

# Datenbanken und Datenbanknutzer

Datenbanken und Datenbanksysteme spielen eine wesentliche Rolle in vielen Bereichen der modernen Gesellschaft. Die meisten von uns kommen täglich mit verschiedenen Aktivitäten in Berührung, die eine oder andere Interaktion mit einem Datenbankmanagementsystem umfassen, wenn wir beispielsweise auf der Bank einen Scheck einreichen, in einem Reisebüro eine Reise buchen, in der Bibliothek ein Buch in einem computergestützten Bibliothekskatalog suchen oder bei einem Medienverlag eine Zeitschrift abonnieren. Sogar der Einkauf von Artikeln in einem Supermarkt umfasst heute in den meisten Fällen eine automatische Aktualisierung der Datenbank, die den Bestand der Artikel verwaltet.

Die obigen Interaktionen sind Beispiele dessen, was unter **traditionellen Datenbankanwendungen** verstanden wird, bei denen die meisten Informationen entweder in Textform oder numerisch gespeichert werden, um einen Zugriff darauf zu ermöglichen. In den letzten Jahren haben technologische Fortschritte zu interessanten neuen Anwendungen für Datenbanksysteme geführt. **Multimedia-Datenbanken** können heute Bilder, Videoclips und Sound-Nachrichten speichern. In **geografischen Informationssystemen (GIS)** werden in Datenbanken Landkarten, Wetterdaten und Satellitenbilder gespeichert und analysiert. Für **Data-Warehouse- und OLAP-Anwendungen (Online Analytical Processing)** werden in vielen Firmen Datenbankmanagementsysteme eingesetzt, um Informationen aus sehr großen Datenmengen für Entscheidungsprozesse zu extrahieren und zu analysieren. **Echtzeit- und aktive Datenbanksysteme** werden zur Steuerung von Industrie- und Fertigungsprozessen eingesetzt. Datenbanksuchtechniken dienen im World Wide Web dazu, den Zugriff auf Daten für die im Internet surfenden Benutzer zu unterstützen.

Um die Grundlagen heutiger Datenbankmanagementsysteme zu verstehen, müssen wir mit den Grundlagen traditioneller Datenbankanwendungen beginnen. In Abschnitt 1.1 dieses Kapitels definieren wir also zuerst, was eine Datenbank ist, und erklären anschließend weitere wichtige Fachbegriffe. In Abschnitt 1.2 verwenden wir mit der Datenbank UNIVERSITY ein einfaches Beispiel, um unsere Diskussion zu illustrieren. In Abschnitt 1.3 werden einige wichtige Merkmale von Datenbanksystemen diskutiert und beschrieben. In den Abschnitten 1.4 und 1.5 werden die Benutzer kategorisiert, deren Aufgabe die Benutzung von und die Interaktion mit Datenbanksystemen beinhaltet. Die Abschnitte 1.6, 1.7 und 1.8 enthalten eine gründlichere Beschreibung der verschiedenen Möglichkeiten, die Datenbankmanagementsysteme bieten,

und die Implikationen der Anwendung des Datenbankansatzes. Abschnitt 1.9 fasst die wesentlichen Aspekte dieses Kapitels zusammen.

Leser, die lediglich eine schnelle Einführung in Datenbanksysteme wünschen, können die Abschnitte 1.1 bis 1.5 durcharbeiten, die Abschnitte 1.6 bis 1.8 überspringen und mit Kapitel 2 fortfahren.

## 1.1 Einführung

Datenbanken und Datenbanktechnologien haben große Auswirkung auf den Einsatz von Computern. Man kann getrost sagen, dass Datenbanken in fast allen Bereichen, in denen Computer zum Einsatz kommen, eine entscheidende Rolle spielen, z.B. in kommerziellen, technischen und Fertigungsumgebungen sowie in den Bereichen Medizin, Recht, Bildung und Bibliothekswissenschaft, um nur einige zu nennen. Das Wort *Datenbank* wird derart häufig gebraucht, dass wir mit der Definition einer Datenbank beginnen müssen. Unsere anfängliche Definition ist recht allgemein.

Eine **Datenbank** ist eine Sammlung von **Daten**, die einen Ausschnitt der realen Welt beschreiben. Unter Daten verstehen wir bekannte Tatsachen, die aufgezeichnet werden können und eine implizite Bedeutung haben. Beispielsweise werden der Name, die Telefonnummern und die Adresse zur Beschreibung einer Person benutzt. Diese Daten sind zum Beispiel in einem (indexierten) Adressbuch eingetragen; alternativ können sie auf Diskette gespeichert werden und auf einem PC mit Software wie Microsoft Access oder Excel bearbeitet werden.

Die obige Definition einer *Datenbank* ist recht allgemein; man betrachte z.B. die Sammlung von Wörtern, aus denen sich diese Textseite zusammensetzt, als zusammenhängende Daten und somit als Datenbank. Der übliche Gebrauch des Begriffs *Datenbank* wird in vielen Fällen aber eingeschränkt. Danach hat eine Datenbank folgende Eigenschaften:

- Eine Datenbank stellt Aspekte der realen Welt dar, die auch als **Miniwelt** oder **Universe of Discourse (UoD)** bezeichnet wird. Änderungen in der Miniwelt spiegeln sich in der Datenbank wider.
- Eine Datenbank ist eine logisch zusammenhängende Sammlung von Daten mit einer bestimmten inhärenten Bedeutung. Eine zufällige Datensammlung wird nicht als Datenbank bezeichnet.
- Eine Datenbank wird für einen bestimmten Zweck entworfen, entwickelt und mit Daten gefüllt. Sie wird von einer bestimmten Benutzergruppe in zweckbezogenen Anwendungen verwendet, an denen diese Benutzer interessiert sind.

Anders ausgedrückt, beschreibt eine Datenbank einen wohl definierten Ausschnitt der realen Welt, an dem ein bestimmter Personenkreis interessiert ist. Diese Personen wollen auf diese Datenbank zugreifen und deren Inhalt möglicherweise verändern.

Eine Datenbank kann jede beliebige Größe und Komplexität aufweisen. Die oben erwähnten Namen und Adressen können z.B. aus nur ein paar Hundert Datensätzen mit jeweils einer einfachen Struktur bestehen. Demgegenüber kann der Kartenkatalog einer großen Bibliothek eine halbe Million Karten enthalten, die unter verschiedenen Kategorien gespeichert werden, z.B. nach Autor, Thema oder Titel, wobei jede Kategorie alphabetisch geordnet ist. Eine noch größere und komplexere Datenbank wird vom Internal Revenue Service (IRS, Finanzamt in den USA) geführt, um die Steuererklärungen der Steuerzahler zu verwalten. Wenn wir davon ausgehen, dass es in den

USA 100 Millionen Steuerzahler gibt, die jeweils eine Steuererklärung einreichen, die sich aus fünf Formularen mit je ca. 200 Informationszeichen zusammensetzt, erhalten wir eine Datenbank mit  $100*(10^6)*200*5$  Zeichen (Byte) an Daten. Wenn das IRS zusätzlich zur aktuellen die jeweils letzten drei Steuererklärungen jedes Steuerzahlers vorhält, erhalten wir eine Datenbank von  $4*(10^{11})$  Byte (400 Gbyte). Diese riesige Datenmenge muss organisiert und verwaltet werden, so dass die Benutzer Daten nach Bedarf suchen, abrufen und aktualisieren können.

Eine Datenbank kann manuell oder computergestützt erzeugt und gepflegt werden. Der Kartenkatalog einer Bibliothek ist ein Beispiel einer Datenbank, die manuell erstellt und gepflegt werden muss. Eine computergestützte Datenbank kann entweder mit einer Reihe von Anwendungen, die speziell für diese Aufgabe geschrieben wurden, oder mit Hilfe eines Datenbankmanagementsystems erstellt und gepflegt werden.

Ein **Datenbankmanagementsystem (DBMS)** ist ein Softwaresystem (i.e. Sammlung von Programmen), das dem Benutzer das Erstellen und die Pflege einer Datenbank ermöglicht. Das DBMS ist folglich ein *generelles Softwaresystem*, das die Prozesse der *Definition*, *Konstruktion* und *Manipulation* von Datenbanken für verschiedene Anwendungen vereinfacht. Die **Definition** einer Datenbank bedeutet die Spezifikation der Datentypen, Strukturen und Einschränkungen für die in der Datenbank zu speichernden Daten. Die **Konstruktion** der Datenbank ist der Prozess des Speicherns der Daten auf einem Speichermedium, das vom DBMS kontrolliert wird. Die **Manipulation** einer Datenbank beinhaltet Funktionen wie Anfragen der Datenbank, um spezifische Daten abzurufen, Fortschreiben der Datenbank Änderungen in der Miniwelt und Erzeugen von Berichten aus den Daten.

Es ist nicht unbedingt notwendig, eine generelle DBMS-Software zu verwenden, um eine computergestützte Datenbank zu implementieren. Wir könnten unsere eigenen Programme schreiben, um die Datenbank zu erstellen und zu verwalten, womit wir effektiv unsere eigene *spezielle* DBMS-Software erstellen würden. In beiden Fällen, ob wir ein generelles DBMS benutzen oder nicht, müssen wir normalerweise einen beträchtlichen Umfang an Software verwenden, um die Datenbank zu manipulieren. Datenbank und DBMS-Software bilden zusammen ein **Datenbanksystem**; das Konzept ist in Abbildung 1.1 dargestellt.

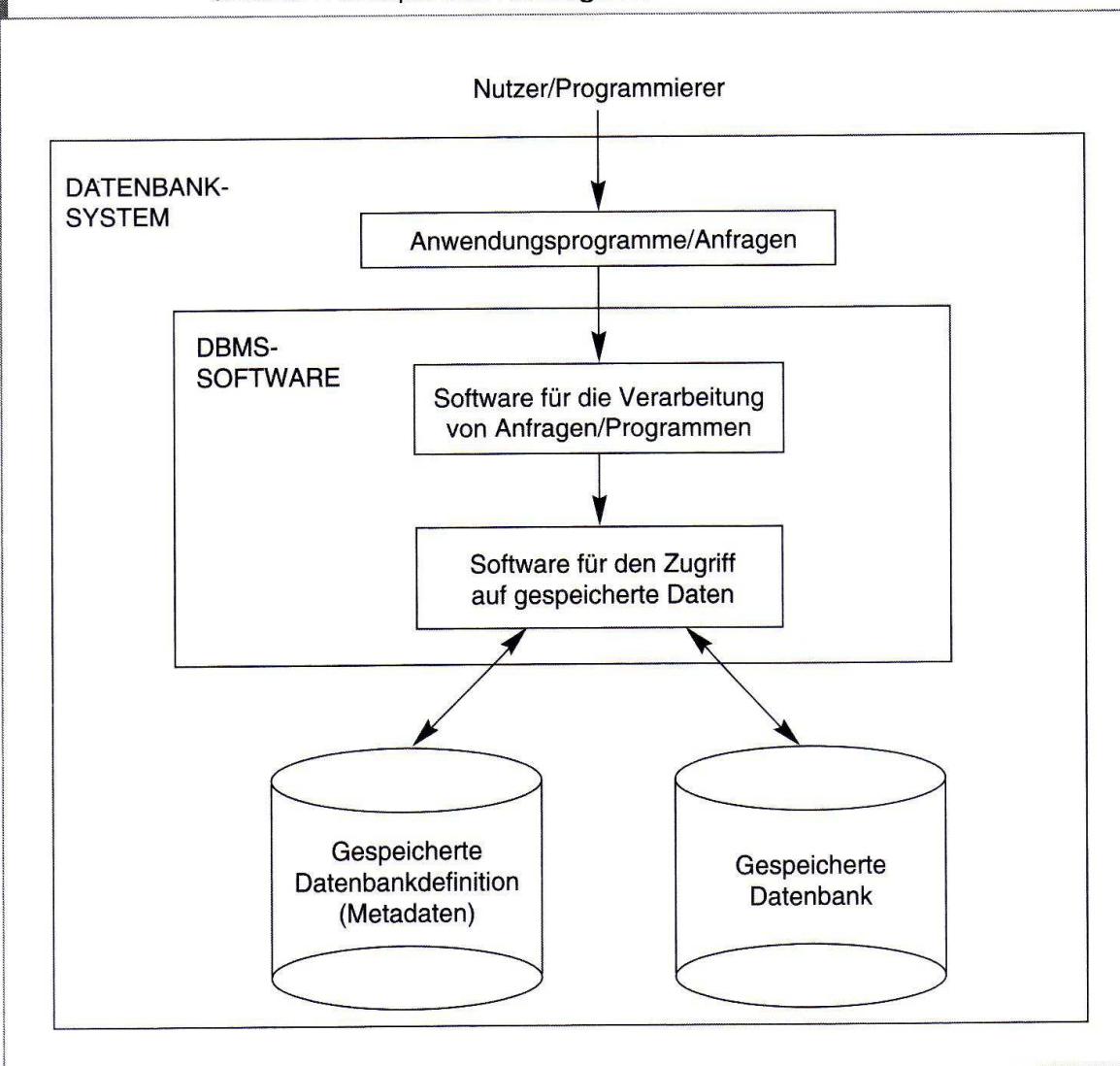
## 1.2 Ein Beispiel

Wir betrachten in diesem Abschnitt ein Beispiel, das den meisten Lesern wahrscheinlich vertraut ist: eine Datenbank UNIVERSITY für die Verwaltung von Daten über Studenten und Vorlesungen in einer Universitätsumgebung. Abbildung 1.2 zeigt die Datenbankstruktur und einige Musterdaten für eine solche Datenbank. Die Datenbank ist in fünf Tabellen organisiert, in denen jeweils Datensätze der gleichen Struktur gespeichert werden.<sup>1</sup> In der Tabelle STUDENT werden Daten über jeden Studenten gespeichert, während die Tabelle COURSE Daten zu allen Kursen aufnimmt. In der Tabelle SECTION werden Daten über jeden Kursabschnitt gespeichert. In der Tabelle GRADE\_REPORT werden die Noten gespeichert, die Studenten in den verschiedenen absolvierten Abschnitten erhalten haben. Und die Tabelle PREREQUISITE enthält die Kursvoraussetzungen.

---

1. Auf konzeptueller Ebene ist eine *Tabelle* eine Sammlung von Datensätzen, die geordnet sein können, aber nicht müssen.

**Abbildung 1.1:** Vereinfachte Datenbanksystemumgebung zur Darstellung der in Abschnitt 1.1 erklärten Konzepte und Fachbegriffe.



Um diese Datenbank zu *definieren*, müssen wir die Struktur der Datensätze in jeder Tabelle definieren. Wir spezifizieren also die verschiedenen Typen der **Datenelemente**, die in jedem Datensatz gespeichert werden sollen. In Abbildung 1.2 enthält jeder Datensatz STUDENT Daten für die Darstellung von Name, StudentNumber, Class (Freshman oder 1, Sophomore oder 2, ...) und Major (MATH, Computer Science bzw. CS, ...) der einzelnen Studenten; jeder Datensatz COURSE beinhaltet Daten für die Darstellung von CourseName, CourseNumber, CreditHours und Department (wo die jeweiligen Kurse stattfinden) und so weiter. Wir müssen auch einen **Datentyp** für jedes Datenelement innerhalb eines Datensatzes spezifizieren. Beispielsweise können wir festlegen, dass der Name eines Studenten eine Kette aus Zeichen eines Alphabets, StudentNumber von STUDENT ein Integer und Grade von GRADE\_REPORT ein einzelnes Zeichen aus der Menge {A, B, C, D, F, I} ist. Wir können außerdem ein Kodierschema anwenden, um ein Datenelement darzustellen. In Abbildung 1.2 wird die Class eines Studenten z.B. als 1 für Freshman, 2 für Sophomore, 3 für Junior, 4 für Senior und 5 für Graduate dargestellt.

Um die Datenbank UNIVERSITY zu *konstruieren*, speichern wir Datensätze in den jeweiligen Tabellen Student, Course, Section, Grade Report und Prerequisite. Man

**Abbildung 1.2:** Beispiel einer Datenbank, in der Studentendatensätze mit Noten gespeichert werden.

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

beachte, dass Datensätze in dcn verschiedenen Tabellen logisch zusammenhängen können. Beispielsweise hängt der Datensatz für »Smith« in der Tabelle STUDENT mit zwei Datensätzen der Tabelle GRADE\_REPORT zusammen, die Smiths Zensuren für zwei Übungen festlegen. Ähnlich hängt jeder Datensatz in der Datei PREREQUISITE mit zwei Veranstaltungsdatensätzen zusammen: Einer stellt die Veranstaltung und der andere die Voraussetzungen für die Teilnahme dar. Die meisten mittleren und großen Datenbanken umfassen eine große Anzahl von Tabellen mit *vielen Beziehungen* zwischen Datensätzen.

Die *Manipulation* der Datenbank umfasst Anfragen und Aktualisierungen. Beispiele von Anfragen sind »Erstelle eine Liste aller Kurse und Zensuren für Smith«;

»Liste die Namen von Studenten, die im Herbst 1999 den Datenbankkurs belegt haben, mit Zensuren und den betreffenden Übungen auf«; und »Welche Voraussetzungen bestehen für die Datenbankvorlesung?«. Beispiele von Aktualisierungen sind »Ändere die Klasse von Smith auf Sophomore«; »Erstelle eine neue Übungsgruppe für die Datenbankvorlesung in diesem Semester«; und »Speichere die Zensur A für Smith in der Datenbankübung des letzten Semesters ab«. Diese informellen Abfragen und Aktualisierungen müssen in der Datenbanksystemssprache genau spezifiziert werden, bevor sie vom DBMS verarbeitet werden können.

## 1.3 Merkmale des Datenbankansatzes

Eine Reihe von Merkmalen unterscheiden den Datenbankeinsatz vom herkömmlichen Ansatz der Programmierung mit Dateien. Bei der herkömmlichen **Dateiverarbeitung** definiert und implementiert jeder Benutzer die Dateien, die für eine bestimmte Anwendung benötigt werden, als Teil der Programmierung der Anwendung. Ein Benutzer, z.B. das *Grade Reporting Office*, könnte eine Datei über Studenten und ihre Zensuren verwalten. Programme zum Ausdrucken der Zeugnisse eines Studenten und zur Eingabe neuer Zensuren in die Datei werden implementiert. Ein zweiter Benutzer, etwa das Accounting Office, führt beispielsweise die Gebühren und Zahlungen von Studenten. Obwohl beide Benutzer an Daten über Studenten interessiert sind, pflegt jeder von ihnen getrennte Dateien und Programme, um diese Dateien zu manipulieren, weil jeder bestimmte Daten benötigt, die nicht in den Dateien des anderen Benutzers verfügbar sind. Diese Redundanz bei der Definition und Speicherung von Daten führt zur Verschwendungen von Speicherplatz und redundantem Arbeitsaufwand, um die Daten auf dem neuesten Stand zu halten.

Beim Datenbankansatz wird ein einziger Datenbestand für die Beschreibung der Tabellen vorgehalten, der einmal definiert wird und dann jedem Benutzer zugänglich ist. In den folgenden Unterabschnitten werden die wesentlichen Merkmale des Datenbankansatzes gegenüber dem Dateiverarbeitungsansatz beschrieben.

### 1.3.1 Der Datenbankkatalog

Ein grundlegendes Merkmal des Datenbankansatzes ist, dass das Datenbanksystem nicht nur die Datenbank selbst, sondern auch eine vollständige Definition der Datenbankstruktur und der Einschränkungen bezüglich der Daten enthält. Diese Definition ist in dem so genannten **Datenbankkatalog** (auch »Systemkatalog«) gespeichert, der Informationen wie die Struktur der einzelnen Tabellen, Typ und Speicherformat jedes Datenelements und verschiedene Einschränkungen bezüglich der Daten enthält. Die im Katalog gespeicherten Informationen nennt man **Metadaten**; sie beschreiben die Struktur der Datenbank (siehe Abbildung 1.1).

Der Katalog wird vom DBMS und auch von den Datenbanknutzern verwendet, die Informationen über die Datenbankstruktur benötigen. Ein universelles DBMS wird nicht für eine spezifische Datenbankanwendung geschrieben; es muss daher auf die Katalogdaten zugreifen, um die Struktur der Dateien einer spezifischen Datenbank zu kennen, z.B. Typ und Format der Daten, auf die sie zugreifen will. Das DBMS muss gleichermaßen gut jede beliebige *Datenbank*, z.B. die Datenbank einer Universität, einer Bank oder einer Firma, verwalten können, solange die Datenbankstruktur im Katalog gespeichert ist.

Bei der traditionellen Dateiverarbeitung ist die Datendefinition normalerweise Teil der Anwendungsprogramme. Folglich sind diese Programme auf die Zusammenarbeit mit nur *einer spezifischen Datenbank*, deren Struktur in den Anwendungsprogrammen deklariert ist, beschränkt. In einem Pascal-Programm können z.B. Datensatzstrukturen dafür deklariert sein; in einem C++-Programm gibt es z.B. die Deklarationen »struct« oder »class«; und ein COBOL-Programm enthält z.B. »Data-Division«-Anweisungen, um Dateien zu definieren. Während Dateiverarbeitungssoftware nur auf vorher festgelegte Datensammlungen zugreifen kann, kann ein DBMS auf verschiedene Datenbanken zugreifen. Im Katalog sind die Tabellendefinitionen gespeichert, die somit von jedem Programm verwendet werden können.

In dem Beispiel von Abbildung 1.2 speichert das DBMS die Definitionen aller dargestellten Tabellen im Katalog. Bei jeder Anfrage, beispielsweise nach dem Namen eines Studenten, sieht das DBMS im Katalog nach, um die Struktur der Tabelle STUDENT sowie Position und Größe des Datenelements NAME in einem STUDENT-Datensatz festzustellen. Demgegenüber sind in einer typischen Dateiverarbeitungsanwendung die Dateistruktur und im Extremfall die genaue Stelle des Namens in einem STUDENT-Datensatz in jedem Programm, das auf dieses Datenelement zugreift, fest kodiert.

### 1.3.2 Datenisolation und Datenabstraktion

Bei der traditionellen Dateiverarbeitung ist die Struktur von Datendateien in den Anwendungsprogrammen eingebettet, so dass Änderungen der Struktur einer Datei möglicherweise zur Änderung *aller Programme führt*, die auf diese Datei zugreifen. Demgegenüber sind solche Änderungen bei DBMS-gestützten Programmen in den meisten Fällen nicht nötig. Wir nennen dieses Merkmal **Programm/Daten-Unabhängigkeit**. Bei dateiorientiertem Zugriff kann z.B. ein Programm so geschrieben werden, dass es nur auf STUDENT-Datensätze mit der in Abbildung 1.3 dargestellten Struktur zugreifen kann. Wenn wir ein weiteres Datenelement zur Struktur des STUDENT-Datensatzes hinzufügen, z.B. das Geburtsdatum, stimmt die Beschreibung des Datensatzes im Programm nicht mehr mit der aktuellen überein und muss geändert werden. Demgegenüber brauchen wir in einer DBMS-Umgebung lediglich die Beschreibung von STUDENT-Datensätzen im Katalog zu ändern, um das weitere Datenelement »Geburtsdatum« aufzunehmen; es müssen keine Programme geändert werden. Wenn das DBMS beim nächsten Mal auf den Katalog zugreift, wird die neue Struktur für STUDENT-Datensätze benutzt.

**Abbildung 1.3:** Internes Speicherformat für einen STUDENT-Datensatz.

Name des Datenelements	Anfangsposition im Datensatz	Länge in Zeichen (Byte)
Name	1	30
StudentNumber	31	4
Class	35	4
Major	39	4

In objektorientierten und objektrelationalen Datenbanken (siehe Teil 3) können Operationen mit Daten im Rahmen der Datenbankdefinitionen definiert werden. Eine **Operation** (auch als *Funktion* bezeichnet) wird in zwei Teilen definiert. Die **Schnittstelle** (oder *Signatur*) einer Operation beinhaltet die Namen der Operation und die Datentypen ihrer Argumente (oder Parameter). Die **Implementierung** (oder *Methode*) der Operation wird getrennt spezifiziert und kann geändert werden, ohne dass sich dies auf die Schnittstelle auswirkt. Anwendungsprogramme können mit den Daten dadurch operieren, dass sie diese Operationen durch ihren Namen und ihre Argumente aufrufen, ungeachtet dessen, wie die Operationen implementiert werden. Dies wird als **Programm/Operation-Unabhängigkeit** bezeichnet.

Die Möglichkeit der Programm/Daten- und Programm/Operation-Unabhängigkeit wird als **Datenabstraktion** bezeichnet. Ein DBMS bietet den Benutzern eine **konzeptuelle Sicht auf die Daten**, die keine Details über die Speicherung der Daten oder die Implementierung der Operationen beinhaltet. Informell ist ein **Datenmodell** eine Form der Datenabstraktion, um die konzeptuelle Darstellung zu ermöglichen. Das Datenmodell verwendet logische Konzepte, wie beispielsweise Objekte und deren Eigenschaften und Beziehungen untereinander, die für die meisten Benutzer leichter als rechnerorientierte Speicherdetails verständlich sind. Folglich *verbirgt* das Datenmodell Speicher- und Implementierungsdetails, die für die Datenbanknutzer nicht von Interesse sind.

Man betrachte als Beispiel wieder Abbildung 1.2. Die interne Implementierung einer Datei wird durch die Datensatzlänge – die Anzahl von Zeichen (Bytes) in jedem Datensatz – bestimmt und jedes Datenelement kann durch sein Anfangsbyte innerhalb eines Datensatzes und seiner Länge in Bytes spezifiziert werden. Der STUDENT-Datensatz würde folglich wie in Abbildung 1.3 dargestellt werden. Ein typischer Datenbanknutzer ist aber nicht an der Position jedes Datenelements innerhalb eines Datensatzes oder an seiner Länge interessiert. Für ihn ist vielmehr von Interesse, dass beispielsweise bei einer Referenz auf den Namen von STUDENT der richtige Wert zurückgegeben wird. Eine konzeptuelle Darstellung der STUDENT-Datensätze sehen Sie in Abbildung 1.2. Weitere Details der Dateispeicherungsorganisation, wie beispielsweise die für eine Datei spezifizierten Zugriffspfade, können vom DBMS vor den Datenbanknutzern verborgen werden. Speicherungsdetails werden ausführlich in Kapitel 5 und 6 beschrieben.

Beim Datenbankansatz werden die detaillierte Struktur und die Organisation jeder Datei im Katalog gespeichert. Datenbanknutzer referenzieren die konzeptuelle Darstellung der Tabellen und das DBMS extrahiert die Details der Dateispeicherung aus dem Katalog, wenn sie vom DBMS benötigt werden. Unterschiedliche Datenmodelle können benutzt werden, um diese Datenabstraktion für Datenbanknutzer bereitzustellen. Ein großer Teil dieses Buchs ist der Darstellung verschiedener Datenmodelle und ihren Konzepten für die Abstraktion der Darstellung von Daten gewidmet.

Angesichts der heutigen Tendenz zu objektrelationalen Datenbanken wird die Abstraktion dahingehend um eine Ebene erweitert, dass nicht nur die Daten, sondern auch die Operationen auf den Daten miteinbezogen werden. Diese Operationen bieten eine Abstraktion von Miniweltaktivitäten aus Benutzersicht. Eine Operation CALCULATE\_GPA kann beispielsweise auf ein Objekt STUDENT angewandt werden, um den »Grade Point Average« zu berechnen. Solche Operationen lassen sich durch Benutzeranfragen oder Programme aufrufen, ohne dass der Benutzer die Details über die interne Implementierung wissen muss. In diesem Sinn wird dem Benutzer eine Abstraktion der Miniweltaktivität als **abstrakte Operation** zur Verfügung gestellt.

### 1.3.3 Unterstützung mehrerer Datensichten

Eine Datenbank hat normalerweise viele Benutzer, die jeweils eine andere Perspektive oder Sicht (View) der Datenbank erfordern. Ein **View** kann beispielsweise eine Teilmenge (Subset) der Datenbank sein oder **virtuelle Daten** enthalten, die aus Datenbankdateien stammen, aber nicht explizit gespeichert sind. Einige Benutzer müssen vielleicht nicht einmal wissen, ob die Daten, auf die sie zugreifen, gespeichert oder abgeleitet sind. Ein Mehrbenutzer-DBMS, dessen Benutzer mit vielen verschiedenen Anwendungen arbeiten, muss Möglichkeiten zur Definition **mehrerer Views** bereitstellen. Beispielsweise ist ein Benutzer der Datenbank aus Abbildung 1.2 vielleicht nur an den Transcripts der Studenten interessiert. Der View für diesen Benutzer ist in Abbildung 1.4 (a) dargestellt. Ein zweiter Benutzer, der nur überprüfen muss, ob alle Voraussetzungen für die Kurse erfüllt wurden, benötigt den in Abbildung 1.4 (b) dargestellten View.

**Abbildung 1.4:** Zwei Views der Beispieldatenbank aus Abbildung 1.2; (a) der View für die Transcripts; (b) der View für die Kursvoraussetzungen.

(a)	TRANSCRIPT	StudentName	Student Transcript				
			CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119	
		MATH2410	B	Fall	99	112	
	Brown	MATH2410	A	Fall	98	85	
		CS1310	A	Fall	98	92	
		CS3320	B	Spring	99	102	
		CS3380	A	Fall	99	135	

(b)	PREREQUISITES	CourseName	CourseNumber	Prerequisites
				CS3320
		Database	CS3380	MATH2410
		Data Structures		CS1310

### 1.3.4 Mehrbenutzer

Wie die Bezeichnung andeutet, muss ein Mehrbenutzer-DBMS mehreren Benutzern gleichzeitig Zugriff auf die Datenbank ermöglichen. Dies ist notwendig, wenn Daten für mehrere Anwendungen in einer einzigen Datenbank eingegeben und gepflegt werden sollen. Das DBMS muss eine Software für die **Nebenläufigkeitskontrolle** (Concurrency Control) beinhalten. Dies soll sicherstellen, dass mehrere Benutzer, die die gleichen Daten aktualisieren wollen, dies auf kontrollierte Weise tun, damit das Ergebnis der Aktualisierungen korrekt ist. Wenn beispielsweise mehrere Reservierungsmitarbeiter versuchen, einen Platz in einem Flugzeug zu reservieren, sollte das DBMS sicherstellen, dass jeder Platz nur von einem Mitarbeiter einem Passagier zugeteilt wird. Diese Art von Anwendungen nennt man allgemein **OLTP-Anwendungen (Online-Transaktionsverarbeitung)**. Eine grundlegende Aufgabe jeder Mehrbenutzer-DBMS-Software ist die Sicherstellung, dass gleichzeitig durchgeföhrte Transaktionen korrekt ablaufen.

Die oben beschriebenen Merkmale sind sehr wichtig für die Unterscheidung eines DBMS von traditioneller Dateiverarbeitungssoftware. Abschnitt 1.6 beschreibt weitere Funktionen, die ein DBMS charakterisieren. Zuvor werden aber die verschiedenen Personen kategorisiert, die in einer Datenbankumgebung arbeiten.

## 1.4 Akteure auf der Bühne

Für eine kleine persönliche Datenbank, wie beispielsweise die in Abschnitt 1.1 erwähnte Adressliste, wird die Datenbank meist von einer Person definiert und erstellt. Gleches gilt für den Zugriff darauf. Am Entwurf, der Benutzung und der Pflege einer großen Datenbank mit ein paar hundert Benutzern sind demgegenüber viele Personen beteiligt. In diesem Abschnitt beschreiben wir die Personen, die täglichlich von einem Datenbanksystem Gebrauch machen; wir nennen sie zusammen »Akteure auf der Bühne«. In Abschnitt 1.5 werden Personen identifiziert, die wir zusammenfassend als »Akteure hinter der Bühne« bezeichnen, d.h. Personen, die an der Pflege der Datenbanksystemumgebung beteiligt, aber nicht selbst an der Nutzung der Datenbank interessiert sind.

### 1.4.1 Datenbankverwalter

In jeder Organisation, in der viele Personen die gleichen Ressourcen benutzen, besteht eine Notwendigkeit für einen übergeordneten Verwalter, der diese Ressourcen überwacht und verwaltet. In einer Datenbankumgebung ist die primäre Ressource die Datenbank und die sekundäre Ressource das DBMS und die damit verbundene Software. Die Verwaltung dieser Ressourcen obliegt dem **Datenbankverwalter oder auch Datenbankadministrator (DBA)**. Der DBA ist zuständig für die Verteilung von Zugriffsrechten auf die Datenbank, für die Koordination und Überwachung ihrer Nutzung und für die Anschaffung von benötigten Software- und Hardware-Ressourcen. Der DBA ist in der Regel der Ansprechpartner bei Problemen wie Verletzung der Sicherheit oder schlechtes Leistungsverhalten des Datenbanksystems. In Großunternehmen wird der DBA durch Mitarbeiter unterstützt, die jeweils bestimmte Teilfunktionen übernehmen.

### 1.4.2 Datenbankdesigner

**Datenbankdesigner** sind für die Festlegung der in der Datenbank zu speichernden Daten und die Auswahl entsprechender Strukturen zur Darstellung und Speicherung dieser Daten verantwortlich. Diese Aufgaben werden notwendig, ehe die Datenbank tatsächlich realisiert und die Daten in der Datenbank gespeichert werden können. In dem Zuständigkeitsbereich des Datenbankdesigners fällt vor allem auch die Kommunikation mit allen potenziellen Datenbanknutzern, um ihre Anforderungen zu verstehen und einen entsprechenden Entwurf der Datenbank zu entwickeln. In vielen Fällen sind die Designer Mitarbeiter des DBA, denen nach Fertigstellung des Datenbankentwurfs andere Aufgaben zugeteilt werden. Datenbankdesigner interagieren normalerweise mit jeder potenziellen Benutzergruppe und entwickeln **Sichten (Views)** der Datenbank, die die Daten- und Verarbeitungsanforderungen der jeweiligen Gruppe erfüllen. Diese Views der verschiedenen Benutzergruppen werden dann analysiert und *integriert*. Der endgültige Datenbankentwurf muss den Anforderungen aller Benutzergruppen genügen.

### 1.4.3 Endbenutzer

Endbenutzer sind diejenigen Personen, deren Aufgabenbereich den Zugriff auf die Datenbank voraussetzt, um Berichte (Reports) abzurufen, zu aktualisieren und zu erzeugen; sie nutzen die Datenbank zur Erledigung ihrer Aufgaben beispielsweise innerhalb eines Betriebs. Endbenutzer können in mehrere Kategorien gegliedert werden:

- **Gelegentliche Endbenutzer:** Diese greifen gelegentlich auf die Datenbank zu; in vielen Fällen wird dabei auf unterschiedliche Daten zugegriffen. Um diesen Zugriff ohne großen Aufwand und Vorkenntnisse zu ermöglichen, muss dem gelegentlichen Endbenutzer eine ausdrucksstarke, aber einfach zu handhabende Möglichkeit des Datenzugriffs zur Verfügung gestellt werden. Gelegentliche Endbenutzer sind typischerweise leitende Mitarbeiter im mittleren oder gehobenen Management, aber auch Mitarbeiter auf der Suche nach bestimmten Informationen.
- **Naive oder parametrische Endbenutzer:** Diese Kategorie stellt eine relativ große Benutzergruppe dar. Ihre Hauptaufgabe ist durch immer wiederkehrende Anfragen und Aufgaben bei der Aktualisierung der Datenbank charakterisiert; sie verwenden dabei fest vordefinierte Anfragen und Standardprogramme zur Aktualisierung der Daten – so genannte **Standardtransaktionen** (Canned Transactions) –, die zuvor sorgfältig programmiert und getestet wurden. Die Aufgaben, die diese Benutzer ausführen, sind vielfältig:
  - Mitarbeiter an Bankschaltern prüfen Kontosalden und buchen Abhebungen und Einlagen von Zahlungsmitteln.
  - Mitarbeiter in Reservierungsbüros für Fluggesellschaften, Hotels und Leihwagenfirmen prüfen die Verfügbarkeit für eine bestimmte Nachfrage und führen Reservierungen durch.
  - Mitarbeiter an Empfangsstationen von Kurierfirmen geben Identifizierungen für Post- und Paketsendungen über Strichcodes und beschreibende Informationen ein und aktualisieren eine zentrale Datenbank für Sendungen, die sich entweder auf dem Weg zum Empfänger befinden oder diesem gerade zugestellt wurden.
- **Professionelle Endbenutzer:** Zu dieser Gruppe zählen Ingenieure, Wissenschaftler, Wirtschaftsanalysten und Fachleute aus anderen Gebieten, die sich eingehend mit den Einrichtungen des DBMS vertraut machen, um ihre Anwendungen entsprechend ihrer komplexen Anforderungen zu implementieren.
- **Einzelbenutzer:** Die Benutzer dieser Kategorie führen persönliche Datenbanken, wobei sie kommerziell erhältliche Programmpakete mit leicht zu bedienenden bzw. grafischen Benutzeroberflächen benutzen. Ein Beispiel ist der Benutzer eines Steuer- und Finanzverwaltungspakets, das eine Reihe persönlicher Finanzdaten für Steuerzwecke verwaltet.

Ein typisches DBMS bietet mehrere Möglichkeiten für den Zugriff auf die Datenbank. Naive Endbenutzer müssen sehr wenig über die vom DBMS bereitgestellten Schnittstellen erlernen. Sie brauchen nur die Standardtransaktionen zu verstehen, die für sie entworfen und implementiert wurden. Gelegentliche Benutzer lernen nur einige

wenige Möglichkeiten zum Zugriff auf die Datenbank kennen, die sie wiederholt benutzen. Professionelle Benutzer hingegen versuchen, möglichst viele DBMS-Funktionen zu erlernen, um ihre komplexen Anforderungen umsetzen zu können. Einzelbenutzer wiederum eignen sich im Allgemeinen sehr gute Kenntnisse in der Verwendung einer bestimmten Software an, um damit auf die Datenbank zuzugreifen.

#### 1.4.4 Systemanalytiker und Anwendungsprogrammierer

**Systemanalytiker** ermitteln die Anforderungen von Endbenutzern, insbesondere der naiven und parametrischen, und entwickeln Spezifikationen für vorgeplante Transaktionen, die diese Anforderungen erfüllen. **Anwendungsprogrammierer** implementieren diese Spezifikationen als Programme; dann testen, diagnostizieren, dokumentieren und pflegen sie diese vorgeplanten Transaktionen. Solche Analytiker und Programmierer (die man heutzutage als **Software-Engineers** bezeichnet) sollten mit der vollen Palette der vom DBMS bereitgestellten Fähigkeiten vertraut sein.

### 1.5 Akteure hinter der Bühne

Neben den Personen, die sich mit dem Entwurf, der Nutzung und der Administration einer Datenbank befassen, gibt es einen Personenkreis, der an Entwurf, Entwicklung und Operation der *DBMS-Software und Systemumgebung* beteiligt ist. Diese Personen sind normalerweise aber nicht an der Datenbank selbst interessiert. Wir nennen sie »Akteure hinter der Bühne«; zu ihnen zählen die folgenden Kategorien:

- **Designer und Implementierer eines DBMS:** Dies sind Personen, die die DBMS-Module und -Schnittstellen als Software-Paket entwerfen und implementieren. Ein DBMS ist ein komplexes Software-System, das sich aus vielen Komponenten oder **Modulen** zusammensetzt, einschließlich solcher für die Verwaltung des Datenbankkatalogs, der Verarbeitung von Anfragen in einer Anfragesprache, der Verarbeitung von Schnittstellenaufrufen, Module zum Datenzugriff, zur Nebenläufigkeitskontrolle, zur Wiederherstellung der Datenkonsistenz im Fehlerfall und zur Überwachung der Sicherheit. Das DBMS muss Schnittstellen zu anderer Systemsoftware, wie beispielsweise zum Betriebssystem und zu Compilern für die Übersetzung von Programmen in verschiedenen Programmiersprachen, zur Verfügung stellen.
- **Werkzeug-Entwickler:** Hierzu zählen Personen, die **Werkzeuge** – also Software-Pakete, die den Entwurf und die Nutzung eines Datenbanksystems vereinfachen und dessen Leistung verbessern – entwerfen und implementieren. Werkzeuge sind optionale Software-Pakete, die oft unabhängig vom DBMS beschafft werden müssen. Hierzu zählen Software-Pakete für den Datenbankentwurf, Leistungsüberwachung, Schnittstellen für natürliche oder grafische Spracheingaben sowie Software-Pakete zur Prototyperstellung, zur Simulation und Erzeugung von Testdaten. In vielen Fällen werden solche Werkzeuge von unabhängigen Software-Anbietern entwickelt und vertrieben.
- **Operatoren und Mitarbeiter zur Wartung:** Dies sind Mitarbeiter der Systemadministration, die für den eigentlichen Betrieb und die Wartung der Hard- und Software-Umgebung des Datenbanksystems verantwortlich sind.

Die obigen Kategorien von Akteuren hinter der Bühne sind an der Bereitstellung eines Datenbanksystems für Endbenutzer beteiligt, normalerweise aber nicht an der Nutzung der Datenbank für ihre eigenen Zwecke interessiert.

## 1.6 Vorteile eines DBMS

In diesem Abschnitt diskutieren wir einige Vorteile der Verwendung eines DBMS und die Eigenschaften, die ein gutes DBMS aufweisen sollte. Der DBA muss diese Möglichkeiten nutzen, um verschiedenen Anforderungen im Zusammenhang mit dem Entwurf, der Administration und dem Einsatz eines Mehrbenutzer-Datenbanksystems gerecht zu werden.

### 1.6.1 Redundanzkontrolle

In der traditionellen Dateiverarbeitung definiert und pflegt jede Benutzergruppe ihre eigenen Dateien, auf die dann durch entsprechende Anwendungen zugegriffen wird und die verarbeitet werden. Man betrachte beispielsweise die Universitätsdatenbank UNIVERSITY aus Abschnitt 1.2; hier wären »Course Registration Personnel« und »Accounting Office« zwei Benutzergruppen. Im dateiorientierten Ansatz führt jede Gruppe unabhängig Dateien über Studenten. Das Accounting Office speichert und verarbeitet Daten über die Registrierung von Studenten und notwendige Daten zur Abrechnung, während das Registration Office die Vorlesungen und Zensuren von Studenten verarbeitet. Ein Großteil der Daten wird also einmal in den Dateien jeder Benutzergruppe doppelt gespeichert. Weitere Benutzergruppen duplizieren möglicherweise einige oder alle Daten in ihren eigenen Dateien.

Diese **Redundanz** durch mehrmaliges Speichern der gleichen Daten führt zu mehreren Problemen. Erstens entsteht die Notwendigkeit, eine einzige logische Aktualisierung, z.B. die Eingabe von Daten über einen neuen Studenten, mehrmals auszuführen: einmal in jeder Datei, in der die Studentendaten erfasst werden. Dies wiederum führt zu *redundantem Arbeitsaufwand*. Zweitens wird *Speicherplatz* dadurch *verschwendet*, dass die gleichen Daten wiederholt gespeichert werden – ein schwerwiegendes Problem insbesondere bei großen Datenbanken. Drittens führen Veränderungen in Dateien, die die gleichen Daten speichern, zu *Inkonsistenzen*, da eine Aktualisierung in bestimmten, möglicherweise aber nicht in allen relevanten Dateien ausgeführt werden. Wird eine Aktualisierung, wie beispielsweise das Einfügen eines neuen Studenten – in allen relevanten Dateien ausgeführt, können dennoch die Daten über den Studenten *inkonsistent sein*, weil die Aktualisierungen unabhängig voneinander ausgeführt wurden. Eine Benutzergruppe gibt z.B. als Geburtsdatum des Studenten irrtümlich JAN-19-1974 ein, während eine andere mit JAN-29-1974 den richtigen Wert eingibt.

Beim Datenbankansatz werden die Sichten (Views) der verschiedenen Benutzergruppen bereits im Datenbankentwurf integriert. Der Konsistenz halber sollte ein Datenbankentwurf also sicherstellen, dass jedes logische Datenelement, wie beispielsweise Name oder Geburtsdatum eines Studenten, an *nur einer Stelle* in der Datenbank gespeichert wird. Dies lässt keine Inkonsistenz zu und spart Speicherplatz. In manchen Fällen kann eine **kontrollierte Redundanz** allerdings nützlich sein, um die Effizienz der Verarbeitung von Anfragen zu verbessern. Man könnte z.B. StudentName und CourseNumber redundant in einer Tabelle GRADE\_REPORT (Abbildung 1.5a) speichern. Mit großer Wahrscheinlichkeit benötigen wir möglicherweise nicht nur Namen

und Vorlesungsnummer (CourseNumber) des Studenten, sondern auch Zensur für eine Vorlesung sowie StudentNumber und SectionIdentifier. Werden alle Daten nur einmal gespeichert, müssen nicht mehrere Tabellen durchsucht werden, um die Daten zusammenzustellen. Für solche Fälle sollte das DBMS in der Lage sein, diese Redundanz so zu *kontrollieren*, dass Inkonsistenzen zwischen den Dateien verhindert werden. Dies wird dadurch erreicht, dass automatisch geprüft wird, ob die Werte des Studentennamens (StudentName) und der Studentennummer (StudentNumber) in einem GRADE\_REPORT-Datensatz in Abbildung 1.5 (a) mit den Werten von Name und StudentNumber in einem STUDENT-Datensatz (Abbildung 1.2) übereinstimmen. Ähnlich können die Werte in SectionIdentifier und CourseNumber in GRADE\_REPORT mit den SECTION-Datensätzen verglichen werden. Solche Prüfungen können während des Datenbankentwurfs spezifiziert und vom DBMS automatisch geprüft und sichergestellt werden, wann immer die Datei GRADE\_REPORT aktualisiert wird. Abbildung 1.5 (b) zeigt einen GRADE\_REPORT-Datensatz, der mit der STUDENT-Datei von Abbildung 1.2 inkonsistent ist und möglicherweise falsch eingegeben wird, weil *keine kontrollierte Redundanz* gegeben ist.

**Abbildung 1.5:** Redundante Speicherung von Datenelementen: (a) *kontrollierte Redundanz* für StudentName und CourseNumber in der Datei GRADE\_REPORT; (b) *unkontrollierte Redundanz*: ein GRADE\_REPORT-Datensatz ist mit den STUDENT-Datensätzen aus Abbildung 1.2 inkonsistent, denn der Name von StudentNumber 17 lautet Smith und nicht Brown.

(a)	GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
		17	Smith	112	MATH2410	B
		17	Smith	119	CS1310	C
		8	Brown	85	MATH2410	A
		8	Brown	92	CS1310	A
		8	Brown	102	CS3320	B
		8	Brown	135	CS3380	A

(b)	GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
		17	Brown	112	MATH2410	B

## 1.6.2 Zugriffsbeschränkungen

Greifen mehrere Benutzer auf die gleiche Datenbank zu, ist es häufig erforderlich, dass nicht jeder Benutzer auf alle Daten in der Datenbank zugreifen kann. Finanzdaten gelten meist als vertraulich, so dass nur bestimmten Personen der Zugriff auf solche Daten gewährt wird. Darüber hinaus haben einige Benutzer vielleicht nur das Recht, auf Daten zuzugreifen, während andere diese Daten sowohl abrufen als auch aktualisieren können. Folglich muss auch die Art der Zugriffsoperation – Zugriff oder Aktualisierung – kontrolliert werden. In der Regel erhalten einzelne Benutzer oder Benutzergruppen Zugriffsrechte auf die Datenbank, die durch Passwörter geschützt sind. Ein DBMS sollte ein **Sicherheits- und Autorisierungssubsystem** bereitstellen, das der DBA benutzen kann, um Zugriffsrechte und Einschränkungen festzulegen. Das DBMS sollte diese Einschränkungen dann automatisch überprüfen und deren

Einhaltung sicherstellen. Ähnliche Kontrollmechanismen lassen sich auf Anwendungen anwenden, die auf die Daten einer Datenbank zugreifen. So kann es z.B. nur DBA-Mitarbeitern gestattet werden, bestimmte **privilegierte Software**, z.B. für das Erzeugen neuer Zugriffssrechte, zu benutzen. Ähnlich kann der Zugriff auf die Datenbank durch naive Benutzer auf speziell für sie entwickelte vorgeplante Transaktionen eingeschränkt werden.

### 1.6.3 Persistente Speicherung von Programmobjekten und Datenstrukturen

Datenbanken lassen sich nutzen, um **persistente Speicher** für Programmobjekte und Datenstrukturen bereitzustellen. Dies ist einer der Hauptgründe für den Einsatz **objektorientierter Datenbanksysteme**. Programmiersprachen erlauben in vielen Fällen die Definition komplexer Datenstrukturen und der auf ihnen operierenden Funktionen (Methoden), wie z.B. Klassendefinitionen in C++. Die Werte von Programmvariablen werden nach dem Ende der Ausführung eines Programms verworfen, es sei denn, der Programmierer speichert sie ausdrücklich permanent in Dateien. Dies setzt oft die Konvertierung dieser komplexen Strukturen in ein für die Dateispeicherung geeignetes Format voraus. Besteht die Notwendigkeit, diese Daten später wieder zu lesen, muss der Programmierer eine Konvertierung vom Dateiformat in die Variablenstruktur des Programms durchführen. Objektorientierte Datenbanksysteme sind mit Programmiersprachen wie C++ und Java kompatibel und das DBMS führt die nötigen Konvertierungen automatisch durch. Folglich kann ein komplexes C++-Objekt permanent in einem objektorientierten DBS, wie beispielsweise ObjectStore oder O2 (wird jetzt als »Ardent« bezeichnet; siehe Kapitel 12), gespeichert werden. Ein solches Objekt wird als **persistent** bezeichnet, weil es nach Beendigung der Ausführung des Programms weiter funktioniert und später direkt von einem anderen C++-Programm aus darauf zugegriffen werden kann.

Die persistente Speicherung von Programmobjekten und Datenstrukturen ist eine wesentliche Funktion eines Datenbanksystems. Viele Datenbanksysteme der Vergangenheit konnten das Problem der unterschiedlichen Darstellung von Daten in einem Anwendungsprogramm und in einer Datenbank nicht zufriedenstellend lösen. Dieses Problem, das so genannte **Impedanz-Mismatch-Problem**, wird von objektorientierten Datenbanksystemen (OODBMS) gelöst. OODBMS bieten in der Regel **Kompatibilität** der Datenstrukturen mit einer oder mehreren objektorientierten Programmiersprachen an.

### 1.6.4 Ableitungen und Aktionen anhand von Regeln

Einige Datenbanksysteme erlauben die Definition von *Herleitungsregeln* für die *Ableitung* neuer Fakten aus Fakten, die in der Datenbank gespeichert sind. Solche Systeme werden als **deduktive Datenbanksysteme** bezeichnet. Beispielsweise kann es in unserem Universitätsbeispiel komplexe Regeln geben, um zu ermitteln, ob ein Student »on probation« ist. Diese Ermittlung kann *deklarativ* durch **Regeln** definiert werden, über die nach der Kompilation und Übersetzung durch das DBMS alle Studenten »on probation« ermittelt werden können. In einem anderen DBMS müsste man ausdrücklich einen *prozeduralen Programmcode* schreiben, um solche Anwendungen zu unterstützen. Ändern sich die Regeln in der »realen Welt«, so ist es im Allgemeinen einfacher, die deklarierten Herleitungsregeln zu ändern, als prozedurale Programmcodes

umzuschreiben. Eine noch bessere Funktionalität wird durch **aktive Datenbanksysteme** bereitgestellt, die sog. »aktive Regeln« bieten, die automatisch Aktionen einleiten können.

### 1.6.5 Mehrbenutzerschnittstellen

Da viele verschiedene Benutzer mit unterschiedlichen technischen Kenntnissen und Anforderungen auf eine Datenbank zugreifen, sollte ein DBMS eine Vielzahl von Benutzeroberflächen bereitstellen. Dazu zählen Anfragesprachen für gelegentliche Benutzer, Programmiersprachenoberflächen für Anwendungsprogrammierer, Formulare und Befehlscodes für parametrische Benutzer sowie menügesteuerte Oberflächen und solche in natürlichen Sprachen für Einzelbenutzer. Die auf Formularen basierenden und die menügesteuerten Benutzeroberflächen werden als **grafische Benutzeroberflächen** (Graphical User Interfaces, GUIs) bezeichnet. Für die Spezifikation von GUIs gibt es viele spezialisierte Sprachen und Umgebungen. Die Möglichkeit, auch über das Web auf eine Datenbank zuzugreifen, ist heute eine Grundvoraussetzung für den Einsatz eines DBMS.

### 1.6.6 Beziehungen zwischen Daten

Eine Datenbank kann zahlreiche Datensätze speichern, die auf vielerlei Art untereinander zusammenhängen. In Abbildung 1.2 ist zu erkennen, dass der Datensatz für Brown in der Tabelle STUDENT mit vier Datensätzen in der Tabelle GRADE\_REPORT logisch zusammenhängt. Ebenfalls hängt jeder SECTION-Datensatz mit einem COURSE-Datensatz und mit einer Reihe von GRADE\_REPORT-Datensätzen zusammen: einer für jeden Studenten, der die betreffende Übung (Section) abgeschlossen hat. Ein DBMS muss die Möglichkeit bieten, eine Vielzahl komplexer Beziehungen zwischen den Datensätzen zu definieren sowie zusammenhängende Daten schnell und effizient miteinander zu verbinden und zu aktualisieren.

### 1.6.7 Integritätsbedingungen

In den meisten Anwendungen ist es notwendig, Einschränkungen bezüglich der Daten zu definieren, die in der Datenbank gespeichert werden können. Diese Einschränkungen werden durch **Integritätsbedingungen** im DBMS definiert. Die einfachste Art einer Integritätsbedingung ist die Spezifikation eines Datentyps für jedes Datenelement. Für die Datenbank in Abbildung 1.2 können wir beispielsweise spezifizieren, dass der Wert des Datenelements Class in jedem Datensatz Student eine ganze Zahl (Integer) zwischen 1 und 5 und der Wert des Namens eine Zeichenkette von höchstens 30 alphabetischen Zeichen sein muss. Eine komplexere, häufig vorkommende Art der Einschränkung legt fest, dass jeder Datensatz in einer Tabelle mit Datensätzen in anderen Tabellen in Beziehung stehen **muss**. Für die Datenbank in Abbildung 1.2 kann dies beispielsweise bedeuten, dass »jeder Section-Datensatz mit mindestens einem Course-Datensatz in Beziehung stehen muss«. Ein andere Art der Einschränkung legt die Eindeutigkeit von Werten für Werte eines Datenelements fest, z.B. »jeder Course-Datensatz muss einen eindeutigen Wert für CourseNumber haben«, d.h., kein anderer Course-Datensatz in der Course-Tabelle besitzt den gleichen Wert. Diese Einschränkungen werden aus der Bedeutung oder **Semantik** der Daten und des von ihnen dargestellten Ausschnitts der realen Welt abgeleitet. Der Datenbankdesigner ist für die Festlegung von Integritätsbedingungen während des Datenbankent-

wurfsprozesses zuständig. Einige Einschränkungen können direkt durch das DBMS spezifiziert und automatisch überprüft werden; andere müssen möglicherweise durch Aktualisierungsprogramme oder zum Zeitpunkt der Dateneingabe überprüft werden.

Ein Datenelement kann fehlerhaft eingegeben werden und dennoch die spezifizierten Integritätsbedingungen erfüllen. Wenn ein Student beispielsweise die Note A erhalten hat, während irrtümlich die Note C in die Datenbank eingegeben wurde, kann das DBMS diesen Fehler *nicht* automatisch entdecken, weil C ein gültiger Wert für den Datentyp Note ist. Solche Dateneingabefehler können nur manuell aufgedeckt (wenn der Student die Note erhält und sich beschwert) und später durch Aktualisierung der Datenbank korrigiert werden. Demgegenüber kann das DBMS die Eingabe der Note Z automatisch ablehnen, weil Z kein gültiger Wert für den Datentyp Note ist.

### 1.6.8 Fehlererholung

Ein DBMS muss in der Lage sein, bei auftretenden Hard- oder Softwarefehlern in der Datenbank wieder einen korrekten Zustand herzustellen. Das **Datensicherungs- und Fehlererholungssubsystem** eines DBMS ist für die Wiederherstellung eines korrekten Datenbankzustands verantwortlich. Wenn das Computersystem beispielsweise bei der Ausführung eines komplexen Aktualisierungsprogramms ausfällt, muss das Fehlererholungssystem sicherstellen, dass die Datenbank wieder in den Zustand zurückgesetzt wird, in dem sie sich befand, als die Ausführung des Programms begann. Alternativ könnte das Fehlererholungssystem sicherstellen, dass das Programm wieder an dem Punkt weiter ausgeführt wird, an dem es unterbrochen wurde, so dass es zu einer vollständigen Ausführung des Programms mit allen seinen Aktionen auf der Datenbank kommt.

## 1.7 Vorteile des Datenbankeinsatzes

Zusätzlich zu den im vorherigen Abschnitt behandelten Aspekten hat die Verwendung eines DBMS weitere Vorteile.

**Nutzung von Standards** Der Einsatz eines DBMS ermöglicht es dem DBA, Standards bei allen Datenbanknutzern in einem Großunternehmen zu definieren und umzusetzen. Dies verbessert die Kommunikation und Zusammenarbeit zwischen verschiedenen Abteilungen, Projekten und Benutzern innerhalb einer Organisation. Standards können für Namen und Formate von Datenelementen, Anzeigeformate, Berichtsstrukturen, Fachbegriffe usw. definiert werden. Der DBA kann Standards in einer zentralen Datenbankumgebung leichter umsetzen und überprüfen als in einer Umgebung, in der jede Benutzergruppe Kontrolle über ihre eigenen Dateien und Programme hat.

**Kürzere Anwendungsentwicklungszeiten** Eines der vorrangigen Argumente für den Einsatz von DBMS sind die kürzere Entwicklungszeit bei neuen Anwendungen, z.B. die Entwicklung eines Programms zum Ausdruck von Daten aus der Datenbank und zur Erstellung eines neuen Berichts. Der Entwurf und die Implementierung einer neuen Datenbank können zunächst zeitaufwendiger sein als das Schreiben eines einzelnen spezialisierten Programms, das auf eine oder mehrere Dateien zugreift. Nachdem eine Datenbank aber einmal realisiert worden ist, ist im Allgemeinen wesentlich

weniger Zeit erforderlich, um neue Anwendungen unter Verwendung der DBMS-Schnittstellen zu erstellen. Der Zeitaufwand bei der Programmentwicklung mit Hilfe eines DBMS beträgt ca. ein Sechstel bis ein Viertel der Zeit im Vergleich zur Realisierung mit Hilfe von Dateien.

**Hohe Flexibilität** Es mag erforderlich sein, die Struktur einer Datenbank entsprechend sich ändernden Anforderungen zu modifizieren. Beispielsweise kann eine neue Benutzergruppe entstehen, die Daten benötigt, die sich momentan nicht in der Datenbank befinden. Als Konsequenz mag es erforderlich sein, die Datenbank um eine Tabelle oder um mehrere Datenelemente in einer vorhandenen Tabelle zu erweitern. Moderne DBMS erlauben bestimmte Arten der Änderung der Datenbankstruktur, ohne dass sich diese auf die gespeicherten Daten und die vorhandenen Anwendungsprogramme auswirken.

**Hohe Verfügbarkeit** Ein DBMS stellt die Datenbank allen Benutzern **gleichzeitig** zur Verfügung. Sobald die Aktualisierung eines Benutzers in der Datenbank fortgeschrieben ist, können alle anderen Benutzer auf diese Aktualisierung sofort zugreifen. Diese Verfügbarkeit aktueller Informationen ist in vielen transaktionsverarbeitenden Anwendungen, wie z.B. Reservierungssystemen oder Bankanwendungen, unabdingbar und wird durch die Nebenläufigkeitskontroll- und Fehlererholungssysteme eines DBMS ermöglicht.

**Große Wirtschaftlichkeit** Der DBMS-Einsatz erlaubt die Konsolidierung von Daten und Anwendungen, so dass sich der Umfang an Überlappungen zwischen den Aktivitäten von datenverarbeitenden Mitarbeitern in verschiedenen Projekten oder Abteilungen reduziert. Dies ermöglicht es dem gesamten Unternehmen, in leistungsstärkere Prozessoren, Speichergeräte oder Kommunikationstools zu investieren, statt jede Abteilung mit einem eigenen (schwächeren) Rechner auszustatten. Somit reduziert der Einsatz eines DBMS insgesamt die Betriebs- und Verwaltungskosten.

## 1.8 Wann ein DBMS nicht benutzt werden sollte

Trotz der Vorteile eines DBMS gibt es einige wenige Situationen, in denen ein solches System zu hohe bzw. unnötige Zusatzkosten mit sich bringt, die bei der traditionellen Dateiverarbeitung nicht entstehen würden. Die Zusatzkosten bei der Nutzung eines DBMS sind vor allem durch folgende Faktoren bestimmt:

- Hohe Anfangsinvestitionen für Hardware, Software und Schulung
- Die Universalität eines DBMS für die Definition und Verarbeitung von Daten
- Mehrkosten für die Bereitstellung von Sicherheit, Nebenläufigkeitskontrolle, Wiederherstellung und Integrität

Weitere Probleme können entstehen, wenn die Datenbankdesigner und der DBA die Datenbank nicht angemessen entworfen haben oder wenn die Anwendungen auf der Datenbank nicht richtig implementiert wurden. Folglich kann es wünschenswert sein, unter den folgenden Umständen reguläre Dateien zu verwenden:

- Die Datenbank und die Anwendungen sind einfach, wohl definiert und allen Erwartungen zufolge keinen Änderungen unterworfen.

- Bei einigen Programmen bestehen feste Echtzeitanforderungen, die aufgrund des hohen Zusatzaufwands in einem DBMS nicht erfüllt werden können.
- Ein gleichzeitiger Zugriff auf die Daten durch mehrere Benutzer ist nicht erforderlich.

## 1.9 Zusammenfassung

In diesem Kapitel wird eine Datenbank als eine Sammlung logisch zusammenhängender Daten definiert; mit *Daten* werden dabei Fakten aus der »realen Welt« bezeichnet, die es gilt, in der Datenbank widerzuspiegeln. Eine Datenbank repräsentiert einen gewissen Ausschnitt der »realen Welt« und wird für wohl definierte Zwecke von einer oder mehreren Benutzergruppen benutzt. Ein DBMS ist eine generalisierte Softwarekomponente für die Implementierung und Pflege einer computergestützten Datenbank. Die Datenbank und das DBMS bilden zusammen ein Datenbanksystem. Wir haben mehrere Merkmale identifiziert, durch die sich der Datenbankansatz von der traditionellen Dateiverarbeitung unterscheidet:

- Existenz eines Katalogs
- Programm/Daten- und Programm/Operation-Unabhängigkeit
- Datenabstraktion
- Unterstützung mehrerer Benutzersichten (Views)
- Gemeinsame Nutzung von Daten durch mehrere Transaktionen

Anschließend wurden die Hauptkategorien von Datenbanknutzern bzw. die »Akteure auf der Bühne« beschrieben:

- Administratoren
- Designer
- Endbenutzer
- Systemanalytiker und Anwendungsprogrammierer

Neben diesen Datenbanknutzern gibt es weitere Personengruppen, die zur Bereitstellung der Datenbankfunktionalität wichtig sind. Wir haben diese als »Akteure hinter der Bühne« bezeichnet:

- Designer und Implementierer eines DBMS
- Werkzeug-Entwickler
- Operateure und Wartungsmitarbeiter

Des Weiteren haben wir die Fähigkeiten aufgeführt, die dem DBA, den Datenbankdesignern und den Benutzern von der DBMS-gestützten Software für den Entwurf, die Verwaltung und die Nutzung einer Datenbank bereitgestellt werden sollten:

- Redundanzkontrolle
- Zugriffsbeschränkungen
- Persistente Speicherung von Programmobjekten und Datenstrukturen

- Ableitungen und Aktionen durch Verwendung von Regeln
- Mehrbenutzerverwaltung der Daten
- Darstellung komplexer Beziehungen zwischen Daten
- Integritätseinschränkungen
- Datensicherung und Wiederherstellung

Schließlich wurden einige weitere Vorteile des Datenbankansatzes im Vergleich zu traditionellen Dateiverarbeitungssystemen genannt:

- Bessere Möglichkeiten für die Umsetzung von Standards
- Kürzere Anwendungsentwicklungszeiten
- Höhere Flexibilität
- Größere Verfügbarkeit aktueller Daten für alle Benutzer
- Erhöhte Wirtschaftlichkeit

Zuletzt wurden die Aspekte des zusätzlichen Aufwands beim Einsatz eines DBMS diskutiert und einige Situationen beschrieben, in denen die Verwendung eines DBMS möglicherweise nicht vorteilhaft ist.

## WIEDERHOLUNGSFRAGEN

1. Definieren Sie folgende Begriffe: *Daten, Datenbank, Datenbankmanagementsystem (DBMS), Datenbanksystem, Katalog, Programm/Daten-Unabhängigkeit, Benutzersicht (View), DBA, Endbenutzer, Standardtransaktion, deduktives Datenbanksystem, persistentes Objekt, Metadaten, Transaktionsverarbeitungsanwendung*.
2. Welches sind die drei wichtigsten Aktionstypen einer Datenbank? Diskutieren Sie kurz jede einzelne Aktion.
3. Diskutieren Sie die Hauptmerkmale des Datenbankansatzes und auf welche Weise sich dieser vom Einsatz traditioneller Dateisysteme unterscheidet.
4. Für welche Aufgaben sind der DBA und der Datenbankdesigner zuständig?
5. Welche verschiedenen Typen von Datenbank-Endbenutzern gibt es? Diskutieren Sie die Hauptaktivitäten jeder Endbenutzerkategorie.
6. Diskutieren Sie die Funktionen, die ein DBMS bereitstellen sollte.

## ÜBUNGEN

7. Identifizieren Sie einige informelle Anfrage- und Aktualisierungsoperationen, die Sie erwartungsgemäß mit der Datenbank in Abbildung 1.2 ausführen könnten.
8. Welcher Unterschied besteht zwischen kontrollierter und unkontrollierter Redundanz? Erläutern Sie Ihre Antwort anhand von Beispielen.
9. Nennen Sie alle Beziehungen zwischen den Datensätzen der Datenbank in Abbildung 1.2.
10. Nennen Sie einige weitere Sichten (Views), die möglicherweise von anderen Benutzergruppen für die Datenbank in Abbildung 1.2 benötigt werden.
11. Nennen Sie einige Beispiele von Integritätsbedingungen, die Ihrer Meinung nach auf die Datenbank in Abbildung 1.2 angewandt werden sollten.

## AUSGEWÄHLTE LITERATUR

Die Ausgabe vom Oktober 1991 von *Communications of the ACM* und Kim (1995) beinhaltet mehrere Artikel, in denen DBMS »der nächsten Generation« beschrieben werden. Viele Datenbankmerkmale, die dort diskutiert werden, sind inzwischen kommerziell verfügbar. Die Ausgabe vom März 1976 von *ACM Computing Surveys* bietet eine frühe Einführung in Datenbanksysteme; interessierte Leser finden dort möglicherweise eine historische Perspektive des Themas.