom Merkmal *Amtsbez* abhängig ist. Deshalb ist definitionsgemäß die Tabelle *Lehrerdaten* nicht in dritter Normalform. Durch Zerlegung befreit man sie von der transitiven Abhängigkeit, indem man das redundante Merkmal *Besoldungsgruppe* mit der *Amtsbez* zusammen als eigenständige Tabelle *Amtlich* führt. Die *Amtsbez* bleibt als Fremdschlüssel mit dem Merkmalsnamen *Amtsbez* in der Resttabelle *Lehrer*. Die Beziehung zwischen *Lehrer* und *Besoldungsgruppe* ist dadurch nach wie vor gewährleistet.

#### 3.4.3 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.

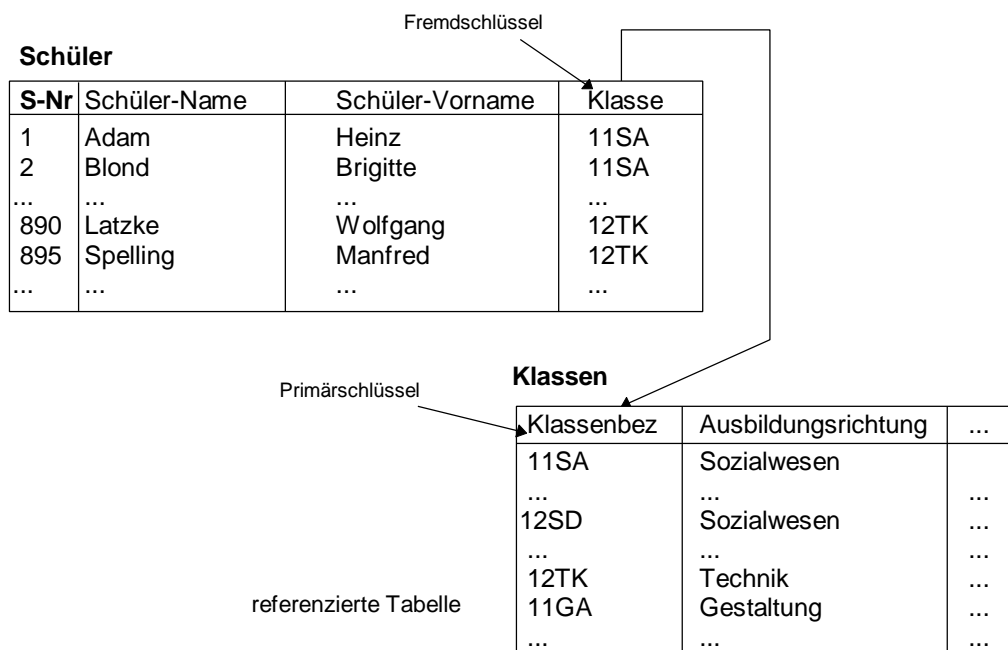
Die **Eindeutigkeitsbedingung** verlangt pro Tabelle einen festgelegten Schlüssel. Natürlich können innerhalb derselben Tabelle mehrere Schlüsselkandidaten vorkommen. In diesem Fall muss aufgrund der Eindeutigkeitsbedingung ein Schlüssel als **Primärschlüssel** deklariert werden. Das Prüfen auf Eindeutigkeit 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 3-28 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.

Nimmt man an, man möchte in die Tabelle *Schüler*, wie sie Bild 3-28 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):



**Bild 3-28: Gewährleistung der referentiellen Integrität**

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 3-28 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.

### 3.5 Zusammenfassung

Die unterschiedlichen Verfahren zur Ermittlung der Informationsstruktur haben folgende Schritte gemeinsam:

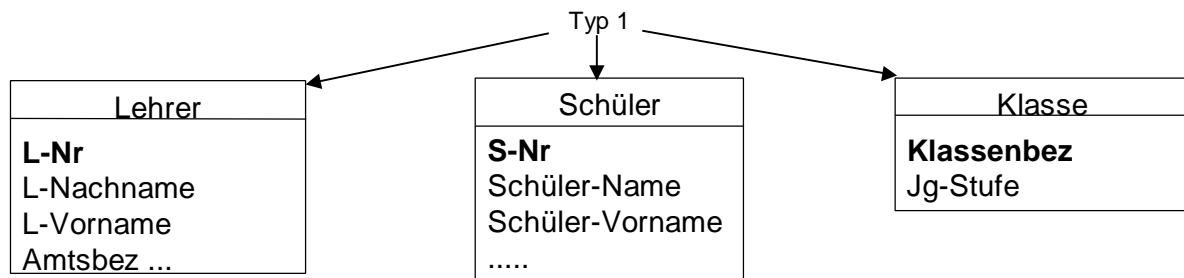
- In der Datenbank abzubildenden Realitätsausschnitt abgrenzen
- Im Ausschnitt der Welt enthaltene relevante Objekte (Entitäten) auswählen
- Beziehungen zwischen den Objekten beschreiben
- Relevante Eigenschaften von Objekten und Beziehungen bestimmen

Die ermittelte Informationsstruktur wird nun auf ein datenbankunabhängiges Modell abgebildet (konzeptionelle Phase). Das in der konzeptionellen Phase entworfene semantische Modell kann die logische Datenmodellierung erheblich erleichtern. Da die zur Zeit verbreiteten Datenbankmanagementsysteme nicht auf semantischen Datenmodellen aufbauen, kann es den logischen Entwurf nicht ersetzen.

Hinzu kommt, dass an der Konzeption und Entwicklung von Datenmodellen und Anwendungen nicht nur Informatikspezialisten mitwirken, sondern auch die Nutzer. Deshalb ist ein methodisches Vorgehen erforderlich, das dem Nutzer verständlich ist. Die logischen Datenmodelle sind jedoch formal sowie technisch ausgerichtet und deshalb dem Nichtspezialisten schwer zugänglich.

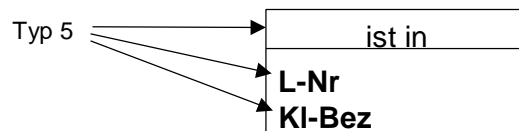
**Hinweis:** Die in der Zusammenfassung in Bild 3-29 (Seite 52) verwendeten Typen werden in Kapitel 3.3.2 Tabelle 3-2 erklärt.

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

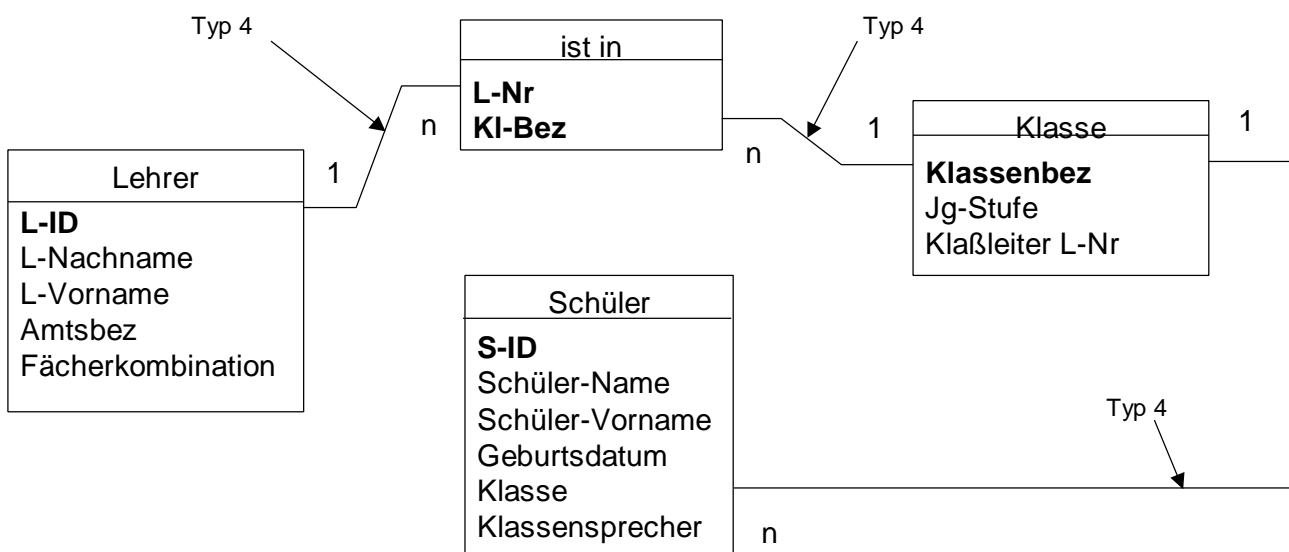


**Regel 2:** Bei komplex-komplexen Beziehungen muss eine eigenständige Beziehungsmengen-Tabelle definiert werden: die Primärschlüssel der zugehörigen Entitätsmengen treten als Fremdschlüssel in der Beziehungsmengen-Tabelle auf.

Bei konditionell-komplexen und konditionell-konditionellen Beziehungen muss nur dann eine eigene Beziehungsmengen-Tabelle erstellt werden, wenn Nullwerte vermieden werden sollen.

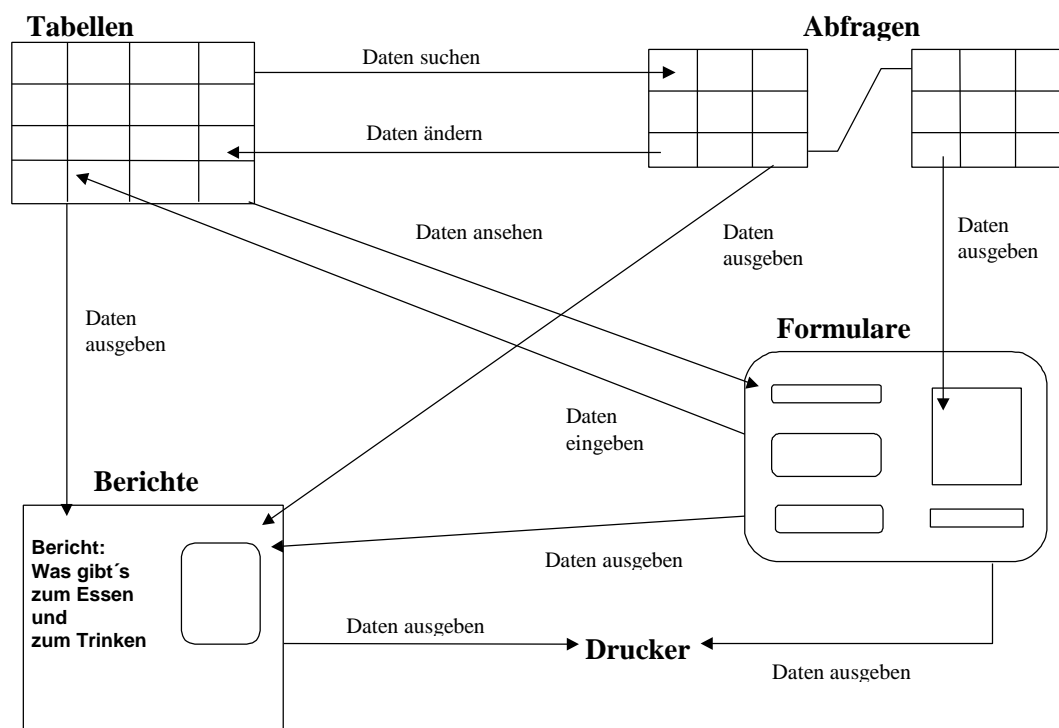


**Regel 3:** Für alle 1:1- und 1:m-Beziehungen gilt:  
Eine eigene Beziehungsmengen-Tabelle ist nicht notwendig: Aufnahme des Primärschlüssels der 1-Relation in der m-Relation.



**Bild 3-29:** Entstehung des relationalen Datenbankschemas des Beispiels Klassenverwaltung im Überblick

## 4 Realisierung des logischen Modells mit einem Softwareprodukt



**Bild 4-1: Bedeutung und Zusammenwirken der Datenbankobjekte in MS-Access**

In der physischen Phase der Datenmodellierung wird das logische Modell mit einem Datenbankmanagementsystem realisiert. Zum erfolgreichen Betreiben einer Datenbank ist eine Datenbanksprache (z. B. SQL) notwendig. Relationale Datenbanksprachen haben den Vorteil, dass man mit ein und derselben Sprache Datenbanken erstellen, Benutzerrechte vergeben oder Tabelleninhalte verändern und auswerten kann.

Nach der Funktion können folgende Sprachteile unterschieden werden:

### **Datendefinitionssprache - Data Definition Language - DDL**

Mit der DDL wird die Daten- und Aufnahmestruktur definiert. Es müssen die Tabellen und die Verknüpfungen zwischen den Tabellen aufgebaut werden. Außerdem werden damit die Namen der Tabellenspalten (Feldnamen, Attributnamen) und die Eigenschaften (z. B. Datentypen) der Felder definiert. Die Informationen darüber werden im Systemkatalog (Data Dictionary) festgehalten.

### **Datenmanipulationssprache - Data Manipulation Language - DML**

Mit der DML können Datensätze in Tabellen eingefügt (insert), geändert (update) und gelöscht (delete) werden.

### **Datenabfragesprache - Data Query Language - DQL**

Die DQL dient dem Anwender, Informationen aus der Datenbasis zu gewinnen. Er kann damit Datensätze auswählen (Selektion = Restriktion), nur bestimmte Attribute (Felder) eines Datensatz-

zes abbilden (Projektion) und Daten aus mehreren Tabellen zusammenführen (Verknüpfung = join). Selektion, Projektion und Verknüpfung können miteinander kombiniert werden.

### **Datenkontrollsprache - Data Control Language - DCL**

Für die Verwaltung der Zugangs- und Zugriffsberechtigungen von Nutzern wird die DCL eingesetzt.

### **Datenübertragungssprache - Transaction Processing Language - TPL**

Die TPL dient dazu, Daten zu sichern, zu exportieren und zu importieren.

## **4.1 Tabellen**

Die Grundfunktion einer Datenbank ist es, Daten in geordneter Form zu speichern. Die Datenbasis besteht in der Regel aus mehreren Tabellen. Jede Tabelle (**Relation**) besteht aus **Zeilen** und **Spalten**. Durch die Schnittstellen von Spalten und Zeilen entstehen **Felder**. Eine einzelne Zeile einer Tabelle enthält einen **Datensatz (Tupel)**. Ein Datensatz besteht aus mehreren **Datenfeldern** (Datenelementen). Die Datenfelder enthalten die Attributwerte der Datensätze. Tabellen enthalten alle Informationen, die in Abfragen, Formularen und Berichten ausgewertet werden können.

Aus diesem Grund wird in diesem Kapitel noch einmal auf die Planung von Tabellen eingegangen und anschließend das Anlegen, Ändern und Bearbeiten einer Tabelle in MS-ACCESS besprochen. Außerdem werden noch einmal kurz die möglichen Beziehungen zwischen Tabellen behandelt. Die Flexibilität und Effektivität einer Datenbank hängt entscheidend davon ab, wie die in den Tabellen vorhandenen Informationen miteinander verknüpft werden.

### **4.1.1 Planen von Tabellen**

Im Kapitel 3 (insbesondere im Abschnitt 3.3) wurde die Planung der gesamten Datenbank besprochen. Bevor man damit beginnen kann, Tabellen anzulegen und Daten einzugeben, muss man einige Vorüberlegungen anstellen:

- Welche Daten in der Datenbank gespeichert werden sollen und
- wie die Daten aufgeteilt werden, damit eine zweckmäßige Struktur entsteht, die möglichst den relationalen Regeln entspricht.

Beim Planen von Tabellen geht es darum, diese Struktur umzusetzen und die Einzelheiten festzulegen. Nach der Planung der Datenbank weiß man, welche Informationen in den Tabellen gespeichert werden sollen. Damit sind im Prinzip die Datenfelder und deren Namen bestimmt. Nun muss noch deren Aufnahmestruktur geplant werden:

- Felddatentypen,
- Primärschlüssel
- weitere Feldeigenschaften, wie Sekundärschlüssel, Feldlänge usw.
- Verknüpfungen zwischen den Tabellen

Mit dem Felddatentyp bestimmt man die grundlegenden Eigenschaften der Daten in einem Feld: ob es sich um Text oder Zahlen handelt, ob mit den Daten Berechnungen durchgeführt werden können oder ob die Datensätze automatisch durchnummeriert werden. Weitere Feldeigenschaften legen z. B. die Länge eines Feldes oder Gültigkeitsregeln für die Eingabe fest.



In der Praxis sollte diese Planung unabhängig vom später verwendeten Datenbankmanagementsystem erfolgen. Dessen Auswahl sollte sich nach der Eignung für die Realisierung der Planung richten.

Man kann eine Tabelle später jederzeit verändern, indem man neue Felder hinzufügt oder Felder löscht. Die Eigenschaften eines Feldes kann man ebenfalls ändern, dabei gibt es aber einige Einschränkungen, wenn in einem Feld schon Daten enthalten sind. Änderungen der Feldgröße können z. B. zu Datenverlusten führen, vor allem, wenn man die Größe verringern möchte, um Speicherplatz zu sparen.

#### 4.1.2 Datendefinition mit dem gewählten DBMS: Tabellen anlegen

Da die Menüführung in MS-ACCESS sehr klar und übersichtlich gestaltet ist, sollen an dieser Stelle nur die wichtigsten Bearbeitungsschritte hervorgehoben werden. Für die jeweiligen Details wird auf die einschlägige Fachliteratur bzw. die Handbücher verwiesen.

Nachdem der Datenbankentwurf abgeschlossen ist, können die Tabellen angelegt werden. Dabei ist beim Anlegen einer neuen Tabelle vor allem auf die Auswahl des Felddatentyps zu achten. Die gängigsten Möglichkeiten in MS-ACCESS sind in Bild 4-2 dargestellt.

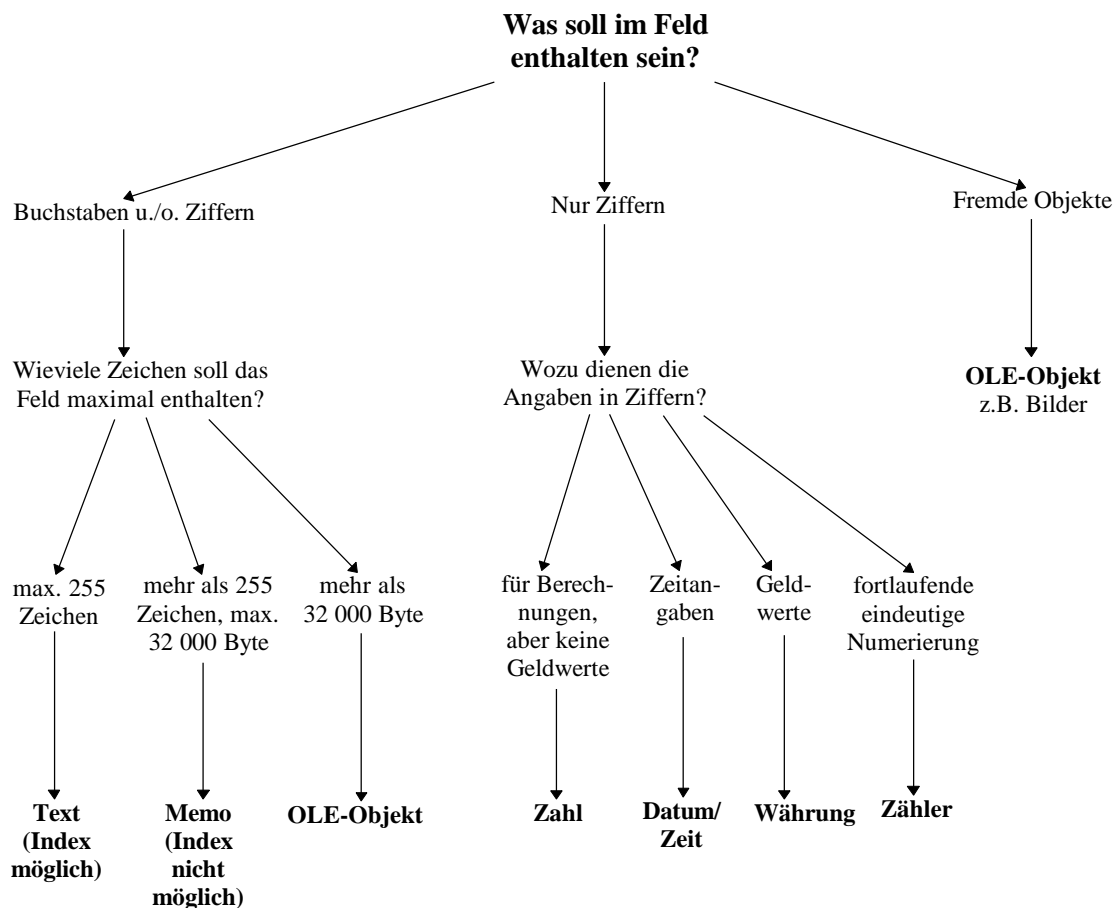


Bild 4-2: Auswahl des richtigen Felddatentyps

### 4.1.3 Funktion eines Index

Beim Sortieren der Datensätze einer Tabelle (Datei) werden alle Datensätze auf dem Datenträger umgeordnet, bis die gewünschte Reihenfolge entstanden ist. Dieses Verfahren ist vor allem bei großen Tabellen (Dateien) sehr zeitintensiv. Mit der **Indizierung** existiert ein Verfahren, das es ermöglicht, die Datensätze einer Tabelle (Datei) in eine logische Reihenfolge zu bringen, ohne dass die Datensätze physikalisch umgeordnet werden müssen. Ein weiterer Effekt dieses Verfahrens ist, dass sich die Zugriffszeit zu gesuchten Informationen verringern lässt.

Bei Datendateien mit sehr vielen Datensätzen und Merkmalen können **zeitliche Probleme** entstehen:

1. Wenn eine Tabelle sortiert werden soll, müssen sehr viele Daten bewegt werden, wenn jeder Datensatz mit allen Datenelementen umsortiert werden muss.
2. Bei der Suche nach einem bestimmten Datensatz muss die Tabelle von vorne nach hinten durchsucht werden. Der Zeitaufwand ist bei großen Tabellen erheblich.

Beim Indizieren wird eine Hilfsdatei (**Indexdatei**) erstellt, die in Verbindung mit der Tabelle (Datei) benutzt wird, die die Datensätze enthält. Eine Indexdatei kann für jedes Datenfeld einer Tabelle (Datei) aufgebaut werden. Sie beinhaltet für jeden Datensatz das Datenfeld, nach dem die Datenbank indiziert wurde, und die Position des zugehörigen Datensatzes in der Datendatei. Die Datensätze dieser Indexdatei, deren Umfang deutlich kleiner ist als der der Datendatei, werden nach dem gewünschten **Schlüssel** (= Sortierkriterium) gespeichert. Durch das Verfahren der Indizierung können auf eine Datendatei mehrere Sortierschlüssel angewendet werden, indem für jeden Schlüssel eine eigene Indexdatei erstellt wird. So ist es z. B. möglich, dass jeder berechnete Anwender blitzschnell auf die von ihm benötigten Informationen zugreifen kann, wobei beispielsweise der Fachlehrer nur die Namen seiner Schüler sehen darf, wohingegen die Schulleitung die gesamte Anschrift des Schülers benötigt.

Bei der Indizierung werden drei **Arten von Schlüsseln** unterschieden:

1. **Primärschlüssel** (Hauptindex, primary key)

Pro Tabelle ist nur ein Primärschlüssel zulässig.

Ein Wert (Attributwert) eines Datenfeldes, für das ein Primärschlüssel vergeben wurde, kann nur genau einmal in einer Datei vorhanden sein, wie beispielsweise *L-ID*.

Felder, die zum Primärschlüssel gehören, dürfen keine NULL-Werte enthalten. (Mit NULL-Wert wird ein undefinierter Feldeintrag bezeichnet, z. B. wenn nichts eingetragen wurde. Die numerische Null (0) oder ein Leerstring ("") ist **kein** NULL-Wert.)

Der Primärschlüssel kennzeichnet aufgrund der obigen Eigenschaften einen Datensatz eindeutig und trägt somit zur Integrität der Entität bei.

Es können auch mehrere Felder zusammen den Primärschlüssel bilden (Mehrfelder-Schlüssel, zusammengesetzter Schlüssel).

2. **Eindeutiger Schlüssel** (Index ohne Duplikate, unique index, not duplicate index)

Im Unterschied zum Primärschlüssel sind mehrere eindeutige Schlüssel pro Tabelle und NULL-Werte zulässig. Wie beim Primärschlüssel werden jedoch gleiche Feldwerte in einer Spalte verhindert.

### 3. Sekundärschlüssel (Index mit Duplikaten, duplicate index)

Ein Wert eines Feldes, für das ein Sekundärschlüssel vergeben wurde, kann mehrfach in der Datei vorhanden sein; beispielsweise können mehrere Lehrer in einer Lehrertabelle die gleiche Amtsbezeichnung *OSTR* haben. Spalten, deren Feldwerte häufig sortiert oder gesucht werden, sollten die Eigenschaft *Index mit Duplikaten* besitzen, damit eine schnellere Verarbeitung erfolgt.

Als Beispiel ist der Aufbau zweier Indizes für eine Lehrertabelle dargestellt. Die Lehrertabelle wurde einmal nach der *Lehrer-ID* und einmal nach dem Lehrernamen indiziert.

**Lehrertabelle**

Satz Nr.	<i>L-ID</i>	<i>Name</i>	<i>Weitere Felder</i>
1	15	Spieking	...
2	9	Kalkulus	...
3	18	Germania	...
4	12	Tüftel	...

**L-Nr-Index**

Index L-ID	L-Nr Satz-Adresse
9	2
12	4
15	1
18	3

**Name-Index**

Index Name	Name Satz-Adresse
Germania	3
Kalkulus	2
Spieking	1
Tüftel	4

#### 4.1.4 Verknüpfung von Tabellen

In MS-ACCESS sind wie bei allen relationalen DBMS nur die Beziehungstypen 1:1 und 1:n möglich.

##### 1:n-Beziehung (Eins-zu-Viele-Verknüpfung) (vgl. Bild 3-28)

Bei der Eins-zu-Viele-Verknüpfung kann die Detailtabelle (Sekundärtabelle, Kindtabelle) zu jedem Datensatz der Mastertabelle (Primärtabelle, Elterntabelle, Referenztable) mehrere Einträge haben. Beispiel: In einer Klasse können mehrere Schüler sitzen. Aber jeder Schüler kann nur einer Klasse angehören. Ein Datensatz in der Mastertabelle (hier: Klassen) kann mehr als einen passenden Datensatz in der Detailtabelle (hier: Schüler) besitzen. Ein Datensatz in der Detailtabelle kann jedoch höchstens einen passenden Datensatz in der Mastertabelle haben. Die Beziehung muss bei diesem Beispiel über das Feld *Klassenbez* hergestellt werden. Um die Beziehung festzulegen, fügt man das Feld, das den Primärschlüssel des Beziehungspartners "1" (hier: Feld *Klassenbez* in der Tabelle *Klassen*) bildet, der Tabelle des Beziehungspartners *n* (hier: Tabelle *Schüler*) hinzu. Das Feld *Klasse* ist in der Detailtabelle ein **Fremdschlüssel** (Übereinstimmungsfeld). Zweck dieser Beziehung ist es, über die *Klasse* für jeden Schüler die Daten zur Klasse zu finden (z. B. die Klassenbezeichnung oder den Klassenleiter des Schülers).

### **1:1-Beziehung (Eins-zu-Eins-Verknüpfung)**

Bei der Eins-zu-Eins-Verknüpfung enthält die Detailtabelle zu jedem Datensatz der Mastertabelle nur einen Datensatz in der Detailtabelle. Die 1:1-Verknüpfung wird oft eingesetzt, wenn Daten aus Sicherheitsgründen in einer eigenen Tabelle gespeichert werden sollen. Beispiel: Das Gehalt eines Mitarbeiters soll aus Datenschutzgründen nicht in der allgemeinen Personaldatei gespeichert werden. Eine 1:1-Beziehung ist auch für selten gebrauchte Daten zweckmäßig, wenn dadurch die Performance erhöht wird. Ein weiterer Grund kann sein, dass das Datenbankmanagementsystem nicht so viele Spalten für eine Tabelle zur Verfügung stellt, wie aufgrund der logischen Datenmodellierung notwendig wären.

Die verknüpften Schlüsselfelder müssen in Bezug auf Größe und Datentyp identisch sein. Wenn aber zwei Tabellen über ein Zählerfeld verknüpft werden, kann nur der Primärschlüssel der einen Tabelle ein Zähler sein, denn sonst müssten beide Tabellen identische Daten enthalten. Normalerweise ist die eine Tabelle der anderen Tabelle in einer Beziehung untergeordnet. In einer Tabelle *Lehrer in Klassen* kommt mit Sicherheit mehrfach die Lehrernummer vor, während in der Tabelle *Lehrer* jede *Lehrer-ID* nur einmal auftaucht. Wenn der Primärschlüssel in der Tabelle *Lehrer* ein Zählerfeld ist, muss das Datenfeld mit dem Fremdschlüssel der Tabelle *Lehrer in Klassen* den Datentyp *Zahl* und die Feldgröße *Long Integer* zugewiesen bekommen.

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

#### **Equi Join (Gleichheitsverknüpfung):**

Die Inhalte der verknüpften Felder beider Tabellen sind gleich.

#### **Outer Join (Left-Join (Inklusionsverknüpfung)):**

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.

#### **Outer Join (Right-Join (Inklusionsverknüpfung)):**

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.

#### **Auto Join**

Stellt einen Selbstbezug einer Tabelle auf sich dar.

## **4.2 Datenmanipulation**

Nachdem die Tabellen definiert wurden, können Datensätze eingegeben werden. Der Fachbegriff hierfür ist: Datensätze einfügen (insert). Im Laufe der Zeit ändern sich Daten: Z. B. wird Lehrer Meier vom Studienrat zum Oberstudienrat befördert. Der Datensatz muss geändert (update) werden. Wenn z. B. ein Lehrer die Schule verlässt, muss der Datensatz gelöscht (delete) werden.

Die Operationen der Datenmanipulation sind also:

- Datensätze **einfügen** (insert)
- Datensätze **ändern** (update)
- Datensätze **löschen** (delete)

Sie dienen der **Datenpflege**.

Gewöhnlich erfolgt die Datenmanipulation nicht direkt in den Tabellen, sondern über Formulare oder Aktionsabfragen (Einfüge-, Änderungs- und Löscharbeiten).

Die Eingabe, das Ändern und Löschen von Daten in Tabellen sowie die Verknüpfung von Tabellen müssen im Unterricht behandelt werden und sind wie oben schon erwähnt sehr benutzerfreundlich gestaltet, so dass diese Aktionen im interaktiven Dialog leicht durchgeführt werden können. Zu beachten ist lediglich, dass Änderungen bei Tabellen, die bereits Daten enthalten, zu Datenverlusten führen können. Außerdem muss eine Verknüpfung zwischen zwei Tabellen aufgelöst werden, bevor Änderungen an einem der verknüpften Felder durchgeführt werden können.

## 4.3 Datenabfrage

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 Amtsbezeichnungen von Lehrern 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 möchte man nur die Schüler einer bestimmten Klasse einsehen.

**Sortieren von Datensätzen:** Für viele Zwecke möchte man die Datensätze in einer bestimmten Reihenfolge anzeigen. So kann man z. B. die Schülerdatensätze alphabetisch geordnet nach Schülernamen 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:** Man kann mit den Daten der Tabellen Berechnungen durchführen, z. B. das Alter eines Schülers.

**Erstellen von Formularen und Berichten auf der Grundlage von Abfragen:** Damit die richtigen Daten für ein Formular oder einen Bericht ausgewählt werden, kann man eine Auswahlabfrage erstellen und dann das Formular / den Bericht auf dieser Abfrage 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 kann man z. B. auch an der ersten Abfrage kleinere Änderungen vornehmen und gleichzeitig die ursprüngliche Abfrage beibehalten

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

**Stellen von Fragen über externe Daten:** Man kann direkt Fragen über externe Daten aus anderen Datenbanken stellen, z. B. Paradox, dBASE, Btrieve, Oracle und Microsoft SQL-Server Datenbanken.

### 4.3.1 SQL

Die Sprache SQL (Structured Query Language) wurde Mitte der siebziger Jahre für *System R* geschaffen; dieses Testsystem war eines der ersten lauffähigen relationalen Datenbanksysteme. In der Zwischenzeit ist die Sprache durch die ISO normiert worden und gilt als führende Sprache auf diesem Gebiet. Darüber hinaus stellt SQL u. a. auch Sprachelemente zum Aufbau und zur Pflege einer Datenbasis zur Verfügung.

Die **Grundstruktur** dieser Sprache sieht folgendermaßen aus:

SELECT	Merkmale
FROM	Tabellen
WHERE	Selektionsprädikat

Die SELECT-Klausel entspricht der Projektion, indem sie eine Liste von Merkmalen angibt. Aus der Abfrage "Wähle Namen und Klasse eines Schülers aus", wird in SQL:

SELECT	Schüler-Name, Klasse
FROM	Schüler

In der FROM-Klausel werden alle benötigten Tabellen aufgeführt. Beispielsweise wird das kartesische Produkt zwischen den Tabellen *Schüler* und *Klassen* wie folgt in SQL dargestellt:

SELECT	S-Nr, S-Name, S-Vorname, Klasse, Ausbildungsrichtung
FROM	Schüler, Klassen

Dieser Befehl erzeugt die Resultattabelle *Schüler x Klassen* des Bildes 4-3. Formuliert man in der WHERE-Klausel das Verbundprädikat *Klasse = Klassenbez*, so erhält man den Gleichheitsverbund zwischen den Tabellen *Schüler* und *Klassen* und somit die Tabelle *Schüler x Klassen = Klassenbez*:

```

SELECT    S-ID, S-Name, S-Vorname, Klasse, Ausbildungsrichtung
FROM      Schüler, Klassen
WHERE     Klasse = Klassenbez

```

Aus den unterschiedlichen Ergebnissen der beiden Tabellen *Schüler x Klassen* und *Schüler x Klassen (mit Beziehung)* wird die Bedeutung von Beziehungen zwischen Tabellen deutlich. Insbesondere erkennt man bei diesem einfachen Beispiel sehr deutlich, dass bei Verwendung der Tabellen ohne Beziehung im Ergebnis sämtliche Kombinationsmöglichkeiten der beiden einzelnen Tabellen gebildet und ausgegeben werden.

Qualifizierte Selektionen lassen sich beschreiben, indem in der WHERE-Klausel verschiedene Aussagen durch die logischen Operatoren AND und OR verknüpft werden:

```

SELECT    *
FROM      Schüler
WHERE     Klasse = '11 A' AND Ausbildungsrichtung = 'Technik'

```

Ein "\*" in der SELECT-Klausel bedeutet, dass alle Merkmale der entsprechenden Tabellen selektiert werden; man bekommt also eine Resultattabelle mit den Merkmalen S-Nr, S-Name, S-Vorname, Klasse.

#### Schüler

<i>S-ID</i>	<i>S-Name</i>	<i>S-Vorname</i>	<i>Klasse</i>
1	Bach	Klaus	11 TA
2	Kunst	Maria	12 GA

#### Klassen

<i>Klassenbez</i>	<i>Kl-Bez</i>	<i>Jahrgangsstufe</i>
11 SA	11 A	11
11 TA	11 A	11
12 GA	12 A	12
12 WA	12 A	12

#### Schüler x Klassen (mit Beziehung)

<i>S-Nr</i>	<i>S-Name</i>	<i>S-Vorname</i>	<i>Klasse</i>	<i>Kl-Bez</i>	<i>Jahrgangsstufe</i>
1	Bach	Klaus	11 TA	11 A	11
2	Kunst	Maria	12 GA	12 A	12

#### Schüler x Klassen (ohne Beziehung)

<i>S-Nr</i>	<i>S-Name</i>	<i>S-Vorname</i>	<i>Klasse</i>	<i>Kl-Bez</i>	<i>Jahrgangsstufe</i>
1	Bach	Klaus	11 TA	11 A	11
1	Bach	Klaus	11 TA	12 A	12
1	Bach	Klaus	11 TA	12 A	12
1	Bach	Klaus	11 TA	11 A	11
2	Kunst	Maria	12 GA	11 A	11
2	Kunst	Maria	12 GA	12 A	12
2	Kunst	Maria	12 GA	12 A	12
2	Kunst	Maria	12 GA	11 A	11

**Bild 4-3: Verbund zweier Tabellen mit und ohne Verknüpfung**

### 4.3.2 QBE (Query By Example)

QBE ist eine Datenbanksprache, bei der der Benutzer seine Auswertungsvorstellungen in Form einer Beispieldatenbank entwerfen kann. Zuerst wird (werden) die für die Abfrage notwendige(n) Tabelle(n) ausgewählt. Gleichzeitig stellt das System ein Gerüst der Beispieldatenbank bereit, in die der Benutzer die gewünschten Selektionen aus der (den) Beispieldatenbank(e)n einträgt (vgl. die Beispiele in den nachfolgenden Abschnitten 4.3.2.1 und 4.3.2.2).

Für die Anfragen in den genannten Abschnitten wird die Datenbank (in der 3NF) mit der folgenden Tabellenstruktur zugrundegelegt (vgl. auch das zugehörige E-R-Diagramm in Bild 3-23 sowie das entsprechende relationale Modell in Bild 3-24). Dabei sind die Primärschlüssel durch einen "\*" gekennzeichnet:

#### Klassen

Pk	Merkmal	Datentyp	Eigenschaft	Indizierung
*	Klassenbez	Text	5	Ja (ohne Duplikate)
	Jg-Stufe	Zahl	Byte	Ja (Duplikate möglich)
	Klassenleiter-Nr	Zahl	Long Integer	Ja (Duplikate möglich)

#### Lehrer

Pk	Merkmal	Datentyp	Eigenschaft	Indizierung
*	L-ID	Zähler		Ja (ohne Duplikate)
	L-Nachname	Text	50	Ja (Duplikate möglich)
	L-Vorname	Text	20	Nein
	Amtsbez	Text	10	Ja (Duplikate möglich)
	Fächerkombination	Text	10	Ja (Duplikate möglich)

#### Schüler

Pk	Merkmal	Datentyp	Eigenschaft	Indizierung
*	S-ID	Zähler		Ja (ohne Duplikate)
	S-Name	Text	50	Ja (Duplikate möglich)
	S-Vorname	Text	20	Ja (Duplikate möglich)
	Geburtsdatum	Datum/Zeit	Datum, kurz	Ja (Duplikate möglich)
	Klasse	Text	5	Ja (Duplikate möglich)
	Klassensprecher	Ja/Nein		Ja (Duplikate möglich)

#### ist in

Pk	Merkmal	Datentyp	Eigenschaft	Indizierung
*	L-Nr	Zahl	Long Integer <> Null	Ja (Duplikate möglich)
*	Kl-Bez	Text	5	Ja (Duplikate möglich)



### 4.3.2.1 Elementare Anfragen an die Datenbank

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

Als Grundoperation auf Tabellen kommt die Projektion in Betracht. Bei der Projektion werden Spalten aus einer Tabelle gestrichen. Übrig bleiben die gewünschten Spalten. Will man die Namen und Vornamen aller Lehrer ermitteln, so streicht man in der Tabelle *Lehrer* alle Spalten bis auf die Spalten Name und Vorname und es ergibt sich die gewünschte Tabelle.

**Beispiel 1:** Ermitteln der Lehrer-ID, der Namen und der Vornamen aller Lehrer, deren Familienname mit einem „B“ beginnt.

#### Entwurfssprache:

Tabelle(n):	Lehrer	
Felder:	alle	← Projektion
wobei (Auswahlkriterien)	der Anfangsbuchstabe des Familiennamens „B“ ist.	← Selektion
Sortiert nach:	Lehrer-Name, Lehrer-Vorname	← Sortierung

#### Beispieltabelle (QBE): Tabelle *Alle Lehrer, deren Nachname mit „B“ beginnt*

Feld:	L-ID	L-Name	L-Vorname	Amtsbez
Sortierung:		Aufsteigend	Aufsteigend	
Anzeigen:	x	x	x	x
Kriterien:		Wie "B*"		

**Beispiel 2:** Ermitteln der Lehrer-ID, der Namen, der Vornamen und der Amtsbez aller Lehrer, die die Lehrbefähigung im Fach Mathematik (M) haben.

#### Entwurfssprache:

Tabelle(n):	Lehrer	
Felder:	alle	← Projektion
wobei (Auswahlkriterien)	In der Fächerkombination kommt „M“ vor.	← Selektion
Sortiert nach:	L-Name, L-Vorname	← Sortierung

#### Beispieltabelle (QBE): Tabelle *Alle Lehrer, die das Fach „M“ unterrichten*

Feld:	L-ID	L-Name	L-Vorname	Amtsbez	Fächerkombination
Sortierung:		Aufsteigend	Aufsteigend		
Anzeigen:	x	x	x	x	x
Kriterien:					(Wie "M/*" Oder Wie "*/M*") Und Nicht Wie "*Mu*"

**Beispiel 3:** Ermitteln der Schülernamen, der Vornamen, des Geburtsdatums, des Alters und der Klasse aller Schüler.

**Entwurfssprache:**

Tabelle(n):	Schüler	
Felder:	S-Name, S-Vorname, Geburtsdatum, Alter, Klasse	← Projektion
wobei (Auswahlkriterien)		← Selektion
Sortiert nach:	Alter	← Sortierung

**Beispieltabelle (QBE):** Tabelle *Alle Schüler und ihr Alter*

Feld:	S-Name	S-Vorname	Geb.-dat.	Alter: Int((Datum() - [Geburtsdatum])/365,25)	Klasse
Sortierung:	Aufsteigend	Aufsteigend			
Anzeigen:	x	x	x	x	x
Kriterien:					

#### 4.3.2.2 Komplexe Anfragen an die Datenbank

In diesem Abschnitt wird die „teuerste“ (zeitaufwendigste) Operation in einem relationalen Datenbanksystem dargestellt: Die Verbindung von Tabellen zu einer neuen Tabelle, der **Join**.

**Beispiel 4:** Von jedem Schüler soll die Schüler-ID, der Familien- und Vornamen, die Klasse und die Jahrgangsstufe ermittelt werden.

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

Gibt man die Beispieltabelle aus, ohne vorher bestimmte Merkmale der beiden Ausgangstabellen *Schüler* und *Klassen* miteinander zu verknüpfen, so werden alle möglichen Kombinationen zwischen den beiden Tabellen *Schüler* und *Klassen* gebildet. So erhält man in diesem Fall (bei 100 Schülern und 4 Ausbildungsrichtungen)  $4 \times 100 = 400$  Datensätze der Beispieltabelle, anstatt der erwarteten 100 Datensätze für die 100 Schüler. Die Verknüpfung der beiden Beispieltabellen erfolgt über die Attribute *Klassen.Klassenbez* und *Schüler.Klasse* (1:n-Beziehung).

**Entwurfssprache:**

Tabelle(n):	Schüler, Klassen Schüler.Klasse = Klassen.Klassenbez	← Join
Felder:	S-Nr, S-Name, S-Vorname, Klasse, Jahrgangsstufe	← Projektion
wobei (Auswahlkriterien)		← Selektion
Sortiert nach:	S-Name	← Sortierung

**Beispieltabelle (QBE):** Tabellen: *Schüler und zugehörige Klassen*

Feld:	S-ID	S-Name	S-Vorname	Klasse	Jahrgangsstufe
Sortierung:		Aufsteigend			
Anzeigen:	x	x	x	x	x
Kriterien:					

In SQL-Notation sieht die Abfrage wie folgt aus:

```
SELECT      S-ID, S-Name, S-Vorname, Klasse, Ausbildungsrichtung
FROM        Klassen INNER JOIN Schüler ON Klassen.Klassenbez = Schüler.Klasse
ORDER BY    Schüler.[S-Name]
```

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

**Beispiel 5:** Ermitteln aller Schüler einer bestimmten Klasse (nach Eingabe der Klassenbez), mit Nachnamen, Vornamen, Klassenbezeichnung sowie Nachname des Klassenleiters.

**Entwurfssprache:**

Tabelle(n):	Schüler, Klassen, Lehrer Schüler.Klasse = Klassen.Klassenbez Lehrer.L-ID = Klassen.Klassenleiter-Nr	
Felder:	S-Name, S-Vorname, Klasse, L-Name	← Projektion
wobei (Auswahlkriterien)	[Bitte geben Sie die Klasse ein (z. B. 11A):]	← Selektion
Sortiert nach:	S-Name	← Sortierung

**Beispieltabelle (QBE):** Tabelle *Alle Schüler einer Klasse (Klassenbez als Parameter)*

Feld:	S-Name	S-Vorname	Klasse	Klassenleiter: L-Name
Sortierung:	Aufsteigend	Aufsteigend		
Anzeigen:	x	x	x	x
Kriterien:			[Bitte geben Sie die Klasse ein (z. B. 11A):]	

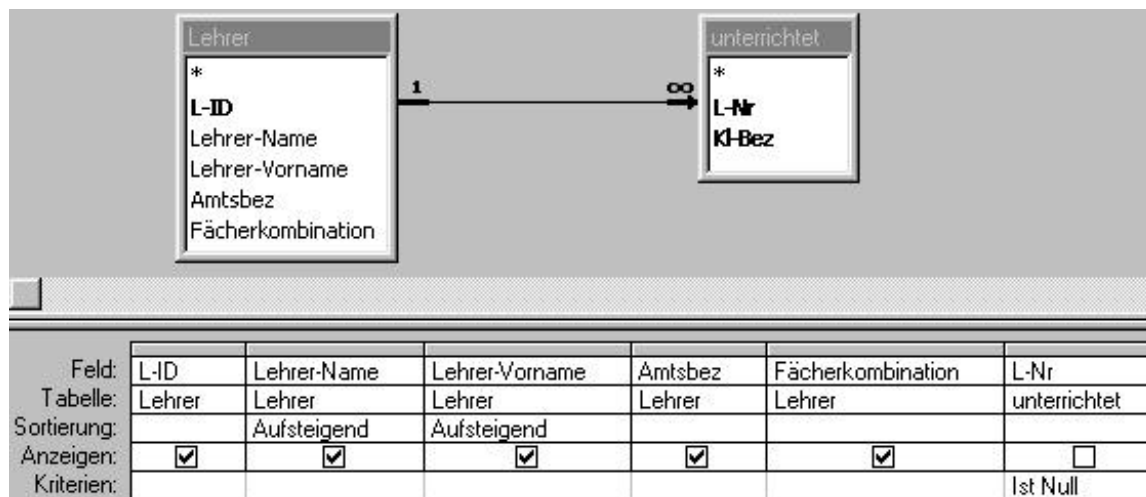
**Beispiel 6:** Ermitteln der Lehrer-ID, des Nachnamens, Vornamens und der Amtsbez derjenigen Lehrer, denen noch keine Klasse zugeteilt wurde.

**Entwurfssprache:**

Tabelle(n):	Lehrer, ist in Outer-Join: Lehrer.[L-ID] = [ist in]. [L-Nr]	
Felder:	alle	← Projektion
wobei (Auswahlkriterien)	L-Nr ist Null	← Selektion
Sortiert nach:	L-Name, L-Vorname	← Sortierung

**Beispieltabelle (QBE):** Tabelle *Alle Lehrer ohne Unterrichtseinsatz*

Feld:	L-ID	L-Name	L-Vorname	Amtsbez	Fächerkombination	L-Nr
Sortierung:		Aufsteigend	Aufsteigend			
Anzeigen:	x	x	x	x	x	
Kriterien:						Ist Null



**Beispiel 7:** Ermitteln aller Schüler, die in der gleichen Klasse wie ein bestimmter Schüler sind.  
Nachnamen, Vornamen, Klassenbezeichnung.

**Entwurfssprache:**

Tabelle(n):	Schüler, Schüler_1 Schüler.Klasse = Schüler_1.Klasse	
Felder:	S-Name, S-Vorname, Klasse	← Projektion
wobei (Auswahlkriterien)	Schüler_1.Name = „Himmel“	← Selektion
Sortiert nach:	S-Name	← Sortierung

**Beispieltabelle (QBE):** Tabelle *Ein bestimmter Schüler und seine Klassenkameraden*

Feld:	S-Name	S-Vorname	Klasse	Schüler-Name
Tabellenname:	Schüler	Schüler	Schüler	Schüler_1
Sortierung:	Aufsteigend	Aufsteigend		
Anzeigen:	x	x	x	
Kriterien:				„Himmel“

Feld:	Schüler-Name	Schüler-Vorname	Klasse	Schüler-Name
Tabelle:	Schüler	Schüler	Schüler	Schüler_1
Sortierung:	Aufsteigend	Aufsteigend		
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:				"Himmel"

**4.3.3 Verschiedene Abfragearten in MS-ACCESS**

Die Abfragen in Abschnitt 4.2 waren **Auswahlabfragen**, die Abfrageart, die zum Abrufen von Daten aus Tabellen verwendet wird. Wenn man eine Auswahlabfrage ausführt, ruft das DBMS die Daten ab und zeigt sie im Dynaset (Abfrageergebnis in Datenblattansicht) an. In den meisten Fällen kann man dann die Daten des Dynaset ändern, um sie damit in den zugehörigen Tabellen zu aktualisieren.

Auswahlabfragen werden verwendet, um Fragen zu den vorhandenen Daten zu stellen und das Ergebnis im Dynaset zu analysieren. Auswahlabfragen können auch als Grundlage für Formulare, Berichte und Diagramme verwendet werden (weitere Hinweise siehe Handbuch).

Es kann vorkommen, dass man umfangreiche Änderungen an seinen Daten vornehmen möchte - Änderungen, die sich auf eine Gruppe von Datensätzen in ähnlicher Weise auswirken. Angenommen, man will den Preis aller Artikel innerhalb einer bestimmten Artikelgruppe um 10 % erhöhen und dabei möglichst das mühselige Errechnen jeder einzelnen Änderung und das Aktualisieren jedes einzelnen Datensatzes vermeiden. Da sich die Änderung auf alle Datensätze zum gleichen Prozentsatz auswirkt, kann man die Berechnung und Aktualisierung in einem Arbeitsgang durch das DBMS durchführen lassen. Hierzu erstellt man eine **Aktionsabfrage**.

Mit Aktionsabfragen kann man entweder neue Tabellen erstellen oder die Daten in bereits vorhandenen Tabellen modifizieren. MS-ACCESS stellt vier Arten von Aktionsabfragen zur Verfügung:

- *Kreuztabellenabfrage*, fasst Daten in einem Zeilen-und-Spalten-Format zusammen.
- Eine *Tabellenerstellungsabfrage* erstellt aus anderen Tabellen oder aus Teilen anderer Tabellen eine neue Tabelle.
- Eine *Löschabfrage* löscht Datensätze aus einer oder aus mehreren Tabellen.
- Eine *Anfügeabfrage* fügt eine Gruppe von Datensätzen an eine andere Tabelle an.
- Eine *Aktualisierungsabfrage* ändert Daten in einer Gruppe von Datensätzen.

Da einige dieser Abfragearten auf Auswahlabfragen aufbauen, werden die näheren Details an dieser Stelle nicht ausgeführt, sondern es wird auf die einschlägigen Handbücher verwiesen. Lediglich ein Beispiel für eine Kreuztabelle soll hier angegeben werden.

**Beispiel 8:** Ermitteln sie pro Klasse die Anzahl aller Schüler jeder Altersgruppe.

**Entwurfssprache:**

Tabelle(n) bzw. Abfrage:	Klassen, alle Schüler und ihr Alter Klassenbez = Klasse	
Felder:	Alter, Klassenbez, Alter (Anzahl)	← Projektion
wobei:		← Selektion
Sortiert nach:	Jg-stufe	← Sortierung

**Beispieltabelle (QBE):** Tabelle *Altersgruppen pro Klasse*

Feld:	Klassenbez	Alter	Alter	Jg-stufe
Funktion	Gruppierung	Gruppierung	Anzahl	Gruppierung
Kreuztabelle	Zeilenüberschrift	Spaltenüberschrift	Tabelleneintrag	
Sortierung:	Aufsteigend	Aufsteigend		Aufsteigend
Anzeigen:	x	x	x	

	Klassenbez	10	11	15	16	17	18
► 5A		6	22				
5B		6	22				
11A				1	4	15	6
11B					2	20	1

## 4.4 Formulare

Bei der bisherigen Arbeit konnte man in einer Tabelle<sup>2</sup> mehrere Datensätze gleichzeitig einsehen und bearbeiten. Es gibt jedoch Situationen, in denen die Darstellung der Datensätze in Tabellenform nicht optimal ist. Häufig wird man auch nur mit einem Datensatz arbeiten wollen.

- Bei der Neueingabe von Datensätzen z. B. wird man aus Gründen der Übersicht häufig nur einen Datensatz mit den zugehörigen Daten, die in verschiedenen Tabellen abgespeichert sein können, betrachten wollen.
- In einem anderen Fall beschreibt eine Grafik einen kompletten Datenbestand weit besser, als jede Zahlentabelle.
- Es kann aber auch die tabellarische Anordnung die geeignete Wahl sein, falls die Informationen klar und übersichtlich auf dem Bildschirm zu erkennen sind.

Dies alles und einiges mehr kann man mit Formularen erreichen. Formulare kennt man aus dem persönlichen Erfahrungsbereich. Formulare sind meist aus Papier erstellte Eingabehilfen für die Erfassung strukturierter Daten (z. B. Karteikarten). In dem Programm *MS-ACCESS* wird zwischen vier verschiedenen Formularformen unterschieden. Jede Formularform erfüllt einen besonderen Zweck:

- Einspaltig
- Tabelle
- Diagramm
- Haupt / Unterformular

Die einspaltige Formularform stellt jeweils einen Datensatz aus einer Tabelle (oder Abfrage) dar. Diese Formularform ähnelt sehr den bekannten Karteikarten. Das tabellarisch angeordnete Formular verwendet man, wenn es um Listen oder Aufstellungen geht. Die zweite Form braucht wohl nicht näher besprochen zu werden. Die dritte Form ist eine besondere Formularform, mit der man numerische Werte veranschaulichen kann. Je nach Verwendungszweck wählt man zwischen Balken-, Kreis- oder Linienform. Die vierte Möglichkeit verwendet man, wenn in einem Formular Daten unterschiedlicher Herkunft (z. B. verschiedene Tabellen) erfasst werden müssen. Rechnungs- oder Eingabeformulare sind dafür Beispiele. Für die Gestaltung dieser Formulare stellt *MS-ACCESS* vier Assistenten (und zusätzlich drei für die Gestaltung von Berichten) zur Verfügung.

Die Assistenten erstellen zu jedem Formulartyp ein Grundmuster. Dieses Muster kann man vielfältig verfeinern, indem man in der Entwurfsansicht die gewünschten Optionen ändert.

---

<sup>2</sup> In der Literatur werden Tabellen als Geschäftsobjekte und Formulare und Berichte als Darstellungsobjekte bezeichnet.

#### 4.4.1 Ein Formular erstellen

In diesem Abschnitt soll ein neues Formular entworfen werden. Das heißt jedoch nicht, dass auf die Hilfe des Formular-Assistenten verzichtet werden soll. Es ist allemal einfacher, an einem vorbereiteten Formular Änderungen vorzunehmen, als ganz von vorn zu beginnen. Zumindest in der Anfangsphase, wenn man noch nicht sehr vertraut mit dem Hilfsmittel *Formular* umgehen kann, sind die Formularassistenten ein wichtiges Hilfsmittel. Das gleiche gilt auch für die in Abschnitt 4.5 behandelten Berichte.

Um in MS-ACCESS ein neues Formular zu erstellen, müssen die folgenden Schritte gewählt werden:

1. Öffnen des Datenbankfensters und drücken der Schaltfläche *Formular*. Anschließend wählen der Option *Neu*.
2. Im Dialogfenster *Neues Formular* diejenige Tabelle markieren, die die Daten enthält, die in das Formular übernommen werden sollen. Sollte noch keine derartige Tabelle zur Verfügung stehen, muss vorher z. B. durch eine Abfrage eine Tabelle der gewünschten Form bereitgestellt werden.
3. Aus der Assistentenliste die gewünschte Formularform auswählen.
4. Die folgenden Dialoge sind im interaktiven Modus anschaulich genug und müssen nicht ausführlich erklärt werden. Wenn man die Absicht hat, ein Formular später selbst zu verfeinern, sollte man alle diejenigen Datenfelder auch in das vom Assistenten zu erstellende Formular aufnehmen, die man später benötigt.

Ein Formular kann insgesamt in drei Ansichten betrachtet werden:

- Entwurfsansicht
- Formularansicht
- Datenblattansicht

In Datenblattansicht ähnelt das Formular einer vertikalen Abfrage. Es erscheinen in dem Datenblatt nur die Datenfelder, die auch im Formular vorkommen.

An dem Formular kann man **nur in der Entwurfsansicht Veränderungen vornehmen**. Für die Gestaltung der Formulare stehen insgesamt sechs verschiedene Werkzeuge zur Verfügung:

- Toolbox
- Eigenschaftenfenster
- Feldliste
- Farbpalette
- Lineale
- Raster

Die Werkzeuge kann man über eine Schaltfläche (*Eigenschaften*, *Feldliste* und *Farbpalette*) oder das *Ansicht-Menü* ein- und ausschalten.

**Hinweis:** Jeweils den gegenwärtigen Stand der Arbeit absichern. Dazu wählt man beim ersten Mal den Befehl *Datei speichern unter* im Datei-Menü. Danach sichert man regelmäßig den Fortgang der Arbeit mit dem Befehl *Datei speichern*.



#### 4.4.2 Die Gestaltungsbereiche

Das Entwerfen eines Formulars ist mit dem Zeichnen einer Karteikarte vergleichbar. Man wählt ein bestimmtes Format und legt fest, wohin Überschrift, Eingabestellen und eventuell erklärende Hinweise geschrieben werden sollen. Man kann dabei ein Formular so planen, dass es nicht nur den Ansprüchen auf dem Bildschirm gerecht wird, sondern auch auf dem Ausdruck.

Man beginnt i. a. mit dem Hintergrund, darin können maximal 5 Bereiche eingerichtet werden:

- Formularkopf
- Seitenkopf
- Detailbereich
- Seitenfuß
- Formularfuß

Außer dem Detailbereich sind alle anderen Bereiche optional. Wenn man ein Formular mit einem Assistenten erstellt, bleiben die Bereiche Seitenkopf und Seitenfuß zunächst unberücksichtigt.

- Der **Formularkopf** enthält die Beschriftung des Formulars. Er ist fest am oberen Bildschirmrand der Formularansicht verankert. Bei einem Ausdruck erscheint der Formularkopf einmalig am oberen Rand auf der ersten Seite. Der Bereich **Formularfuß** übernimmt die gleichen Aufgaben am unteren Fensterrand bzw. auf der letzten Druckseite.
- Im **Detailbereich** befinden sich die ausgewählten Datenfelder der Tabelle. Dieser Bereich wiederholt sich für jeden Datensatz.
- Der **Seitenkopf** wird nur bei einem Ausdruck sichtbar und wiederholt sich auf jeder Seite. Er eignet sich für Seitenüberschriften. Das Gegenstück ist der Bereich **Seitenfuß**; er könnte beispielsweise die Seitennummerierung übernehmen.

#### 4.4.3 Steuerelemente und deren Eigenschaften

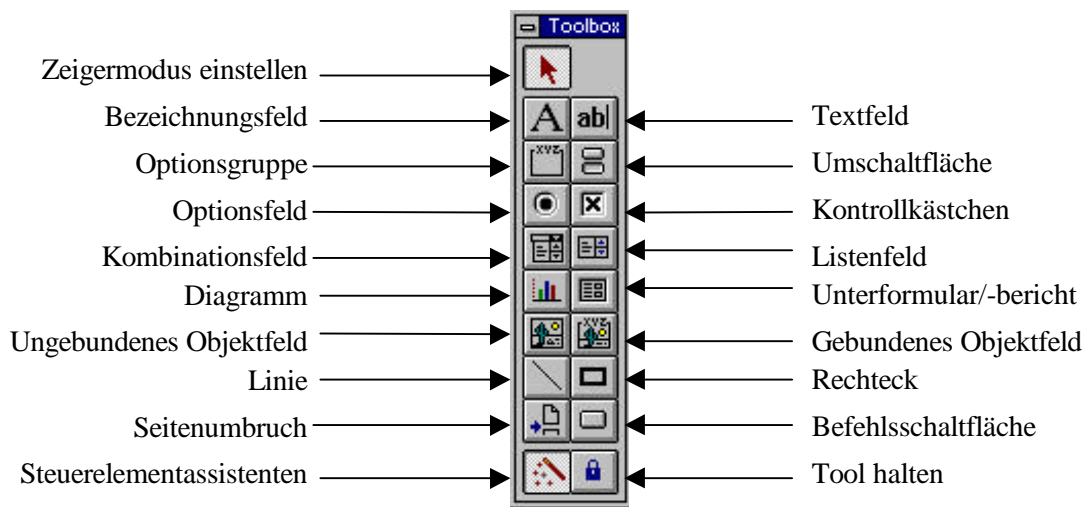
Die in 4.4.2 beschriebenen Bereiche nehmen die Steuerelemente des Formulars auf. Mit *Steuerelementen* werden von MS-ACCESS alle Bestandteile des Formulars bezeichnet. Sie lassen sich grob in folgende drei Klassen einteilen:

- gebundene Steuerelemente
- ungebundene Steuerelemente
- grafische Elemente

Die meisten Steuerelemente setzen sich aus einem **variablen** und einem **festen Bestandteil** zusammen. Es handelt sich um die beiden Teile jeder Datentabelle. Das Datenfeld (genauer dessen Wert) ist der variable Teil des Steuerelements und wird mit Textfeld bezeichnet.

Zu seiner Identifizierung erhält das Datenfeld einen Feldnamen; dieser feststehende Teil wird Bezeichnungsfeld genannt. In einem Formular können Bezeichnungsfelder auch allein vorkommen, wenn sie z. B. als Überschriften oder erklärende Hinweise verwendet werden. Von einem gebundenen Steuerelement spricht man, wenn es eine Verbindung zur Datentabelle hat. Ein Textfeld ist also immer ein gebundenes Steuerelement, ein allein verwendetes Bezeichnungsfeld ein ungebundenes Steuerelement.

Einen Überblick über alle zur Verfügung stehenden Steuerelemente liefert die *Toolbox*. Die Toolbox hat keine eigene Schaltfläche in der Symbolleiste und muss mit dem *Befehl Toolbox im Ansicht-Menü* eingeschaltet werden.



**Bild 4-4: Die Toolbox**

Die Elemente der Toolbox lernt man am besten im Gebrauch kennen. Die Reihenfolge des Vorgehens wird dabei nicht durch die Anordnung der Elemente in der Toolbox bestimmt, diese ist eher zufällig. Vielmehr werden die Elemente aus dem Arbeitsablauf heraus erklärt und eingesetzt. Dabei ist es auch hier günstig, wenn man ein Formular zunächst mit Hilfe eines Assistenten erstellt, und dann gegebenenfalls die gewünschten Änderungen vornimmt.

#### 4.4.4 Weitere Hinweise

Beim Ausdruck eines Formulars kann man zwischen einem Ausdruck in der Datenblattansicht und der Formularansicht wählen. Welche Form gedruckt wird, hängt davon ab, in welcher Ansicht gerade die Daten betrachtet werden. Wenn z. B. die Formularansicht eingeschaltet ist, erhält man die Informationen auch in dieser Ansicht ausgedruckt.

Außer dem schon besprochenen Formulartyp, gibt es noch die Möglichkeit, Diagramme oder Unterformulare zu erzeugen.

- **Diagramme** bilden ein wichtiges Element zur Auswertung numerischer Werte. Will man z. B. Tabellenwerte vergleichen, so lässt sich das sehr anschaulich mit einem Säulendiagramm tun. Voraussetzung für ein Diagramm ist eine Abfrage. Insgesamt kann man aus elf verschiedenen Diagrammformen auswählen, die sich in folgende Gruppen einteilen lassen: Liniendiagramm, Balken- und Säulendiagramm, Flächendiagramm, Kreisdiagramm, Spannweitendiagramm.
- Das Grundprinzip einer relationalen Datenbank ist die Verteilung der Daten auf mehrere Tabellen. Wenn man in einem Formular ein **Unterformular** verwendet, kann man Informationen aus mehreren Tabellen gleichzeitig einsehen. Haupt- und Unterformular(e) werden so miteinander verknüpft, dass im Unterformular nur Daten erscheinen, die mit den Informationen im Hauptformular zu tun haben.

#### 4.4.5 Beispiel für die Erstellung eines Formulars mit Unterformular

Als Beispiel für den Aufbau und die Erstellung eines benutzerfreundlichen Formulars sollen an dieser Stelle die wesentlichen Schritte zur Konstruktion des folgenden Formulars mit Unterformular durchgeführt werden. Das Formular dient der Eingabe und Pflege der Klassendaten sowie der zugehörigen Lehrer:

**Klassen und zugehörige Lehrer**

**Jg-Stufe:**  
5

**Klassenbez:**  
5A

**Klassenleiter:**  
Nenner

**Eingabe eines neuen Lehrers**

	Lehrer-Name	Lehrer-Vorname	Amtsbez	Fächerkombination
▶	Spiekings	Josef	OSTR	D/E
	Nenner	Marie	StRin	Ph/M/Inf
	Spring	Jan	StR	Sm/C/B
	Hüpf	Johanna	OSTRin	Sw/E
	Seel	Fridolin	OSTR	Ev/Eth
	Blume	Eva	OSTRin	C/B/Sk
	Maler	Ludwig	OSTR	Ku
	Tralala	Amadeus	OSTR	Mu
*				

**Datensatznavigation**

**Erster** **Vorheriger** **Nächster** **Letzter** **Neuer** **Suchen**

**Bild 4-5: Hauptformular mit Unterformular (Formularansicht)**

Wie aus dem logischen Modell (Bild 3-24) erkennbar ist, besteht zwischen den Tabellen *Lehrer* und *Klassen* eine n-m-Beziehung, die durch die Beziehungsmengentabelle *ist in* in zwei 1-n-Beziehungen aufgelöst wird. D. h. jedem Datensatz in der Tabelle *Klassen* können mehrere Datensätze in der Tabelle *Lehrer* zugeordnet werden und umgekehrt. Damit diese Zuordnung benutzerfreundlich von einer Stelle aus durchgeführt werden kann, konstruiert man das obige Formular. Dabei muss entschieden werden, welche Seite der n-m-Beziehung Vorrang hat:

- Sollen jedem einzelnen Lehrer seine Klassen zugeordnet werden, dann stehen die Lehrerdaten im Hauptformular und die Klassendaten im Unterformular.
- Sollen, wie oben, jeder Klasse die entsprechenden Lehrer zugeordnet werden, dann stehen die Klassendaten im Hauptformular und die Lehrerdaten im Unterformular.

#### 4.4.5.1 Erstellung des Hauptformulars

Sämtliche Daten für die Erstellung des Hauptformular befinden sich in der Tabelle *Klassen*. Infolgedessen kann diese Tabelle als Datenherkunft für das Hauptformular dienen.

Bild 4-6: Hauptformular mit Unterformular (Entwurfsansicht)

1. Man geht in das Formularfenster der Datenbank und wählt die Option *Neu*.
2. In dem sich anschließenden Menü wird die Option *Formular-Assistent* und in dem Tabellen/Abfrage-Auswahlfenster die Tabelle *Klassen* ausgewählt.
3. Im Anschluss daran wird der Benutzer gefragt, welche Felder in das Formular übernommen werden sollen: hier werden mit Hilfe des Doppelpfeils alle Felder ausgewählt.
4. Nun wird der Benutzer nach dem *Formular-Layout* gefragt. Da das zu erstellende Formular ein Hauptformular wird, soll jeder Datensatz (entspricht einer Klasse) auf einer eigenen Seite stehen; folglich wird als Layout *Einspaltig* ausgewählt.
5. Die nächste Frage zielt auf den *Formular-Stil*. Hier sollte eine möglichst sachliche Form (z. B. *Standard*) gewählt werden. Nachbesserungen mit unterschiedlichen Farben können jederzeit gemacht werden.
6. Zum Schluss wird der Anwender noch nach dem *Formular-Titel* gefragt. Dies ist die Bezeichnung, die in der Formularüberschrift steht (Standardeinstellung ist der Name, den die Tabelle bzw. Abfrage der Datenherkunft trägt); gleichzeitig speichert MS-Access das Formular unter diesem Namen. Deshalb sollte an dieser Stelle schon der für das Formular gewünschte Name eingegeben werden.

7. Abschließend werden noch die Formatierungen im Hauptformular vorgenommen. Dabei handelt es sich um Arbeiten wie: Positionieren der Text- und Bezeichnungsfelder sowie Festlegung der Farben.

#### 4.4.5.2 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 (siehe Bild 4-4) 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 (siehe Punkt 3.) oder ein Symbol beinhalten soll; hier wurde das entsprechende Symbol gewählt.
2. Im Bereich *Datensatznavigation* kommen sowohl Befehlsschaltflächen als auch Bezeichnungsfelder vor. Die Befehlsschaltflächen werden ebenso wie unter 1. erstellt (die Auswahl erfolgt diesmal aber aus der Kategorie *Datensatznavigation*, anschließend wird die jeweilige Aktion ausgewählt). Die Bezeichnungsfelder werden wiederum mit Hilfe der Toolbox erstellt und anschließend formatiert.
3. Ebenso ist die Schaltfläche mit dem Text „Eingabe eines neuen Lehrers“ eine Befehlsschaltfläche, sie bewirkt, dass beim Klicken ein anderes Formular (*Lehrer (Eingabe / Pflege)*) geöffnet wird, in dem Lehrerdaten aktualisiert oder neu eingegeben werden können.

Die Eingabe des Klassenleiters unter dem gleichnamigen Bezeichnungsfeld 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 Klassenleiters gezeigt werden.

4. Als erstes wählt man aus der Toolbox (vgl. Bild 4-4) 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 <i>Lehrer</i>
3	Auswahl der Felder der gewählten Tabelle	In diesem Fall alle Felder

	Frage(n):	Antwort:
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 Lehrernamen 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 wird in dem Feld <i>Klassenleiter-Nr</i> des Hauptformulars abgespeichert.
6	Nun wird nach der Bezeichnung (Erklärung) für das Kombinationsfeld gefragt.  Der Benutzer wird aber vom Assistenten <b>nicht</b> nach dem Namen für das Kombinationsfeld gefragt. Trotzdem sollte später über das Eigenschaftsfenster ein Name für das Kombinationsfeld vergeben werden.	Hier wird die <i>Bezeichnung Klassenleiter:</i> eingegeben.  In diesem Beispiel bietet sich als Name für das Kombinationsfeld die Bezeichnung <i>Lehrer-Auswahl</i> an.
7	Nun ist das Kombinationsfeld fertig.	Nachbesserungen können über das zugehörige Eigenschaftsfenster vorgenommen werden.

Eine Nachbesserung ist z. B. die alphabetische Sortierung der Lehrer 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 der zugrundegelegten Abfrage. Hier können die gewünschten Verbesserungen eingestellt und anschließend abgespeichert werden (z. B. alphabetische Reihenfolge).

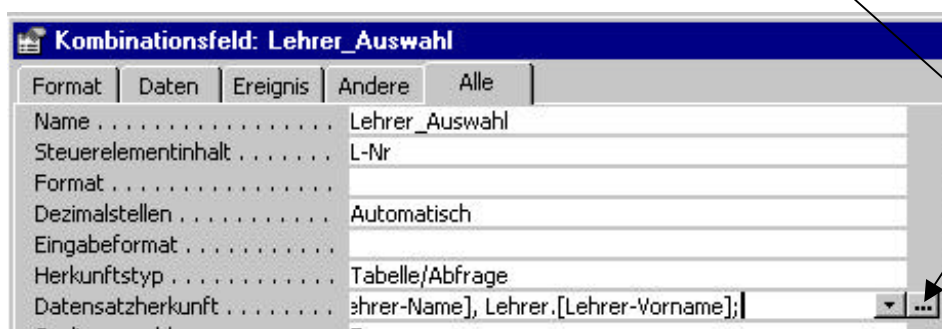


Bild 4-7: Eigenschaftsfenster des Kombinationsfeldes Lehrer-Auswahl

#### 4.4.5.3 Konstruktion und Einbau des Unterformulars

Prinzipiell kann das Unterformular auf zwei Arten erstellt werden:

1. Zusammen mit dem Hauptformular
2. Als unabhängiges Formular, das anschließend in das Hauptformular eingebaut wird.



An dieser Stelle soll der zweite Weg gewählt werden, da man im ersten Fall schon beim Erstellen des Hauptformulars (HF) auch alle Daten für das Unterformular (UF) kennen und mit den Daten des HF zusammen in einer Abfrage bereitstellen muss. So kann der Formularassistent beim Erstellen des HF die Frage stellen, welche Daten für das HF und welche für das UF bestimmt sind. Beim zweiten Verfahren kann das UF auch noch nachträglich in ein HF eingebaut werden, selbst wenn dieser Fall ursprünglich nicht vorgesehen war.

Zunächst überlegt man sich, aus welchen Tabellen die Daten für das UF stammen. Erkennbar ist in der Formularansicht (Bild 4-5) nur, dass es sich um Lehrerdaten (alle Felder aus der Tabelle *Lehrer*) handelt. An dieser Stelle spielt aber das zugrundegelegte logische Modell (Bild 3-24) eine entscheidende Rolle. Hier erkennt man, dass die eingetragenen Lehrerdaten ausschließlich in der Tabelle *ist in* gespeichert werden. D. h. die im UF angezeigten Daten sind nicht identisch mit den abgespeicherten Werten, da in der Tabelle *ist in* lediglich die *L-Nr* und die *Kl-Bez* der zugehörigen Lehrer bzw. Klassen stehen. Ein Hinweis, wie der Speichervorgang ablaufen könnte, ist der Pfeil für ein Kombinationsfeld neben dem Namen des Lehrers. Dies deutet an, dass durch die Auswahl des Lehrernamens die *L-ID* bestimmt wird, die dann in der *L-Nr* der Tabelle *ist in* gespeichert wird. Diese Technik soll nun in mehreren Schritten beschrieben werden:

1. Da die *L-Nr* im UF nicht zu sehen sein soll, dafür andere charakteristische Lehrerdaten, kommt als Datenherkunft für das UF nicht nur die Tabelle *ist in*, sondern auch die Tabelle *Lehrer* in Frage. Zunächst wird also eine Abfrage erstellt, die diese beiden Tabellen enthält. Dabei werden folgende Felder ausgewählt: *Kl-Bez*, *L-Nr*, *Lehrer-Vorname*, *Amtsbez*, *Fächerkombination*. Das Feld *Lehrer-Name* erscheint hier nicht, da es später im UF über das noch zu erstellende Kombinationsfeld zur Bestimmung der *L-Nr* angezeigt wird. Die Abfrage wird unter dem Namen *Unterrichtseinsatz für UF* abgespeichert.



Bild 4-9: Entwurfsansicht des Unterformulars UF Unterrichtseinsatz

2. Anschließend wechselt man in das Formularfenster, um das UF als ein neues Formular zu erstellen. Dabei werden alle Felder der Abfrage, die aus der Tabelle *Lehrer* stammen (*Lehrer-Vorname*, *Amtsbez*, *Fächerkombination*) in das UF übernommen. Die Felder *Kl-Bez*, *L-Nr* werden im UF nicht benötigt, müssen aber in der Abfrage vorhanden und angezeigt sein, da vom HF aus in diese Felder gespeichert bzw. das HF mit dem UF verknüpft wird. Als Layout

für das UF wird die Datenblattansicht gewählt (ist am übersichtlichsten und spart Platz im HF). Der Formularstil spielt in der Datenblattansicht keine Rolle, der Name für das UF ist *UF Unterrichtseinsatz*.

3. Nun müssen noch Nachbesserungen am UF vorgenommen werden. Insbesondere soll die Auswahl des Lehrers vom HF aus mit Hilfe eines Kombinationsfeldes erfolgen (damit soll auch über die Abfrage *Unterrichtseinsatz für UF* die *L-Nr* in der Tabelle *ist in* festgelegt werden). Dies erreicht man, indem im UF ein Kombinationsfeld erstellt wird (vgl. Kap. 4.4.5.2 Unterpunkt 4.). Das Kombinationsfeld enthält die Felder *L-ID*, *Lehrer-Name*, *Lehrer-Vorname*, *Amtsbez*; die *L-ID* als Schlüsselspalte wird nicht angezeigt. Der ausgewählte Wert (*L-ID*) wird in dem Feld *L-Nr* der zugrunde gelegten Abfrage gespeichert (nun erkennt man die Bedeutung dieses Feldes!). Der Name für das Kombinationsfeld ist *Lehrer-Name*.
4. Da das Kombinationsfeld als letztes Steuerelement im UF erstellt wurde, ist es auch in der Bearbeitungsreihenfolge am Ende (nach allen anderen Feldern). Um dies zu ändern, muss man in der Entwurfsansicht im Hauptmenü den Punkt *Ansicht* und dann die Option *Aktivierreihenfolge* auswählen. Hier zieht man das Kombinationsfeld an die gewünschte Position der Bearbeitungsreihenfolge (Platz 1) und bestätigt diese Änderung mit OK. Anschließend kann man noch über das Eigenschaftsfenster des Kombinationsfeldes Nachbesserungen vornehmen (siehe Kap. 4.4.5.2 Unterpunkt 4).
5. Abschließend muss das UF noch in das HF eingebunden werden. Dazu muss man sich auf jeden Fall in der Entwurfsansicht des HF's (in diesem Beispiel *Klassen und zugehörige Lehrer*) befinden. Nun hat man zwei Möglichkeiten: entweder zieht man nach Auswahl des zugehörigen Assistenten der Toolbox (Unterformular/bericht, siehe Bild 4-4) eine entsprechende Fläche für das UF im HF auf, oder man öffnet das HF nur in der Teilbildansicht, so dass das Datenbankfenster mit der Formularansicht im Hintergrund sichtbar ist. Nun zieht man das gewünschte UF (hier *UF Unterrichtseinsatz*) an die entsprechende Position im HF.
6. In der Regel muss das UF noch mit dem HF verknüpft werden, da sonst im UF nicht die zu dem jeweiligen Datensatz des HF gehörigen Daten, sondern alle möglichen Kombinationen angezeigt werden. Dies geschieht im Eigenschaftsfenster des UF. Hier müssen in den Zeilen *Verknüpfen von* bzw. *Verknüpfen nach* die richtigen Einstellungen vorgenommen werden. In der Zeile *Verknüpfen von* muss der Feldname des Verknüpfungsfeldes des UF stehen (hier: *Kl-Bez*). In der Zeile *Verknüpfen nach* muss der Feldname des Verknüpfungsfeldes aus dem HF stehen (hier: *Klassenbez*).



**Bild 4-10: Eigenschaftsfenster des Unterformulars UF Unterrichtseinsatz: Verknüpfung mit dem HF**

Man erkennt an dieser Stelle sehr deutlich, wie wichtig das Verständnis des logischen Datenbankmodells ist, da die Kenntnis der Verknüpfungsstruktur der verschiedenen Tabellen bei der Konstruktion von effizienten Formularen und Unterformularen unerlässlich ist.



## 4.5 Berichte

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.

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 drei verschiedene Berichtsassistenten zur Verfügung, die die am häufigsten benötigten Lösungen abdecken:

- Die einfachste Form ist **Einspaltig**. Diese Form ist vergleichbar mit dem einspaltigen Formular. Die Daten werden zeilenweise untereinander angeordnet. Die einzige Auswertungsmöglichkeit besteht darin, bestimmte Felder zu selektieren und eine Sortierfolge festzulegen.
- Der Bericht **Gruppierungen** fasst die Daten in Gruppen zusammen. Eine Gruppe entsteht, wenn mehrere Datensätze gleiche Feldinhalte haben. Diese Datensätze werden zusammengefasst und bilden eine Datengruppe. Als Auswertungsmöglichkeit kann man für jede Gruppe eine Summe bilden und einen Gesamtbetrag für alle Gruppen berechnen.
- Mit **Adressetiketten** 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 Berichtsassistenten 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. Ansonsten empfiehlt es sich, einfach einen Bericht zu erstellen und im Bedarfsfall im Handbuch nachzuschlagen, da die Menüführung in MS-ACCESS meistens sehr klar und übersichtlich ist und die interaktive Arbeit in diesem Fall effektiver ist als das vorherige Erarbeiten von theoretischen Grundlagen.

#### 4.5.1 Beispiel für die Erstellung eines Berichts

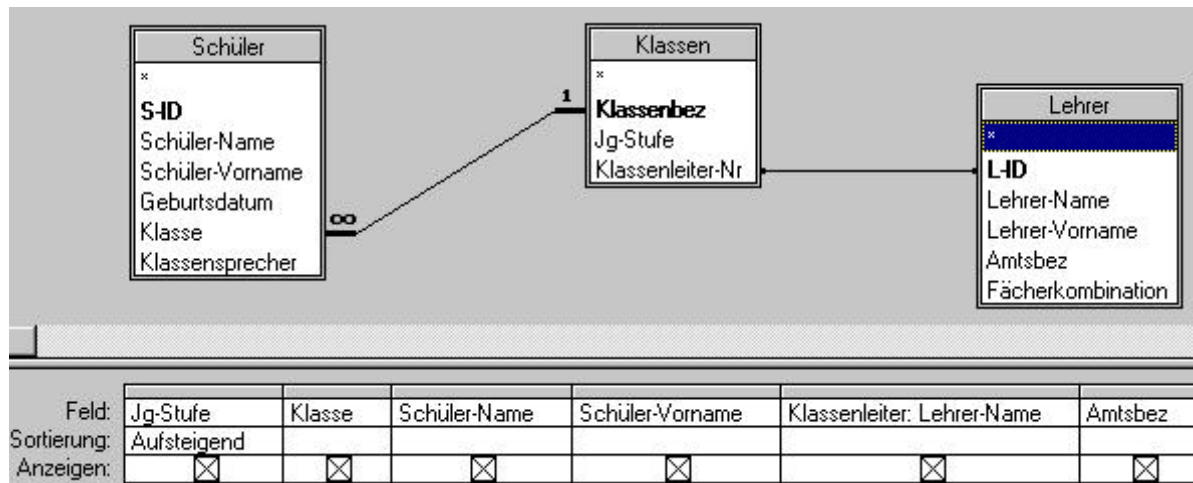
Als Beispiel für den Aufbau und die Konstruktion eines Berichts sollen an dieser Stelle die wesentlichen Schritte zur Erstellung des folgenden Berichts, der die Klassenliste jeder Klasse ausgibt, durchgeführt werden.

<b>Staatliches Gymnasium Irgendwo</b>	
<b>Klasse:</b> 5B	Klassenleiter: Tralala, OStR
Nr	Familienname, Vorname
1	Alt Walter
2	Anfang Sepp
3	Benz Vreni
4	Braun Walter
5	Daimler Mercedes
6	Diesel Otto

**Bild 4-11: Bericht Klassenliste (Ausschnitt der Seitenansicht)**

1. Zunächst muss, ebenso wie bei der Erstellung eines Formulars, entschieden werden, woher die Daten kommen, die in dem Bericht dargestellt werden sollen. Auch hier ist das Wissen über die Struktur der Datenbank notwendig. Wie man erkennt, werden in obigem Beispiel Daten verwendet, die aus den Tabellen *Schüler* (*Sr-Name*, *S-Vorname*) und *Lehrer* (*Nachname* und *Amtsbezeichnung des Klassenleiters*) stammen.
2. Weil die erforderlichen Daten nicht alle aus einer einzigen Tabelle kommen, muss also eine Abfrage als Datenherkunft für den Bericht verwendet werden. Da die genannten Tabellen in dem zugehörigen logischen Modell nicht direkt miteinander verknüpft sind, muss folglich eine

Tabelle in der Abfrage Verwendung finden, über die die Tabelle *Lehrer* mit der Tabelle *Schüler* verknüpft werden kann. Dies wird die Tabelle *Klassen* sein, da von den zugehörigen Klassen und Lehrern nur die Klassenleiter benötigt werden. Außerdem wird noch das Feld *Jg-Stufe* verwendet, da die Klassenlisten zuerst nach Jahrgangsstufe und dann nach Klassenbezeichnung und Schülernamen sortiert ausgegeben werden sollen. Diese Abfrage wird unter dem Namen *Alle Schüler und zugehörige Klassen (nach Klassen alphabetisch)* abgespeichert.



**Bild 4-12:** Entwurfsansicht der Abfrage *Alle Schüler und zugehörige Klassen (nach Klassen alphabetisch)*

- Nun wechselt man in das Berichtsfenster der Datenbank und wählt die Option *Neu*, um einen neuen Bericht zu erstellen. Dabei wird der Berichtsassistent verwendet, die ausgewählte Abfrage/Tabelle ist die gerade erstellte Abfrage *Alle Schüler und zugehörige Klassen (nach Klassen alphabetisch)*.
- Nun muss der geeignete Berichtsassistent ausgewählt werden. Da die Daten klassenweise ausgegeben werden (d. h. pro Klasse eine eigene Seite), innerhalb der Klassen aber alphabetisch nach Schülernamen sortiert sein sollen, muss der Assistent für Gruppierungen ausgewählt werden. Außerdem muss folgendes beachtet werden: das Feld *Klassenbez* ist vom Datentyp *Text*, so dass bei der Sortierung nach diesem Feldnamen die Klassen mit der Bezeichnung 11A vor den Klassen mit der Bezeichnung 5A, 5B, ...etc. ausgegeben würden. Folglich muss als erstes nach den Jahrgangsstufen und innerhalb der Jahrgangsstufen nach den Klassenbezeichnungen sortiert werden.
- Als nächstes wird festgelegt, welche Felder der zugrundegelegten Abfrage in dem Bericht verwendet werden sollen. Hier werden alle Felder ausgewählt.
- Nun wird über die Gruppierung bzw. Sortierung entschieden. Bei der Gruppierung wird an erster Stelle die *Jg-Stufe* und dann die *Klasse* ausgewählt, als Gruppierungskriterium wird jeweils die Voreinstellung *Normal* übernommen. Anschließend wird die Sortierung festgelegt: zuerst nach *Schüler-Name*, dann nach *Schüler-Vorname* (Begründung siehe 4.).
- Nun wählt man den geeigneten Berichts-Stil (*Präsentation* sowie Seite im Hochformat) aus.
- Zum Schluss werden noch Entscheidungen über den Berichtstitel (kann später problemlos geändert werden, ist aber ab Version 97 gleichzeitig der Name, unter dem der Bericht abgespeichert wird) getroffen.

speichert wird) sowie einige weitere Optionen getroffen. Hier wird lediglich der Punkt *Alle Felder auf einer Seite sehen* ausgewählt.

9. Der so entstandene Entwurf des Assistenten ist schon ganz brauchbar und muss nur in einigen Punkten nachgebessert werden:
  - Das Feld *Jg-Stufe* wird nicht angezeigt; ebenso werden der zugehörige Gruppenkopf bzw. Gruppenfuß nicht angezeigt bzw. deren Höhe jeweils auf 0 cm eingestellt.
  - Die Felder *Klassenleiter* und *Amtsbez* kommen in den Gruppenkopf der Klassenbezeichnung, *Schüler-Name* bzw. *Schüler-Vorname* in den Detailbereich der Klasse.
  - Damit nach jeder Klasse eine neue Seite beginnt, stellt man im Eigenschaftsfensters des Gruppenfußes der Klassenbez (Klasse-Fußbereich) das Feld *Neue Seite* von der Option *Keine* auf *Nach Bereichsende*.
  - Um für jeden Schüler innerhalb einer Klasse eine laufende Nummer zu erhalten, (siehe Bild 4-11), konstruiert man mit Hilfe der Toolbox ein Textfeld vor dem Feld *Schüler-Name*. In dieses Feld wird als Steuerelementinhalt „= 1“ eingetragen und im zugehörigen Eigenschaftsfenster im Feld *Laufende Summe* die Option *Über Gruppe* ausgewählt.
  - Damit der Nachname und der Vorname des Schülers in einem Feld, und nur durch Leerzeichen getrennt erscheinen, schreibt man z. B. in das Feld *Schüler-Name* die Formel:  $=[\text{Schüler-Name}] \& " " \& [\text{Schüler-Vorname}]$  (siehe Bild 4-13). Dabei muss aber beachtet werden, dass in diesem Bericht kein einziges Steuerelement einen in der Formel verwendeten Namen hat. Deshalb muss in diesem Fall der Namen des Feldes *Schüler-Name* geändert werden (z. B. *Schüler\_Name*). Das Feld *Schüler-Vorname* wird nicht mehr benötigt und infolgedessen ganz gelöscht. Analog verfährt man bei der Gestaltung des entsprechenden Feldes für den Klassenleiter und die zugehörige Amtsbezeichnung.

0 Berichtskopf											
0 Seitenkopf											
0	Staatliches Gymnasium Irgendwo										
1	Klasse:	Klasse	Klassenleiter:		=[Klassenleiter] & " " & [Amtsbez]						
2	Nr	Familiename, Vorname									
Klasse - Kopfbereich											
Detailbereich											
0	=1	=[Schüler-Name] & " " & [Schüler-Vorname]									
Klasse - Fußbereich											
Seitenfuß											
Berichtsfuß											

Bild 4-13: Bericht Klassenliste (Ausschnitt der Entwurfsansicht)

Auch in diesem Beispiel ist zu erkennen, dass bei der Erstellung von Berichten die Kenntnisse des logischen Modells der Datenbank zur Geltung kommen. Dies wird sich insbesondere dort verstärkt auswirken, wo, wie im Falle des Formulars mit Unterformular, Berichte mit Unterberichten konstruiert werden müssen.

### 4.5.2 Zusammenfassung

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.

## 4.6 Datenschutz und Datensicherheit

In diesem Abschnitt soll noch einmal darauf hingewiesen werden, dass bei der Bearbeitung personenbezogener Daten auf die Einhaltung der Richtlinien des Datenschutzes zu achten ist (siehe hierzu auch Vorwort und Anhang des Skriptums sowie Kapitel 3.2.3, Festlegung des Datenrahmens).

Außerdem sollen in diesem Zusammenhang kurz die Sicherheitssysteme von MS-ACCESS erwähnt werden. Auch wenn das Programm nur auf einem einzelnen PC eingesetzt wird, ist es möglich, verschiedene Sicherheitsstufen einzuhalten. Das Programm sieht hierzu u. a. standardmäßig drei verschiedene Benutzergruppen vor, denen verschiedene Rechte zustehen oder denen zusätzliche Rechte eingeräumt werden können. Dies sind die Gruppen:

- Administratoren
- Benutzer
- Gäste

Näheres zu diesen Gruppen und der zugehörigen Rechtestruktur entnehmen Sie bitte dem Handbuch bzw. der entsprechenden Fachliteratur.



## 5 Literaturverzeichnis

- (1) Prof. Dr. H. Gehring unter Mitarbeit von M. Lontke  
FernUniversität Hagen  
Datenbanksysteme, 1993  
Vorlesung für Wirtschaftsinformatiker
- (2) U. Matthey, U. Meiser  
ACCESS 1.1 für Einsteiger  
Data Becker, 1993  
ISBN 3-8158-1008-6
- (3) G. Matthiesen, M. Unterstein  
Relationale Datenbanken und SQL  
Addison-Wesley  
ISBN 3-8273-1167-5
- (4) A. Meier  
Relationale Datenbanken, Eine Einführung für die Praxis;  
Springer-Verlag, Berlin Heidelberg New York, 2. Auflage 1995  
ISBN 3-540-57486-7
- (5) Microsoft  
ACCESS 2.0 Benutzerhandbuch
- (6) A. Roßkamp  
ACCESS, Einführung in das vielseitige Datenbanksystem  
dtv Verlag, 1993  
ISBN 3-423-50137-5
- (7) G. Schlageter, W. Stucky  
Datenbanksysteme: Konzepte und Modelle  
B.G. Teubner Verlag, Stuttgart, 1983  
ISBN 3-519-12339-8
- (8) Prof. Dr. P. Stahlknecht  
Einführung in die Wirtschaftsinformatik, 6. Auflage  
Springer-Verlag, Berlin Heidelberg New York, 1993  
ISBN 3-540-56370-9
- (9) M. Vetter  
Aufbau betrieblicher Informationssysteme  
mittels objektorientierter konzeptioneller Datenmodellierung  
B. G. Teubner Stuttgart, 1991  
ISBN 3-519-12495-5





## 6 Anhang

### 6.1 Datenschutz

Auch in der Schule werden Datenbanken benutzt, die personenbezogene Daten beinhalten. Dies sollte für Lehrer und Schüler gleichermaßen zum Anlass genommen werden, sich mit der Problematik des Datenschutzes auseinander zusetzen. Insbesondere sollten schon beim Entwurf solcher Datenbanken die entsprechenden Vorschriften des Datenschutzgesetzes beachtet werden. Deshalb werden an dieser Stelle diejenigen Paragraphen aus dem bayerischen Datenschutzgesetz bzw. dem Bundesdatenschutzgesetz aufgeführt, die für den schulischen Bereich die wesentlichen Richtlinien darstellen: § 28, § 39, § 40, BayEUG Art 85. Sollten wie im Einführungsbeispiel die Daten von Lehrern und Schülern bearbeitet werden, müssen darüber hinaus die Richtlinien der Bekanntmachung des bayerischen Staatsministeriums für Unterricht, Kultus, Wissenschaft und Kunst vom 19. März 1996 Nr. III/8-III/4-L0572-1/41785 (KWMBI. I Nr. 9/1996) eingehalten werden. Diese Bekanntmachung enthält erläuternde Hinweise für die Schulen zum Vollzug des Bayerischen Datenschutzgesetzes.

#### § 28

##### *Datenspeicherung, -übermittlung und -nutzung für eigene Zwecke*

*(1) Das Speichern, Verändern oder Übermitteln personenbezogener Daten oder ihre Nutzung als Mittel für die Erfüllung eigener Geschäftszwecke ist zulässig*

- 1. im Rahmen der Zweckbestimmung eines Vertragsverhältnisses oder vertragsähnlichen Vertrauensverhältnisses mit dem Betroffenen,*
- 2. soweit es zur Wahrung berechtigter Interessen der speichernden Stelle erforderlich ist und kein Grund zu der Annahme besteht, daß das schutzwürdige Interesse des Betroffenen an dem Ausschluß der Verarbeitung oder Nutzung überwiegt,*
- 3. wenn die Daten aus allgemein zugänglichen Quellen entnommen werden können oder die speichernde Stelle sie veröffentlichen dürfte, es sei denn, daß das schutzwürdige Interesse des Betroffenen an dem Ausschluß der Verarbeitung oder Nutzung offensichtlich überwiegt,*
- 4. wenn es im Interesse der speichernden Stelle zur Durchführung wissenschaftlicher Forschung erforderlich ist, das wissenschaftliche Interesse an der Durchführung des Forschungsvorhabens das Interesse des Betroffenen an dem Ausschluß der Zweckänderung erheblich überwiegt und der Zweck der Forschung auf andere Weise nicht oder nur mit unverhältnismäßigem Aufwand erreicht werden kann.*

*Die Daten müssen nach Treu und Glauben und auf rechtmäßige Weise erhoben werden.*

*(2) Die Übermittlung oder Nutzung ist auch zulässig*

*1. a) soweit es zur Wahrung berechtigter Interessen eines Dritten oder öffentlicher Interessen erforderlich ist oder*

*b) wenn es sich um listenmäßig oder sonst zusammengefaßte Daten über Angehörige einer Personengruppe handelt, die sich auf*

- eine Angabe über die Zugehörigkeit des Betroffenen zu dieser Personengruppe*
- Berufs-, Branchen- oder Geschäftsbeziehung*
- Namen,*
- Titel*
- akademische Grade,*
- Anschrift,*
- Geburtsjahr*

*beschränken und*

*kein Grund zu der Annahme besteht, daß der Betroffene ein schutzwürdiges Interesse an dem Ausschluß der Übermittlung hat. In den Fällen des Buchstabens b kann im allgemeinen davon ausgegangen werden, daß dieses Interesse besteht, wenn im Rahmen der Zweckbestimmung eines Vertragsverhältnisses oder vertragsähnlichen Vertrauensverhältnisses gespeicherte Daten übermittelt werden sollen, die sich*

- auf gesundheitliche Verhältnisse,*
- auf strafbare Handlungen,*
- auf Ordnungswidrigkeiten,*
- auf religiöse oder politische Anschauungen sowie*
- bei Übermittlung durch den Arbeitgeber auf arbeitsrechtliche Rechtsverhältnisse*

*beziehen, oder*

*2. wenn es im Interesse einer Forschungseinrichtung zur Durchführung wissenschaftlicher Forschung erforderlich ist, das wissenschaftliche Interesse an der Durchführung des Forschungsvorhabens das Interesse des Betroffenen an dem Ausschluß der Zweckänderung erheblich überwiegt und der Zweck der Forschung auf andere Weise nicht oder nur mit unverhältnismäßigem Aufwand erreicht werden kann.*

*(3) Widerspricht der Betroffene bei der speichernden Stelle der Nutzung oder Übermittlung seiner Daten für Zwecke der Werbung oder der Markt- oder Meinungsforschung, ist eine Nutzung oder Übermittlung für diese Zwecke unzulässig. Widerspricht der Betroffene beim Empfänger der nach Absatz 2 übermittelten Daten der Verarbeitung oder Nutzung für Zwecke der Werbung oder der Markt- oder Meinungsforschung, hat dieser die Daten für diese Zwecke zu sperren.*

*(4) Der Empfänger darf die übermittelten Daten für den Zweck verarbeiten oder nutzen, zu dessen Erfüllung sie ihm übermittelt werden. Eine Verarbeitung oder Nutzung für andere Zwecke ist nur unter den Voraussetzungen der Absätze 1 und 2 zulässig. Die übermittelnde Stelle hat den Empfänger darauf hinzuweisen.*

### § 39

#### ***Zweckbindung bei personenbezogenen Daten, die einem Berufs- oder besonderen Amtsgeheimnis unterliegen***

- (1) *Personenbezogene Daten, die einem Berufs- oder besonderen Amtsgeheimnis unterliegen und die von der zur Verschwiegenheit verpflichteten Stelle in Ausübung ihrer Berufs- oder Amtspflicht zur Verfügung gestellt worden sind, dürfen von der speichernden Stelle nur für den Zweck verarbeitet oder genutzt werden, für den sie sie erhalten hat. In die Übermittlung an eine nicht-öffentliche Stelle muß die zur Verschwiegenheit verpflichtete Stelle einwilligen.*
- (2) *Für einen anderen Zweck dürfen die Daten nur verarbeitet oder genutzt werden, wenn die Änderung des Zwecks durch besonderes Gesetz zugelassen ist.*

### § 40

#### ***Verarbeitung und Nutzung personenbezogener Daten durch Forschungseinrichtungen***

- (1) *Für Zwecke der wissenschaftlichen Forschung erhobene oder gespeicherte personenbezogene Daten dürfen nur für Zwecke der wissenschaftlichen Forschung verarbeitet oder genutzt werden.*
- (2) *Die Übermittlung personenbezogener Daten an andere als öffentliche Stellen für Zwecke der wissenschaftlichen Forschung ist nur zulässig, wenn diese sich verpflichten, die übermittelten Daten nicht für andere Zwecke zu verarbeiten oder zu nutzen und die Vorschrift des Absatzes 3 einzuhalten.*
- (3) *Die personenbezogenen Daten sind zu anonymisieren, sobald dies nach dem Forschungszweck möglich ist. Bis dahin sind die Merkmale gesondert zu speichern, mit denen Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren Person zugeordnet werden können. Sie dürfen mit den Einzelangaben nur zusammengeführt werden, soweit der Forschungszweck dies erfordert.*
- (4) *Die wissenschaftliche Forschung betreibenden Stellen dürfen personenbezogene Daten nur veröffentlichen, wenn*
  1. *der Betroffene eingewilligt hat oder*
  2. *dies für die Darstellung von Forschungsergebnissen über Ereignisse der Zeitgeschichte unerläßlich ist.*

***Bayer. Gesetz über das Erziehungs- und  
Unterrichtswesen (BayEUG)***  
***in der Fassung der Bekanntmachung vom 7. Juli 1994***

**§ 85**

***Erhebung und Verarbeitung von Daten***

- (1) <sup>1</sup> Zur Erfüllung der den Schulen durch Rechtsvorschriften jeweils zugewiesenen Aufgaben sind die Erhebung und die Verarbeitung von Daten zulässig. <sup>2</sup> Dazu gehören personenbezogene Daten der Schüler und der Erziehungsberechtigten, insbesondere Adreßdaten, schulische Daten, Leistungsdaten sowie Daten zur Vorbildung und Berufsausbildung <sup>3</sup> Der Betroffene ist zur Angabe der Daten verpflichtet; er ist bei der Datenerhebung auf diese Rechtsvorschrift hinzuweisen.
- (2) Die Weitergabe von Daten und Unterlagen über Schüler und Erziehungsberechtigte an außerschulische Stellen ist im übrigen untersagt, falls nicht ein rechtlicher Anspruch auf die Herausgabe der Daten nachgewiesen wird.
- (3) Gibt eine Schule für die Schüler und Erziehungsberechtigten einen Jahresbericht heraus, so dürfen darin folgende personenbezogene Daten enthalten sein:  
Name, Geburtsdatum, Jahrgangsstufe und Klasse der Schüler, Name, Fächerverbindung und Verwendung der einzelnen Lehrkräfte, Angaben über besondere schulische Tätigkeiten und Funktionen einzelner Lehrkräfte, Schüler und Erziehungsberechtigter.

## 6.2 Index

### 1

1:1-Beziehung .....	58
1:n-Beziehung .....	57
1. Normalform (1NF).....	39, 45

### 2

2. Normalform (2NF).....	41, 45, 47
--------------------------	------------

### 3

3. Normalform (3NF).....	42, 45, 49
--------------------------	------------

### 4

4. Normalform (4NF).....	45
--------------------------	----

### 5

5. Normalform (5NF).....	45
--------------------------	----

## A

Abbildungsregeln.....	30, 31
Abfragesprache.....	16
Administratoren.....	83
Adresstiketten .....	79
Aktionsabfrage .....	67
Aktualisierungsabfrage .....	68
ändern .....	59
Änderungsanomalien .....	44
Anfügeabfrage .....	68
Anomalien.....	44
ANSI.....	17
ANSI-Architekturmodell .....	17
Anwendungsadministrator .....	18
anwendungsorientiertes Datenmodell.....	22
Assoziation.....	25
atomar .....	45
Attribut.....	22, 24, 26, 43
Attribut / Eigenschaft.....	43
Attribut / Merkmal.....	43
Attributs-Inhalt.....	27
Attributs-Name.....	27
Attributwerte .....	30
Ausprägung .....	27
Auswahlabfragen .....	67

Auto Join.....	58
----------------	----

## B

BCNF (Boyce-Codd Normalform) .....	45
Benutzer.....	83
Benutzerfreundlichkeit .....	11
Benutzersichten .....	18
berechenbare Daten .....	42
Berichte.....	23, 79
Berichtsfuß.....	79
Berichtskopf.....	79
Beziehung .....	22, 24, 25, 61
Beziehungen zwischen Entitäten .....	25
Beziehungsmengen-Relation .....	30, 31
Bottom-up-Ansatz .....	22
Boyce-Codd Normalform (BCNF) .....	45
Bottom-up-Methode .....	23

## D

Data Control Language (DCL) .....	54
Data Definition Language (DDL) .....	53
Data Manipulation Language (DML) .....	53
Data Query Language (DQL).....	53
Dateisysteme .....	12
Daten .....	21
Datenabfrage .....	17
Datenabfragesprache .....	53
Datenbank.....	14, 15, 43
Datenbankadministrator .....	18
Datenbankbasis .....	14
Datenbankmanagementsystem (DBMS)15, 16, 17,	20
Datenbanksystem (DBS).....	14, 15, 17, 18
Datenbankverwaltungssystem.....	17, 20
Datenbasis.....	15, 17
Datendefinition .....	17
Datendefinitionssprache .....	53
Datenfeld.....	16, 54
Datenintegrität .....	12, 49
Datenkonsistenz .....	12
Datenkontrolle.....	17
Datenkontrollsprache .....	54
Datenmanipulation .....	17
Datenmanipulationssprache.....	53

Datenmodell .....	19, 21, 23
Datenmodellierung .....	21
Datenorganisation.....	11
Datenpersistenz .....	12
Datenpflege .....	59
Daten-Programm-Abhängigkeit .....	13
Datensatz.....	16, 43, 54
Datenschutz .....	12, 83
Datensicherheit.....	12, 83
Datenübertragung .....	17
Datenübertragungssprache.....	54
Datenunabhängigkeit.....	11
DBMS (Datenbankmanagementsystem).....	15, 16, 17, 20
DBS (Datenbanksystem) .....	18
DCL (Data Control Language).....	54
DDL (Data Definition Language).....	53
delete .....	59
Detailbereich .....	71, 79
Diagramme.....	72
DML (Data Manipulation Language).....	53
DQL (Data Query Language).....	53
duplicate index .....	57

## E

Effizienz .....	11
eindeutige Identifikation .....	40
Eindeutiger Schlüssel.....	56
Eindeutigkeitsbedingung.....	49
einfach.....	25
einfach-bedingt.....	25
einfach-einfache Beziehungen .....	26
einfach-komplexe Beziehungen .....	26
Einfügeanomalie.....	44
einfügen.....	59
Einspaltig .....	79
elementare Attribute .....	39
Entität .....	24, 30
Entität / Beziehung.....	43
Entitäts- / Beziehungsmenge.....	43
Entitätsklassen.....	23
Entitätsmenge.....	24, 26, 27, 30
Entity-Relationship-Diagramm .....	28
Entity-Relationship-Modell.....	23
Entwurfssprache .....	63, 64, 65, 66, 68
Equi Join .....	58

E-R-Diagramm.....	28, 43
E-R-Modell .....	24, 43
externe Ebene.....	18
externe Phase .....	21, 43
externes Schema.....	19

## F

Feld .....	43, 54
Flexibilität.....	11
Formulare.....	23, 69
Formularfuß .....	71
Formularkopf .....	71
Fremdschlüssel.....	30
funktional abhängig.....	47

## G

Gäste .....	83
Gleichheitsverknüpfung.....	58
globales Datenmodell .....	21
Gruppenfuß .....	80
Gruppenkopf .....	80
Gruppierungen .....	79

## H

Hauptindex.....	56
hierarchisches Modell.....	29

## I

Identifikationsschlüssel .....	30
Index .....	56
Index mit Duplikaten.....	57
Index ohne Duplikate .....	56
Indexdatei .....	56
Indizierung.....	56
Inflexibilität .....	13
Informationsstruktur.....	21, 43
inkonsistent.....	37
Inkonsistenz .....	13
insert .....	59
Integrität .....	49
Integrität der Entität.....	37
interne Ebene .....	18
internes Schema .....	19

**J**

Join..... 59

**K**

Kardinalität ..... 25

Kardinalität einer Assoziation..... 25

Kardinalitätstyp ..... 25

Komplex-komplexe Beziehungen..... 26

Konsistenz ..... 49

konzeptionelle Ebene ..... 18

konzeptionelle Phase..... 21, 43

konzeptionelles Schema..... 18, 19

**L**

Left-Join (Inklusionsverknüpfung) ..... 58

logisch widersprüchlich ..... 37

logische Dateneinheiten ..... 19

logische Phase ..... 21, 43

logische Sicht ..... 18

logisches Modell..... 21, 35

Löschabfrage ..... 68

Löschanomalie..... 44

löschen ..... 59

**M**

Mehrbenutzerbetrieb ..... 11

mehrfach ..... 25

mehrfach-bedingt..... 25

Mehrfachzugriff..... 11

Mehrfelder-Primärschlüssel ..... 37, 38

Merkmale ..... 22, 24

Merkmalsausprägungen ..... 24

Migration ..... 23

Modell..... 21

**N**

Netzwerkmodell..... 29

Nichtschlüsselattribute..... 47

Normalformen ..... 37

1. Normalform (1NF) ..... 39, 45

2. Normalform (2NF) ..... 41, 45, 47

3. Normalform (3NF) ..... 42, 45, 49

4. Normalform (4NF) ..... 45

5. Normalform (5NF) ..... 45

normalisierte Relationen ..... 23

not duplicate index ..... 56

**O**

Objekt / Beziehung ..... 43

Objekt-/Beziehungsmenge ..... 43

Objekte..... 22, 43

objektorientiertes Modell ..... 29

Objektstruktur ..... 43

Operationen..... 43

Outer Join ..... 58

**P**

physikalische Organisation..... 19

physikalische Phase..... 43

physische Phase..... 21

physische Sicht..... 18

physischer Speicher..... 18

Primärschlüssel ..... 30, 37, 56

primary key ..... 56

Projektion..... 59

**Q**

Query By Example (QBE) ..... 62

**R**

RDBS ..... 43

redundant ..... 44

Redundanz ..... 13, 44

Redundanzfreiheit ..... 12

referentielle Integrität ..... 12

referentielle Integritätsbedingung ..... 49

Regeln..... 43

Relation..... 43, 54

relationales Modell ..... 36, 43

Relationenmodell ..... 29, 43

Relationenstruktur ..... 43

Restriktion..... 59

restriktive Löschung ..... 51

Right-Join (Inklusionsverknüpfung) ..... 58

**S**

Schema ..... 21

Schlüssel ..... 56

Seitenfuß ..... 71, 79

Seitenkopf ..... 71, 79

Sekundärschlüssel.....	30, 57
Selektion.....	59
semantisches Datenmodell.....	21
semantisches Modell.....	35
Spalte der Tabelle .....	16
Spalten .....	54
SQL (Structured Query Language).....	60
Steuerelement .....	71
Structured Query Language (SQL).....	60
Strukturelle Integritätsbedingungen .....	49
Subschema.....	19

## T

Tabelle.....	54
- Spalte der Tabelle .....	16
- Zeile der Tabelle .....	16
Tabelle / Datendatei.....	43
Tabellenerstellungsabfrage.....	68
Toolbox .....	72
Top-down-Ansatz .....	21
Top-down-Methode .....	23
Transaction Processing Language (TPL).....	54
transitiv abhängig .....	49

Tupel.....	43, 54
------------	--------

## U

unique index.....	56
Unterformular .....	72
Unternehmensadministrator .....	18
update .....	59

## V

Verknüpfung von Tabellen .....	57
Vielfachzugriff .....	16
voll funktional abhängig.....	47

## W

Wert.....	27
Wertebereichsbedingung .....	49

## Z

Zeile der Tabelle.....	16
Zeilen.....	54
Zugriffsrechte.....	17