Einleitung

Integrierte Informationsverarbeitung durch Datenbanksysteme

Im Gesamtsystem der betrieblichen Informationsverarbeitung spielen Datenbanksysteme eine zentrale Rolle. Einerseits dienen sie der Verwaltung der Stamm-, Bestands- und Bewegungsdaten, die bei dem Einsatz der verschiedenen betrieblichen Informationssysteme – insbesondere auch den die Leistungserstellung unterstützenden operativen Systemen bzw. ERP-Systemen – benötigt und verarbeitet werden. Zudem beinhalten Datenbanksysteme Funktionalitäten, die der Sicherstellung der Konsistenz und der Integrität der verwalteten und verarbeiteten Daten dienen. Schließlich ermöglichen Datenbanksysteme einen Zugriff unterschiedlicher Informationssysteme auf einen (umfassenden) Datenbestand. Insofern sind sie ein zentrales Mittel zur Realisierung des Grundgedankens der integrierten betrieblichen Informationsverarbeitung, d.h. eines Verarbeitungskonzeptes, das z.B. für die umfassende informationstechnologische Unterstützung der betrieblichen Leistungserstellung unabdingbar ist.

Datenbankkonzept

Das der Realisierung von Datenbanksystemen zugrunde liegenden Datenbank-konzept umfasst zwei Kernelemente:

- eine Datenbank als zentrale Datensammlung einschließlich eines die Struktur der Datensammlung bestimmenden Datenmodells und
- ein Datenbankverwaltungssystem zum Aufbau, zur Pflege und zur Erschließung der Datenbank für die Gesamtheit der Datenbankbenutzer.

Ziele des Datenbankkonzepts

Mit dem Datenbankkonzept verfolgt man u.a. folgende, die Gesamtheit der betrieblichen Datenverarbeitungsprozesse betreffenden Ziele:

- Beherrschung des Massendatenaufkommens in einem Unternehmen.
- Strukturierung der gesamten Datenwelt eines Unternehmens in Form eines sogenannten konzeptionellen Schemas.
- Gewährleistung der Auswertbarkeit der Daten für beliebige Anwendungszwecke.
- Sicherstellung der Korrektheit und der Konsistenz der Daten.

Die Entwicklung und die Ausgestaltung des Datenbankkonzepts verliefen in mehreren Stufen. Zu Beginn entwickelte man hierarchische Systeme, die lediglich die Verwendung restriktiver, hierarchischer Datenstrukturen gestatteten. Mehr Freiheitsgrade bei der Datenmodellierung eröffneten die dann folgenden netzwerkartigen Systeme. Doch eine grundlegende Auseinandersetzung mit dem Problem der redundanzarmen Datenmodellierung fand erst bei der Konzipierung der heute in der Regel eingesetzten relationalen Systeme statt.

Heute ist die betriebliche Datenverarbeitung ohne den Einsatz von Datenbanken nicht mehr denkbar. Datenbankgestützte Anwendungssysteme dominieren im gesamten betrieblichen Anwendungsspektrum, von laufzeitintensiven, interaktionsarmen (Standard-) Anwendungen bis hin zu dialogintensiven Abfragesystemen.

Ziel des Kurses

Ziel des vorliegenden Kurses ist es nun nicht, möglichst viele Datenbanksysteme und Datenbanksprachen zu behandeln, und auch nicht, Details über die Funktionsweise von Datenbanksystemen zu vermitteln. Neben der Darstellung der prinzipiellen Funktionsweise und der Architektur von Datenbanksystemen stehen vielmehr die Datenmodellierung, die Datenmanipulation und die Datenabfrage im betrieblichen Anwendungsbezug

im Vordergrund. Zugrunde gelegt werden hierbei ein erweitertes relationales Datenmodell und die Abfragesprache SQL, die mittlerweile als Standardabfragesprache gelten kann.
Sämtliche theoretischen Ausführungen zur relationalen Datenmodellierung und zur Datenmanipulation mit SQL werden an einem praxisorientierten Beispiel demonstriert. Mit dem Beispiel verknüpfen sich zwei Ziele. Einerseits soll es das Erlernen der behandelten Vorgehensweisen und Sprachelemente erleichtern. Andererseits soll es eine Hilfestellung bei der selbständigen Anwendung der vermittelten Kenntnisse - beispielsweise bei der Anfertigung von Diplomarbeiten im Bereich Wirtschaftsinformatik - geben.

Gliederung des Kurses

Der Kurs in vier Kurseinheiten mit insgesamt sechs Kapiteln gegliedert.

erste Kurseinheit

In der ersten Kurseinheit wird im einleitenden Kapitel 1 die Entwicklung von Datenbanksystemen in den Stufen Datenverwaltung durch Anwendungsprogramme, Einsatz von Dateiverwaltungssystemen und Datenbankkonzept beleuchtet. Auf die Architektur von Datenbanksystemen geht das Kapitel 2 ein. Vorgestellt wird insbesondere das sogenannte ANSI-Architekturmodell, das mit seiner 3-Ebenen-Architektur sowie mit seinen ebenenbezogenen Datenbanksprachen und personellen Instanzen einen Meilenstein in der Entwicklung der Datenbank-Technologie darstellt.

zweite Kurseinheit

Gegenstand der aus dem Kapitel 3 bestehenden zweiten Kurseinheit ist die logische Datenorganisation. Behandelt werden einige Grundzusammenhänge der Datenmodellierung sowie das hierarchische, das netzwerkartige und das relationale Datenmodell. Besonderes Gewicht wird hierbei auf das klassische Relationenmodell und auf ein erweitertes Relationenmodell gelegt.

dritte Kurseinheit

Auf die Datenbankbenutzung geht die aus dem Kapitel 4 bestehende dritte Kurseinheit ein. Neben allgemeineren Aspekten der Datenbankbenutzung, die vor allem die Arten der Datenbankbenutzer und die Formen der Datenbankbenutzung betreffen, werden eingehend zwei Datenbanksprachen behandelt. Zum einen die speziell auf Relationensysteme zugeschnittene Relationenalgebra und zum anderen die ebenfalls für relationale Systeme entwickelte Sprache SQL.

vierte Kurseinheit

Die vierte Kurseinheit umfasst zwei Kapitel. Das Kapitel 5 thematisiert die Abbildung konzeptionell-logischer Datenstrukturen auf physische Strukturen unter Verwendung bekannter Methoden der physischen Datenorganisation. Der Problemkreis der Integrität von Datenbanken mit den drei Integritätsebenen "Datenkonsistenz", "Datensicherheit" und "Datenschutz" wird im Kapitel 6 aufgegriffen. Im Vordergrund stehen hierbei Fragen der Datenkonsistenz sowie die Formulierung von Konsistenzbedingungen mit der Sprache SQL.

Der Kurs ist so abgefasst, dass sich ein ergänzendes Literaturstudium erübrigt. Die Literaturhinweise am Ende der einzelnen Kurseinheiten sind in zwei Gruppen untergliedert. Die Hinweise der ersten Gruppe beziehen sich auf die im jeweiligen Lehrtext zitierte Literatur und die Hinweise der zweiten Gruppe auf weiterführende Literatur.

Lernziele

Nach dem Durcharbeiten dieser Kurseinheit sollen Sie darlegen können, in welchen Stufen sich die Entwicklung von den konventionellen Formen der betrieblichen Datenverarbeitung zu der heute üblichen datenbankgestützten Datenverarbeitung vollzogen hat. Im Einzelnen sollen Sie die Entwicklungsstufen der automatisierten betrieblichen Datenverarbeitung, nämlich

Entwicklungsstufen der Datenverarbeitung

- die sequentielle Verarbeitung von Daten, welche durch die Anwendungsprogramme selbst verwaltet werden,
- die sequentielle und die nichtsequentielle Verarbeitung von Daten mit Hilfe von Dateiverwaltungsprogrammen und
- die datenbankgestützte Verarbeitung von Daten unter Anwendung des 2-Schichten-Konzepts und später des 3-Schichten-Konzepts,

Datenunabhängigkeit

charakterisieren und bewerten können. Außerdem sollen sie erläutern können, welche Bedeutung der physischen und der logischen Datenunabhängigkeit in der betrieblichen Datenverarbeitung zukommt und warum die physische bzw. logische Datenunabhängigkeit mittels des 2-Schichten- bzw. 3-Schichten-Konzepts verwirklicht werden kann.

Architektur von Datenbanksystemen

Schließlich sollen Sie die prinzipielle Architektur von Datenbanksystemen, bestehend aus einem abstrakten Speicher "Datenbank" und einem diesen Speicher für die Anwendungsprogramme erschließenden Datenbankverwaltungssystem, beschreiben und die Arbeitsweise von Datenbanksystemen in groben Zügen erklären können. Insbesondere sollen Sie die mit dem ANSI-Architekturmodell vorgegebene 3-Ebenen-Architektur von Datenbanken, welche

Schemata

- die ebenenweise Datenbeschreibung in Form von externen, konzeptionellen und internen Schemata

Datenbanksprachen

- mit Hilfe von Datenbeschreibungssprachen zur Beschreibung externer und konzeptioneller Schemata sowie von Speicherbeschreibungssprachen zur Definition interner Schemata

einschließt, darstellen und die im Ebenenübergang vorzunehmenden Datentransformationen erläutern können.

1 Entwicklung von Datenbanksystemen

Im Verlaufe eines halben Jahrhunderts hat der rasche Fortschritt im Bereich der Computertechnologie die Entwicklung umwälzender Verarbeitungskonzepte induziert, die das heutige Bild der betrieblichen Informationsverarbeitung prägen. Eines dieser Konzepte ist das Datenbankkonzept. Speicherte man zu Beginn der betrieblichen Datenverarbeitung Datenobjekte geringer Komplexität seriell auf Magnetbändern ab, so charakterisieren Begriffe wie "verteilte Datenbanken" und "Wissensbanken" die heutigen Möglichkeiten der Organisation und Nutzung hochkomplexer Datenstrukturen.

Einige wesentliche Schritte in dieser Entwicklung möchte die Abb. 1.1 verdeutlichen. Sie zeigt, dass in den späten sechziger Jahren - und zwar erst nachdem leistungsfähige externe Direktzugriffspeicher zur Verfügung standen - die ersten Datenbankverwaltungssysteme eingesetzt wurden. In den Jahren zuvor wurden mit der Entwicklung von effizienten Speicherungs- und Zugriffsverfahren die Voraussetzungen für solche Systeme geschaffen. Weite Verbreitung fand z.B. das auf einem hierarchischen Datenmodell beruhende System IMS der Firma IBM. Kurze Zeit nach seiner Markteinführung entwickelte die Gruppe CODASYL-DBTG (Data Base Task Group) einen Vorschlag zur Gestaltung von Datenbanksystemen, dem ein flexibleres netzwerkartiges Datenmodell zugrunde lag und der für mehrere Jahre Normcharakter besaß.

hierarchisches und netzwerkartiges Datenmodell

Mit der Vorstellung des relationalen Datenmodells von CODD zu Beginn der siebziger Jahre und des Entity-Relationship-Modells von CHEN in der Mitte der siebziger Jahre wurden die Grundlagen für eine Umwälzung geschaffen: die Verdrängung herkömmlicher hierarchischer und netzwerkartiger Systeme durch relationale Datenbanksysteme. In den frühen achtziger Jahren vollzog sich diese Umwälzung in massiver Form. Vor allem auch im Bereich der sich stürmisch ausbreitenden individuellen Datenverarbeitung setzte man nun (quasi-)relationale Datenbanksoftware ein. Schon nach relativ kurzer Zeit hatte das Dateisystem DBase auf Mikrocomputerebene die Bedeutung eines Standards erlangt.

relationales Datenmodell

In den achtziger Jahren kam auch das Konzept der Datenbankmaschinen auf. Darunter sind Rechnersysteme zu verstehen, deren Architektur speziell auf die Verwaltung großer Datenbanken zugeschnitten ist. Ebenso begann in diesen Jahren, stimuliert durch die zunehmende Vernetzung von Rechnersystemen, die Auseinandersetzung mit verteilten Datenbanksystemen. Schließlich scheinen Wissensbanken das zentrale Thema der neunziger Jahre abzugeben. Mit der softwaretechnischen Entwicklung gingen die Fortschritte bei der Entwicklung neuer Speichertechnologien einher.

Datenbankmaschinen

verteilte Datenbanksysteme

Nicht alle der genannten Entwicklungen und Konzepte können im vorliegenden Kapitel näher beschrieben werden. Das Augenmerk gilt vielmehr den wesentlichen Schritten, die zur Entwicklung von Datenbanksystemen geführt haben. Unterscheiden lassen sich etwa drei Entwicklungsstufen:

- die Dateiverwaltung durch Anwendungsprogramme,
- die Verwendung von Dateiverwaltungssystemen und
- die Verwendung von Datenbanksystemen.

Inhalt des Kapitels

Auf diese Konzepte gehen die Kapitel 1.1, 1.2 und 1.3 ein.

Entwicklung der Datenspeicherung und der Datenverwaltung

Ära	Speichermedium	Basiskonzepte
40er Jahre Magnetband-Ära	1949 Magnetband mit 4.200 Speicherstellen	serielle Datenaufzeichnung
50er Jahre Magnetplatten-Ära	1956 128 KB Trommelspeicher (fester Kopf) 1958 4 MB Festplattenspeicher (verschiebbarer Kopf)	Indizierung, Direktadressierung von Daten
60er Jahre Dateiverwaltungs-Ära	1962 42 MB Plattenspeicher 1964 7 MB Wechselplatten-speicher	1968 Datenbankverwaltungs-system IMS von IBM (hierarchisches Datenmo-dell) 1969 CODASYL-DBTG (netzwerkartiges Datenmo-dell)
70er Jahre Großrechner-DBVS-Ära	1973 58 MB Plattenspeicher 1976 100 MB Plattenspeicher 1978 200 MB Plattenspeicher	1970 relationales Datenmodell von CODD Datenschutzbestrebungen 1976 Entity-Relationship-Modell von CHEN
80er Jahre Microcomputer-DBVS-Ära	1981 2.5 GB Plattenspeicher 1984 1.2 MB Floppy 1985 100 MB Festplatten 5.25" 1985 Standard für CD-ROM 1987 720 MB CD-ROM	1980 DBase II Datenbankmaschinen verteilte Datenbanksysteme
90er Jahre Wissensverarbeitung	1990 optische Speicher 1991 10 MB Flashspeicher	Wissensbanken objektorientierte Programmierung

Abb. 1.1. Entwicklung der Datenmodellierung und der Datenverwaltung.

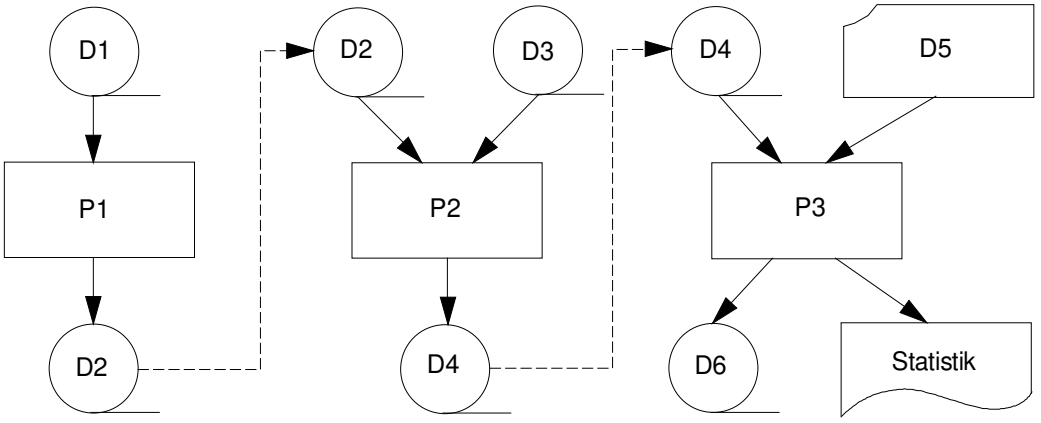
1.1 Dateiverwaltung durch Anwendungsprogramme

Charakterisierung der Dateiverwaltung durch Anwendungsprogramme

Ursprünglich verwaltete jedes Anwendungsprogramm die von ihm benutzten und extern abgelegten Daten selbst. Ein Programm enthielt also sämtliche Befehle und Datendefinitionen, die zum Schreiben von Daten auf einen Datenträger - anfänglich dem Magnetband - und zum Lesen von Daten erforderlich waren. Dieses Konzept der separaten Datenverwaltung veranschaulicht die Abb. 1.2.

Folgende Merkmale kennzeichnen die Datenverwaltung durch die Anwendungsprogramme selbst:

- Einzelne Dateien sind bestimmten Programmen exklusiv zugeordnet und werden nur von diesen Programmen benutzt.



Datenverwaltung durch Anwendungsprogramme selbst

Abb. 1.2. Separate Datenverwaltung durch Anwendungsprogramme.

- Jede der verarbeiteten Dateien besteht aus sequentiell abgespeicherten Datensätzen eines Typs.
- Die logische Struktur einer Datei, d.h. die logische Folge der Datensätze, stimmt mit der physischen Speicherungsfolge überein.
- Die Beschreibung einer Datei, d.h. die Beschreibung des Datensatzformates, der Sortierung und der Dateninhalte, ist in jedem Programm, welches die Datei verwendet, separat vorzunehmen.

Programme enthalten Datensatzformate und Befehle zur Dateimanipulation

- Jedes Programm, das eine Datei verarbeitet, enthält die zum Manipulieren der Datei erforderlichen Befehle.
- Unmittelbare menschliche Benutzer der Dateien sind ausschließlich Programmierer; andere (End-)Benutzer, die z.B. bestimmte Auswertungen benötigen, müssen die Dienste von Programmierern in Anspruch nehmen.

In Abb. 1.2 könnte der Zweck des Programms P1 z.B. in der Selektion von Datensätzen aus der Datei D1 bestehen, die in einem weiteren Verarbeitungsschritt - hier beispielsweise vollzogen mit dem Programm P2 - benötigt werden. Das Programm P2 möge dem Mischen zweier Dateien dienen. Erzeugt wird die Datei D4, die neben der Lochkartendatei D5, in den nächsten Verarbeitungslauf eingeht. Ergebnis dieses Laufs ist z.B. eine in D6 abgelegte und als Liste ausgegebene Statistik. Zur inhaltlichen Verdeutlichung dieser Verarbeitungsform sei die Abb. 1.3 betrachtet.

Fakturierung und Umsatzfortschreibung

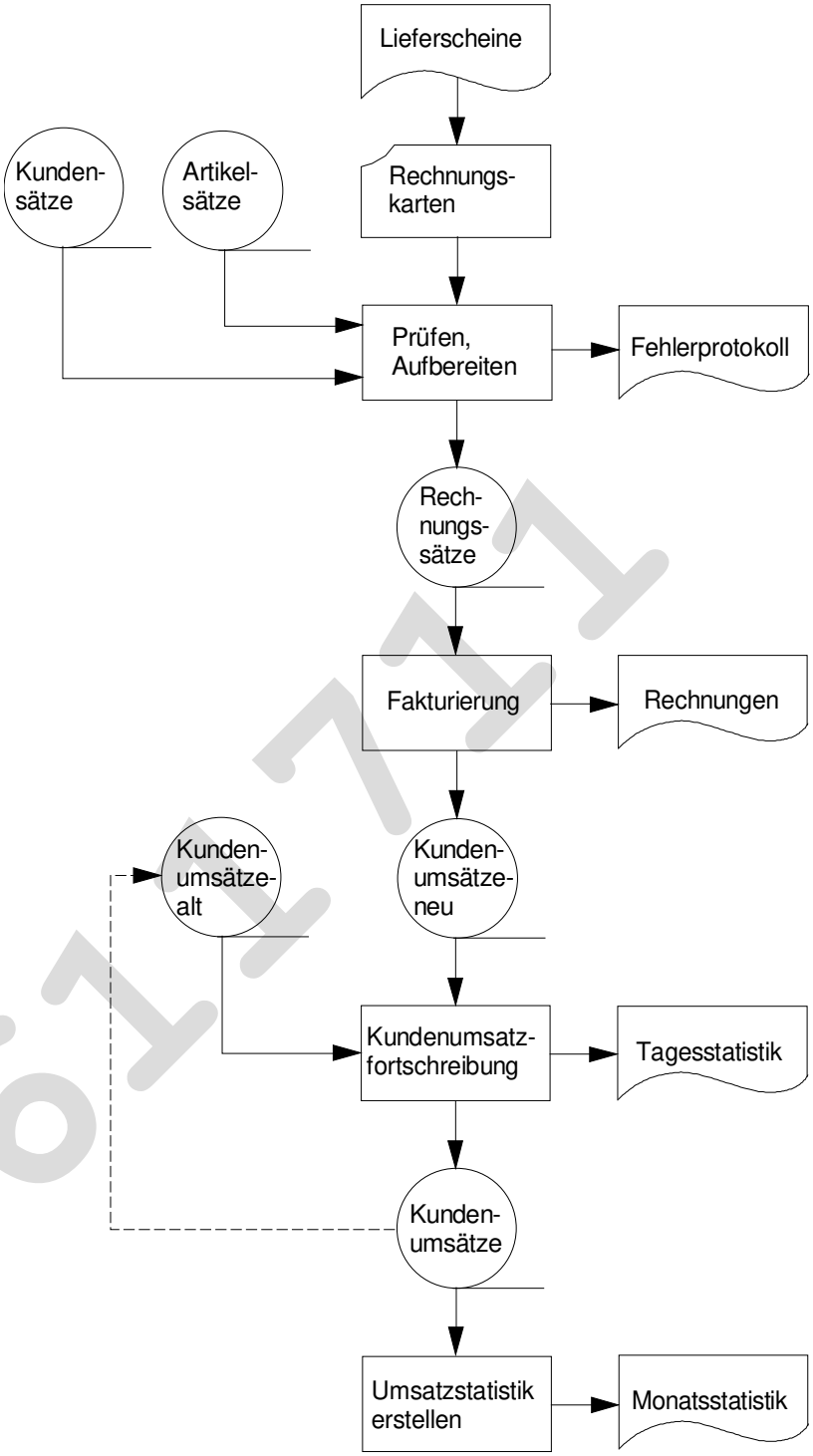


Abb. 1.3. Fakturierung und Umsatzfortschreibung.

Bei dem in Abb. 1.3 gezeigten Arbeitsablauf geht es um die Erfassung von Lieferscheinen, das Schreiben von Rechnungen und das Erstellen von Kundenumsatzstatistiken. Die in Rechnungskarten erfassten Lieferscheindaten werden unter Einbezug von Kunden- und Artikeldaten zu Rechnungssätzen zusammengestellt. Festgestellte Datenfehler hält ein Fehlerprotokoll fest. Ausgehend von den nach Kunden und innerhalb von Kunden nach Artikeln sortierten Rechnungssätzen können nun Rechnungen geschrieben und zugleich die Kundenumsätze festgehalten werden. Die bei täglicher Rechnungsschreibung pro Tag anfallenden Umsatzdaten werden jeweils in einer Tagesstatistik ausgegeben so-

wie zu Monatsumsätzen kumuliert. Die monatsbezogenen Umsatzdaten gehen in eine monatliche Kundenumsatzstatistik ein.

Übungsaufgabe 1.1

Betrachtet werde der in Abb. 1.3 dargestellte Arbeitsablauf. Beschreiben Sie den Vorgang der Kumulierung der Umsatzdaten etwas genauer und geben Sie an, wie viele Magnetbandlaufwerke bzw. Dateien an diesem Arbeitsschritt beteiligt sind.

Als Nachteile der Dateiverwaltung durch Anwendungssysteme können gelten:

- Redundanz, d.h. mehrfache Speicherung gleicher Daten in verschiedenen Dateien, da verschiedene Programme häufig gleiche Daten verarbeiten.
- Inkonsistenz, d.h. inhaltliche Widersprüche zwischen den in den Dateien abgelegten Daten, da eine zentrale Kontrolle der Dateiinhalte nicht erfolgt.
- Physische Datenabhängigkeit, d.h. die Änderung einer Datei hat Änderungen in allen Programmen zur Folge, welche diese Datei benutzen.
- Inflexibilität, d.h. die Erweiterung oder Änderung einer bearbeiteten Informationsverarbeitungsaufgabe ist mit aufwendigen Programm- und Dateiänderungen verbunden.

Nachteile der Dateiverwaltung durch Anwendungsprogramme

Durch den Einsatz eines Dateiverwaltungssystems können einige dieser Nachteile zumindest gemildert werden.

Übungsaufgabe 1.2

Ein besonders unerwünschter Nachteil der behandelten Verarbeitungsform ist die physische Datenabhängigkeit. Begründen Sie, warum die Änderung einer Datei Änderungen in den benutzenden Programmen zur Folge hat und warum dieser Effekt besonders unerwünscht ist.

1.2 Dateiverwaltungssysteme

Dateiverwaltungssysteme beruhen auf einer Zentralisierung der Dateiverwaltung. Aus den Anwendungsprogrammen werden also die Programmteile, die der Dateiverwaltung dienen, ausgelagert und in ein zentral gehaltenes Dienstprogramm eingebracht. Dieses Dienstprogramm unterstützt den Zugriff von Anwendungsprogrammen zu Dateien. Im Falle der Verwendung von Direktzugriffspeichern sind neben seriellen auch direkte Dateizugriffe möglich. Das Arbeiten mit einem Dateiverwaltungssystem veranschaulicht die Abb. 1.4.

Einsatz eines Dateiverwaltungssystems

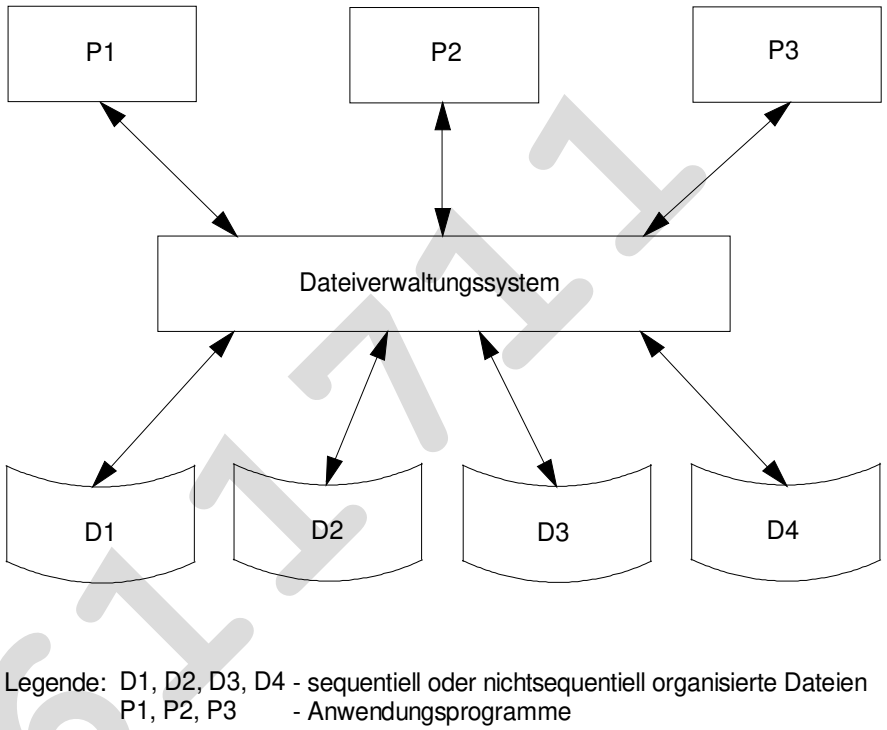


Abb. 1.4. Einsatz eines Dateiverwaltungssystems.

In der Regel ist ein Dateiverwaltungssystem auf nur eine Organisationsform von Dateien, z.B. die index-sequentielle Dateiorganisation, zugeschnitten. Verwendet ein Anwendungsprogramm Dateien unterschiedlicher Organisationsformen, so sind entsprechend mehrere Dateiverwaltungssysteme einzusetzen.

Folgende Merkmale charakterisieren das Arbeiten mit Dateiverwaltungssystemen:

Charakterisierung des Arbeitens mit Dateiverwaltungssystemen

- Mehrere Dateien werden mit einem Komplex von Dienstprogrammen verwaltet, welche - je nach Organisationsform der Dateien - serielle oder direkte Dateizugriffe unterstützen.
- Jedes Anwendungsprogramm benutzt eine oder mehrere dieser zentral verwalteten Dateien.
- Im Falle einer nichtseriellen Organisationsform einer Datei realisiert das Dateiverwaltungssystem direkte Dateizugriffe unter Verwendung von Hilfsorganisationen wie Indextabellen, B-Bäumen usw. (vgl. hierzu auch Kap. 5.2).
- Da die Datenerhebung, die Datenkontrolle und die Datenspeicherung für alle Dateien nun zentral abgewickelt werden können, lassen sich Inkonsistenzen vermeiden und übermäßige Redundanzen beseitigen.

- Die Direktzugriffe zu Daten ermöglichen Anwendungen, bei denen die Endbenutzer bestimmte Aufgaben interaktiv abwickeln, ohne dass die Dienste von Programmierern in Anspruch zu nehmen sind.

In den sechziger Jahren wurden Dateiverwaltungssysteme zunächst für sequentiell, index-sequentiell und gestreut gespeicherte Dateien entwickelt. Die für diese Organisationsformen verwendeten Zugriffsmethoden wurden häufig wie folgt bezeichnet:

spezielle Dateiverwaltungssysteme in den sechziger Jahren

SAM - Sequential Access Method,
ISAM - Indexed Sequential Access Method,
DAM - Direct Access Method.

Mit der Einführung des Konzepts der virtuellen Speicherung wurden die Zugriffsmethoden entsprechend erweitert. Beispielsweise bezeichnet VSAM (Virtual Storage Access Method) die Verbindung des virtuellen Speicherkonzepts mit der index-sequentiellen Zugriffsmethode. Zugriffsmethoden der genannten Art standen bei den kommerziell eingesetzten Rechnern schon bald als Dienstprogramme im Rahmen des Betriebssystems zur Verfügung und konnten somit von allen Anwendungsprogrammen genutzt werden.

Direktzugriffspeicher und Dateiverwaltungssysteme gestatten eine - gegenüber der sequentiellen Verarbeitung mit Magnetbändern - wesentlich elegantere Organisation von typischen betrieblichen Informationsverarbeitungsaufgaben. Beispielsweise lässt sich der in Abb. 1.3 veranschaulichte Arbeitsschritt der Kundenumsatzfortschreibung nun in der in Abb. 1.5 gezeigten Weise organisieren.

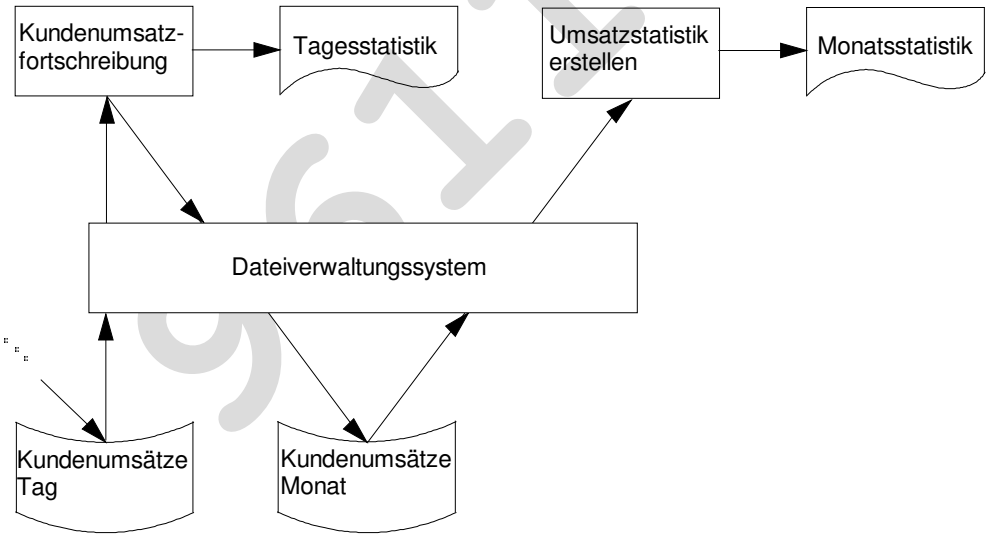


Abb. 1.5. Umsatzfortschreibung unter Verwendung eines Dateiverwaltungssystems mit Direktzugriffsspeichern.

Übungsaufgabe 1.3

Bei der Umsatzfortschreibung gemäß Abb. 1.5 werden die Tagesumsätze eines Monats kundenweise zu Monatsumsätzen kumuliert und am Ende eines Monats wird eine Monatsstatistik erstellt. Inwiefern lässt sich diese Informationsverarbeitungsaufgabe im Falle der Verwendung eines Dateiverwaltungssystems in Verbindung mit Direktzugriffen zu Daten eleganter bewältigen als im Falle der Verwendung von Magnetbandspeichern (vgl. hierzu Abb. 1.3)?

Gegenüber der separaten Datenverwaltung durch Anwendungsprogramme bewirkt der Einsatz von Dateiverwaltungssystemen zwar die aufgezeigten Vorteile, doch einige Nachteile bestehen weiterhin:

Nachteile von Dateiverwaltungssystemen

- Redundanzen können zwar gemildert werden, treten aber immer noch in einem erheblichen Umfang auf.
- Physische Datenabhängigkeiten betreffen gegebenenfalls mehr Anwendungsprogramme als im Fall der separaten Datenverwaltung.
- Inflexibilitäten sind nach wie vor zu verzeichnen, da die verwalteten Dateien nicht alle Anwendersichten in gleicher Weise unterstützen. Der Aufbau einer Datei ist für alle Benutzer gleich.

Übungsaufgabe 1.4

Begründen bzw. erläutern Sie kurz die eben genannten Nachteile bei der Verwendung von Dateiverwaltungssystemen.

1.3 Datenbankkonzept

anwendungsübergreifende zentrale Datenhaltung

Anders als die bisher angesprochenen Datenverwaltungsansätze baut das Datenbankkonzept auf einer anwendungsübergreifenden Denkweise auf, bei der zwischen Anwendungen einerseits und zentraler Datenhaltung für die Anwendungen andererseits unterschieden wird. An die Stelle der Einrichtung einzelner, auf spezielle Anwendungen zugeschnittener Dateien tritt nun die Modellierung der Datenwelt eines größeren Anwendungsbereichs oder gar eines ganzen Unternehmens. Die gemäß einem Datenmodell in einer Datenbank abgelegten und zentral verwalteten Daten stehen für alle Anwendungen zur Verfügung.

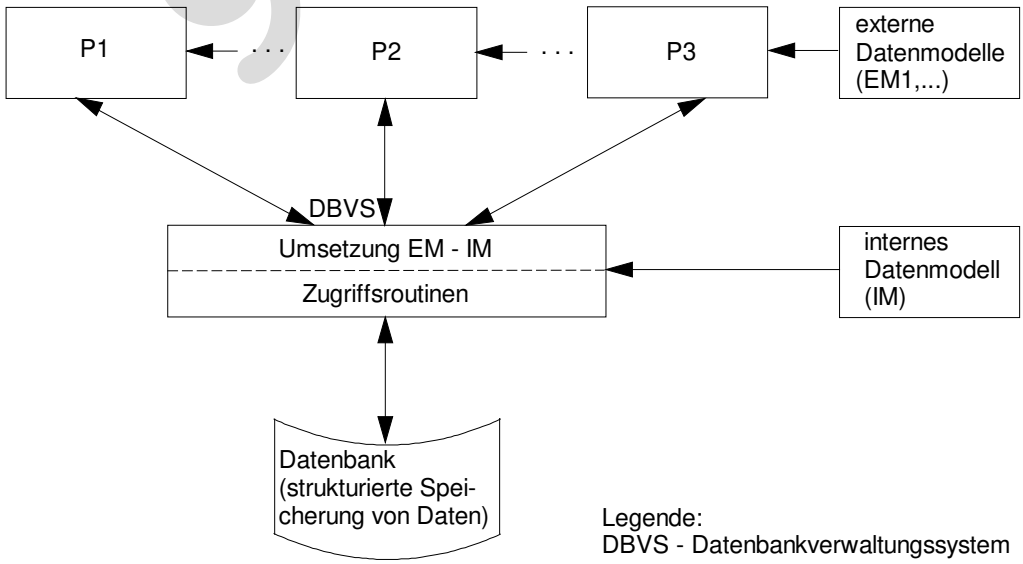
Vorteile des Datenbankkonzepts

Vorteile des Datenbankkonzepts sind die Reduktion von Redundanzen sowie die Elimination von physischen Datenabhängigkeiten und Inflexibilitäten. Redundanzen können zwar nicht vollständig beseitigt, jedoch auf ein Mindestmaß begrenzt werden, wie es aus Gründen der Verarbeitungseffizienz erforderlich ist. Inflexibilitäten werden durch die Berücksichtigung der Belange sämtlicher Benutzergruppen - man spricht in diesem Zusammenhang auch von Benutzersichten - bei der Datenmodellierung vermieden. Und der physischen Datenabhängigkeit begegnet man durch eine geschichtete Architektur von Datenbanksystemen. Das sogenannte "2-Schichten-Konzept" zeichnet sich durch physische Datenunabhängigkeit aus. Noch weitreichender ist das "3-Schichten-Konzept". Es zielt darüberhinaus auf logische Datenunabhängigkeit ab. Beide Schichtenkonzepte werden nachfolgend erläutert.

a) Physische Datenunabhängigkeit im 2-Schichten-Konzept

Trennung zwischen der logischen und der physischen Sicht auf Daten

Grundlage des 2-Schichten-Konzepts ist die strikte Trennung zwischen der Handhabung von Daten auf der logischen Ebene und ihrer physischen Speicherung und Manipulation. Die logische Betrachtungsebene bezeichnet man auch als externe Sicht und die physische Betrachtungsebene als interne Sicht. Entsprechend führt die Modellierung von Daten auf diesen Ebenen zu externen und internen Datenmodellen bzw. Datenschemata. Diesen Zusammenhang beschreibt die Abb. 1.6 in schematischer Weise.



2-Schichten-Konzept

Abb. 1.6. Schematische Darstellung des 2-Schichten-Konzepts.

In der Abb. 1.6 repräsentieren P1, P2 und P3 ganze Komplexe von zusammengehörigen Anwendungsprogrammen und damit auch entsprechende Benutzergruppen. Jede Benutzergruppe ist an einem bestimmten Ausschnitt aus der Datenbank interessiert. Die einzelnen Benutzersichten schlagen sich in unterschiedlichen externen Datenmodellen nieder. Letztere bilden eine Schicht, der im 2-Schichten-Konzept die durch das interne Datenmodell gebildete zweite Schicht gegenüber steht. Inhaltlich deckt das interne Datenmodell die durch alle Benutzersichten umrissene Datenwelt ab.

Funktionen eines Datenbankverwaltungs-systems

Das Datenbankverwaltungssystem (DBVS) stellt die Verbindung zwischen den in der Datenbank abgespeicherten Daten und den Anwendungsprogrammen her. Seine Verwaltungsfunktionen erstrecken sich auf das Speichern, Löschen, Ändern und Bereitstellen von Daten. Insbesondere setzt es die von den Anwendungsprogrammen angeforderten (logischen) Verwaltungsfunktionen in physische Speicheroperationen um. Diese Umsetzung ist mit einem Übergang aus der mit einem externen Modell definierten logischen Ebene der Datenbeschreibung in die durch das interne Modell gegebene physische Beschreibungsebene, der physischen Datenmanipulation mittels einer Zugriffsroutine und der Rücktransformation des Manipulationsergebnisses in die logische Beschreibungsebene verbunden. Die Umsetzungen zwischen der externen logischen und der internen physischen Ebene sind erforderlich, weil die Anwendungsprogramme lediglich logische Datenbeschreibungen enthalten und folglich keine Kenntnisse über physische Datenstrukturen besitzen.

Zusammenfassend lassen sich die Merkmale des 2-Schichten-Konzepts wie folgt darstellen:

Merkmale des 2-Schichten-Konzepts

- Es besteht eine strikte Trennung zwischen den Benutzersichten auf Daten und der physischen Datenspeicherung.
- Jede Benutzersicht repräsentiert eine logische Beschreibung der von einer Benutzergruppe benötigten Daten; diese wird auch als externes Datenmodell oder als externes (Daten-)Schema bezeichnet.
- In der Datenbank werden die von allen Benutzern benötigten Daten abgelegt; die physische Speicherstruktur wird durch das interne Modell - auch internes (Daten-)Schema genannt - festgelegt und ist den Benutzern unbekannt.
- Auf der Datenbank operiert ausschließlich das DBVS; es stellt mit Verwaltungsfunktionen wie Speichern, Löschen, Ändern usw. die Verbindung zwischen den externen Benutzersichten und den intern gespeicherten Daten her.

physische Datenunabhängigkeit

Bei dem 2-Schichten-Konzept entkoppelt das DBVS die externen Schemata und das interne Schema. Eine Folge dieser Entkopplung ist die Verwirklichung der physischen Datenunabhängigkeit: Änderungen der physischen Struktur der Daten, also Änderungen der Datenorganisation bzw. der Zugriffspfade, werden vom DBVS abgefangen und schlagen nicht auf die Anwendungsprogramme durch. Und zwar deshalb, weil die Anwendungsprogramme von der Beschreibung physischer Datenstrukturen bzw. dem internen Schema keine Kenntnis haben. Datenanforderungen werden in den Anwendungsprogrammen vielmehr ausschließlich auf der logischen Ebene, d.h. auf der Ebene externer Schemata, formuliert.

Übungsaufgabe 1.5

Bekanntlich stellt das DBVS in einem Datenbanksystem die Verbindung zwischen den Anwendungsprogrammen und der Datenbank her. Möchte ein Anwendungsprogramm auf ein bestimmtes Datenobjekt in der Datenbank zugreifen, so ist dies mit einer Transformation aus dem entsprechenden externen Schema in das interne Schema und einer Rücktransformation verbunden. Erläutern Sie kurz, was man sich unter diesen Schemaübergängen vorzustellen hat.

b) Logische Datenunabhängigkeit im 3-Schichten-Konzept

konzeptionelles Datenmodell als 3. Schicht

Kommen beim 2-Schichten-Konzept zu den bestehenden Anwendungen neue Anwendungsprogramme hinzu, deren Datenanforderungen durch die existierenden externen Datenmodelle nicht abgedeckt werden, so sind Änderungen bzw. Erweiterungen der Datenmodelle unumgänglich. Diese Änderungen können ihrerseits Änderungen in den bestehenden Anwendungsprogrammen zur Folge haben. Solche unerwünschten Folgewirkungen werden im 3-Schichten-Konzept durch das sogenannte konzeptionelle Datenmodell vermieden.

Abgrenzung der drei Schichten

Im 3-Schichten-Modell tritt das konzeptionelle Modell als dritte Schicht zwischen die Schicht der externen Modelle und das interne Modell. Das konzeptionelle Modell stellt ein von allen Benutzergruppen akzeptiertes Datenmodell eines Ausschnitts aus der realen Welt dar. Der zugrundeliegende Realitätsausschnitt kann einen größeren Anwendungsbereich oder ein ganzes Unternehmen umfassen. Sämtliche Anwendungsprogramme der Benutzergruppen beziehen sich auf diesen Realitätsausschnitt. Folglich stellen die externen Datenmodelle der Benutzergruppen, also die Benutzersichten, Ausschnitte aus dem konzeptionellen Modell dar. Jede Benutzersicht schließt nur den Teil des als logische Gesamtsicht zu begreifenden konzeptionellen Modells ein, der für die jeweilige Benutzergruppe von Interesse ist. Eine schematische Darstellung des 3-Schichten-Modells zeigt die Abb. 1.7.

logische Gesamtsicht

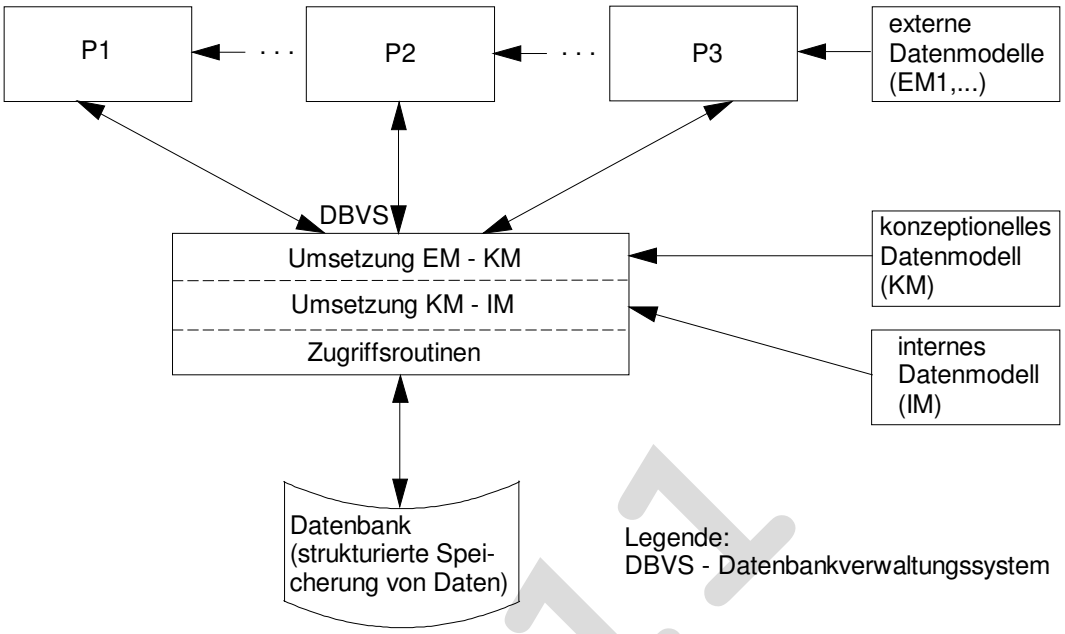


Abb. 1.7. Schematische Darstellung des 3-Schichten-Konzepts.

Umsetzungen im 3-Schichten-Konzept

Da bei dem 3-Schichten-Konzept das konzeptionelle Modell als Verbindungsglied zwischen der externen logischen und der internen physischen Schicht fungiert, ist bei der Anforderung von Datenbankfunktionen durch Anwendungsprogramme eine zweistufige Umsetzung erforderlich. Erstens zwischen der Ebene der externen Modelle und dem konzeptionellen Modell und zweitens zwischen dem konzeptionellen und dem internen Modell. Abgesehen von diesen zweistufigen Transformationen gelten die Ausführungen zur Funktionsweise des DBVS für den Fall des 2-Schichten-Konzepts hier in analoger Weise.

logische Datenunabhängigkeit im 3-Schichten-Konzept

Gegenüber dem 2-Schichten-Konzept zeichnet sich das 3-Schichten-Konzept durch die Eigenschaft der logischen Datenunabhängigkeit aus. Die logische Datenunabhängigkeit hängt mit der Funktion des konzeptionellen Modells als eine alle Einzelsichten überdeckende logische Gesamtsicht zusammen. Sie besagt etwa folgendes: Eine durch neu hinzukommende Anwendungen bedingte Weiterentwicklung bzw. Erweiterung des konzeptionellen Modells wirkt sich nicht auf die bestehenden Anwendungsprogramme aus. Wird also die logische Gesamtsicht erweitert, so bleiben die (bisherigen) Teilsichten stabil. Die bestehenden Anwendungsprogramme müssen somit nicht geändert werden.

Übungsaufgabe 1.6

Begründen Sie kurz, warum sich bei dem 3-Schichten-Konzept neu hinzukommende Anwendungen nicht auf bestehende Anwendungsprogramme auswirken.

Zu Beginn dieses Kapitels wurden einige Vorteile des Datenbankkonzepts bereits genannt. Zusammen mit einigen weiteren Vorteilen seien die Vorzüge des Datenbankkonzepts hier zusammengefasst. Sie lauten:

Vorteile des Datenbankkonzepts

- Geringe Redundanz: Daten werden nur noch in dem Umfang mehrfach gespeichert, der aus Effizienzgründen unbedingt erforderlich ist.
- Hohe Datenkonsistenz: Die zentrale Verwaltung der Daten ermöglicht eine wirksame Überprüfung der Korrektheit der Daten.
- Flexible Nutzung der Daten: Auswertungen können sich, je nach den Benutzerbedürfnissen, sowohl auf kleine Teile der Datenbank als auch auf den gesamten Datenbestand beziehen.
- Unterschiedliche Benutzungsformen: Benutzer können Programmierer sein, die komplexe datenbankgestützte Anwendungsprogramme entwickeln, aber auch Nichtprogrammierer, die lediglich Datenbankabfragen abwickeln.
- Verbesserter Datenschutz und verbesserte Datensicherung: Datenschutz- und Datensicherungsmaßnahmen lassen sich wegen der zentralen Datenhaltung auf effizientere Weise organisieren.

Hinzuweisen ist auch auf einige Nachteile des Datenbankkonzepts. Sie resultieren meist aus der Komplexität von Datenbanksystemen. Nachteile sind u.a.:

Nachteile des Datenbankkonzepts

- Höhere Qualifikationsanforderungen: Die Einrichtung und der Betrieb von komplexen Datenbanksystemen erfordern höhere Fachkompetenz als die dateiorientierte Informationsverarbeitung.
- Neue Abhängigkeiten: Die einzelnen Datenbankbenutzer sind von der zentralen, die Datenbank betreibenden Instanz abhängig, da sie sich an zentral vorgegebene Standards bezüglich Sprachen, Geräteschnittstellen usw. anzupassen haben.
- Geringere Verarbeitungseffizienz: Die höhere Auswertungsflexibilität wird durch Leistungseinbußen bei der Verarbeitung erkauft, die sich vor allem bei nichtstandardisierten interaktiven Anwendungsprogrammen bemerkbar machen kann.
- Höherer Systemaufwand: Für umfangreichere und komplexere Systeme wie sie Datenbanksysteme darstellen, ist ein höherer Hardwareaufwand (leistungsfähigere Plattenspeicher, größere zentrale Prozessorleistung, leistungsfähigere Datenkanäle und Kommunikationseinrichtungen usw.) und Softwareaufwand (leistungsfähigeres Betriebssystem) in Kauf zu nehmen.
- Totalrisiken: Falls die ergriffenen Datenschutz- oder Datensicherungsmaßnahmen versagen, kann jeweils die gesamte Datenbasis betroffen sein.

Freilich wiegen die Vorteile des Datenbankkonzepts seine Nachteile auf. In der betrieblichen Praxis spielt die datenbankgestützte Informationsverarbeitung längst eine zentrale Rolle.

2 Architektur von Datenbanksystemen

Komponenten eines Datenbanksystems

Datenbank

Datenbankverwaltungssystem

Unter einem Datenbanksystem wird hier ein Gesamtsystem verstanden, das aus zwei Komponenten besteht, einer Datenbank und einem auf der Datenbank operierenden Datenbankverwaltungssystem (DBVS). Mit dem Begriff "Datenbank" umschreibt man die Gesamtheit der für einen größeren Anwendungsbereich zentral gespeicherten und verwalteten Daten einschließlich der Zugriffspfade zu den Daten; die Dateninhalte und die Strukturierung der Daten ergeben sich aus einem für den Anwendungsbereich erstellten Datenmodell. Wie in Kap. 1 bereits erwähnt wurde, bezeichnet man mit dem Begriff "Datenbankverwaltungssystem" ein Softwaresystem zum Speichern, Löschen, Ändern, Bereitstellen usw. von Daten einer Datenbank. Ein DBVS erfüllt darüberhinaus noch weitere Zwecke, etwa die Durchführung von Zugriffskontrollen und Integritätsprüfungen sowie die Synchronisation von Datenbankoperationen. An dieser Stelle mögen diese Begriffserläuterungen genügen. Begriffliche Präzisierungen und Erläuterungen zu Begriffen wie "Datenmodell", "Integrität", "Synchronisation" usw. folgen noch.

Inhalt des Kapitels

Grundüberlegungen zur Architektur von Datenbanksystemen finden sich bereits in Kap. 1 mit dem dort behandelten 2-Schichten- bzw. 3-Schichten-Konzept und den Anmerkungen zur physischen und logischen Datenunabhängigkeit. Im vorliegenden Kapitel werden diese Überlegungen weitergeführt und ausgeweitet. Die Unterscheidung von Datenbankebenen in Kap. 2.1 ist stärker auf die Komponente "Datenbank" eines Datenbanksystems ausgerichtet. Bei den in Kap. 2.2 behandelten Datenbankverwaltungssystemen stehen dagegen funktionelle Aspekte und die prinzipielle Realisierung des Datenbankkonzepts im Vordergrund.

2.1 Datenbankebenen

Datenbank als abstrakter Speicher

Auf der externen, dem Benutzer zugekehrten Ebene präsentiert sich eine Datenbank als ein abstrakter Speicher. Daten werden auf dieser Ebene nicht durch ihre Organisation im physisch vorhandenen Speicher beschrieben, sondern auf einer höheren, logischen Ebene - der Ebene eines abstrakten Speichers. Anwenderprogramme und Benutzerabfragen greifen auf diesen abstrakten Speicher zu und nicht auf den physischen Speicher. Entsprechend wird der Datenzugriff nicht mittels Adressen spezifiziert, sondern durch die (logische) Beschreibung von (Ziel-)Daten.

DBVS-Schnittstelle

Realisiert wird ein abstrakter Speicher durch ein DBVS. Ein abstrakter Speicher stellt gleichsam eine DBVS-Schnittstelle dar, die definiert ist durch

- die Struktur der an dieser Schnittstelle übergebenen Datenobjekte und
- die auf diesen Datenstrukturen ausführbaren datenbankbezogenen Operationen bzw. Datenbankfunktionen.

Die Umsetzung von logischen Datenstrukturen und darauf definierten (logischen) Datenbankfunktionen in physische Datenstrukturen und deren physische Manipulation in der Datenbank übernimmt das DBVS. Wie aus dem 2-Schichten- bzw. 3-Schichten-Konzept bekannt ist, erfolgt diese Umsetzung abgestuft und über mehrere Datenbankebenen hin-

weg. Auf diese Weise ist es insbesondere möglich, den Anspruch auf physische und logische Datenunabhängigkeit zu erfüllen.

In Kap. 1.3 wurden die drei Ebenen des 3-Schichten-Konzepts implizit genannt und lediglich kurz erläutert. Nicht erwähnt wurde, dass das 3-Schichten-Konzept ein Bestandteil des sogenannten ANSI-Architekturmodells ist. Dieses Modell sieht eine 3-Ebenen-Architektur und die Definition von sogenannten Schemata auf den drei Ebenen vor. Diese Schemata sind das externe, das konzeptionelle und das interne Schema. In Abb. 1.7 wurden die Schemata als Datenmodelle bezeichnet. Im Folgenden werden das ANSI-Architekturmodell und die genannten Schemata behandelt. Dabei finden auch sprachliche und organisatorische Aspekte, die das ANSI-Architekturmodell ebenfalls einschließt, Berücksichtigung.

a) ANSI-Architekturmodell

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.

Wesentliche Merkmale des auch heute noch aktuellen ANSI-Architekturmodells sind:

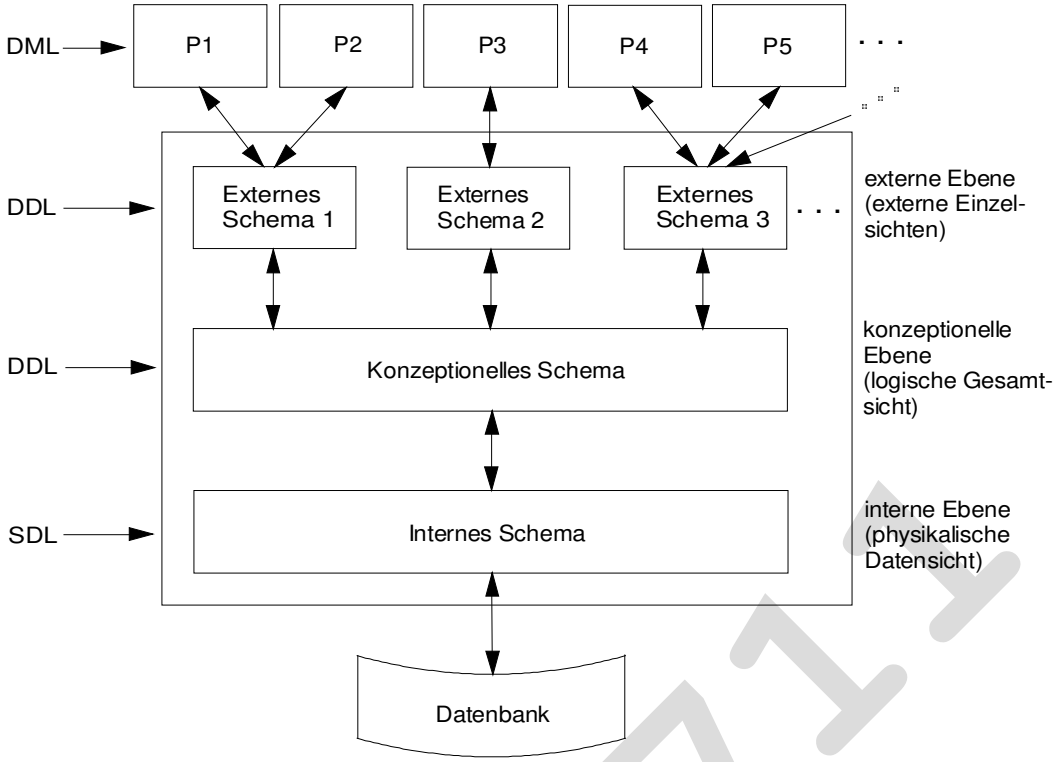
- Die Verwirklichung der physischen und der logischen Datenunabhängigkeit mittels einer Ebenenhierarchie, welche die externe, die konzeptionelle und die interne Ebene bzw. Sicht einschließt.

Merkmale des ANSI-Architekturmodells

- Die Beschreibung von Daten auf den drei Ebenen in Form von sogenannten Schemata mit Hilfe von speziellen Sprachen, nämlich Datenbeschreibungssprachen zur Beschreibung externer und konzeptioneller Schemata sowie Speicherbeschreibungssprachen zur Definition interner Schemata.
- Die ebenenweise Zuordnung von personellen Instanzen, die als Anwendungs-, Unternehmens- und Datenbankadministrator bezeichnet werden und für die externe, konzeptionelle und interne Ebene zuständig sind.

Von herausragender Bedeutung sind die ebenenbezogenen Schemata. Ihre Einbettung zwischen Anwendungsprogrammen und Datenbank und ihr Bezug zu den genannten Sprachen wird in Abb. 2.1 in schematischer Form veranschaulicht.

Laut Abb. 2.1 beschreibt je ein externes Schema den Ausschnitt aus der logischen Gesamtsicht, der für je eine Gruppe zusammengehöriger Anwendungsprogramme von Interesse ist. Der Zugriff eines Anwendungsprogramms auf die Datenbank durchsetzt alle drei in Abb. 2.1 ausgewiesenen Ebenen. Die hierbei vorzunehmenden Umsetzungen zwischen den Schemata werden in Kap. 2.2 behandelt. An dieser Stelle stehen die Schemata selbst im Vordergrund.



Datenbankebenen im ANSI-Architekturmodell

Legende:
DML - Datenmanipulationssprache (engl. data manipulation language)
DDL - Datenbeschreibungssprache (engl. data description language)
SDL - Speicherbeschreibungssprache (engl. storage description language)

Abb. 2.1. Datenbankebenen im ANSI-Architekturmodell.

Übungsaufgabe 2.1

Welche Bedeutung kommt der Empfehlung des nationalen Normenausschusses der USA für ein Architekturmodell für Datenbanksysteme zu?

b) Konzeptionelles Schema

Begriff des konzeptionellen Schemas

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 (vgl. hierzu Kap. 3). Keinesfalls enthält ein konzeptionelles Schema Angaben zur physischen Organisation von Datenstrukturen. Angaben dieser Art bleiben dem internen Schema vorbehalten.

Ein Sprachmittel zur Beschreibung der konzeptionellen Datensicht nennt man Datenbeschreibungssprache (DDL). Der Sprachumfang einer DDL ergibt sich aus den Beschreibungszwecken. Zu diesen gehören u.a.:

Aufgaben einer Datenbeschreibungssprache (DDL)

- Benennung von logischen Dateneinheiten,
- Benennung und Charakterisierung der Beziehungen zwischen Dateneinheiten,
- Definition der Typen, der Wertebereiche und gegebenenfalls der Schlüsseleigenschaften von logischen Dateneinheiten,
- Definition von Zugriffsrechten für Dateneinheiten,
- Definition von Integritätsbedingungen für Dateneinheiten.

Der weiteren Erläuterung möge ein einfaches Beispiel für ein konzeptionelles Schema dienen.

Beispiel 2.1

Aus Vereinfachungsgründen werde hier ein Realitätsauschnitt gewählt, der lediglich aus dem Objekt *Mitarbeiter* einer bestimmten Firma besteht. Entsprechend reduziert sich das konzeptionelle Schema für die Firma wie folgt:

Beispiel eines konzeptionellen Schemas

```
SCHEMA FIRMA;
    RECORD TYPE MITARBEITER;
        DATA ITEM mitarbeiternr, INTEGER, KEY,
                ASSERT mitarbeiternr > 0;
        DATA ITEM name, CHARACTER;
        DATA ITEM adresse, ALPHANUMERIC;
        DATA ITEM alter, INTEGER, UNIT IS JAHRE,
                ASSERT alter IS 16 THRU 65;
        DATA ITEM abteilung, CHARACTER;
        DATA ITEM beruf, CHARACTER;
        DATA ITEM gehalt, REAL, UNIT IS DM,
                ASSERT gehalt IS 0 THRU 80000;
    END RECORD TYPE;
END SCHEMA.
```

In dem Beispiel werden DDL-Schlüsselwörter groß und Bezeichner für Dateneinheiten klein geschrieben. Das konzeptionelle Schema für die Firma umfasst hier nur den Satztyp MITARBEITER. Dieser Satztyp setzt sich aus den Dateneinheiten *mitarbeiternr*, *name*, *adresse* usw. zusammen. Die Dateneinheiten gehören den Typen INTEGER, REAL, CHARACTER oder ALPHANUMERIC an. Als Schlüssel fungiert, wie das Schlüssel-

wort KEY anzeigt, die Dateneinheit *mitarbeiternr*. Falls erforderlich, kann bei numerischen Dateneinheiten eine Maßeinheit angegeben werden, z.B. DM für die Dateneinheit *gehalt*.
Mit dem Schlüsselwort ASSERT wird die Definition von Integritätsbedingungen eingeleitet. Die Integritätsbedingung

ASSERT alter IS 16 THRU 65

fordert beispielsweise, dass die Dateneinheit *alter* nur Werte im Bereich von 16 bis 65 annehmen darf.

Ausdrücklich sei darauf hingewiesen, dass das gezeigte konzeptionelle Schema keinerlei Angaben zur physischen Datendarstellung enthält.

In umfänglicheren konzeptionellen Schemata werden zusätzlich auch die Beziehungen zwischen Datenobjekten modelliert.

Unternehmensadministrator

Nach dem ANSI-Vorschlag ist eine eigenständige personelle Instanz, genannt Unternehmensadministrator (engl. enterprise administrator), für das konzeptionelle Schema vorzusehen. Der Unternehmensadministrator ist für die Erstellung und die Pflege des konzeptionellen Schemas eines Unternehmens zuständig. Als übergeordnete Instanz vertritt er nicht einzelne Anwenderinteressen. Vielmehr hat er der Gesamtheit der Informationsverarbeitungsaufgaben in einem Unternehmen Rechnung zu tragen. Dies schließt insbesondere auch die sorgfältige Beobachtung der Unternehmensentwicklung und die Berücksichtigung der sich daraus ergebenden Konsequenzen für die betriebliche Informationsverarbeitung ein.

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

Vorteile des konzeptionellen Schemas

- 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.

Hingewiesen sei darauf, dass das konzeptionelle Schema eines Unternehmens keine Dateninhalte enthält und somit auch keine Datenbank darstellt. Es beschreibt lediglich die Datenwelt eines Unternehmens in anwendungsübergreifender Form.

Übungsaufgabe 2.2

Entwickeln Sie in Analogie zu dem in Beispiel 2.1 vorgestellten konzeptionellen Schema ein konzeptionelles Schema für einen Realitätsausschnitt, der lediglich aus dem Objekt *Artikel* besteht. Gehen Sie dabei von einer hypothetischen Firma Ihrer Wahl aus. Versuchen Sie, zumindest eine Integritätsbedingung in das Schema einzubeziehen.

Übungsaufgabe 2.3

Geben Sie zwei Beispiele für Unternehmensentwicklungen an, die sich auf das konzeptionelle Schema eines Unternehmens auswirken. Nennen und begründen Sie die Auswirkungen.

c) Internes Schema

Begriff des internen Schemas

Als internes Schema bezeichnet man die Beschreibung der physischen Organisation der in einem konzeptionellen Schema definierten logischen Datenstrukturen. Ein internes Schema legt also die physische 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 sollten vor allem die Zugriffshäufigkeit zu den Daten und das Zeitverhalten der Anwendungen ermittelt werden, da sie Rückschlüsse auf geeignete Speicherungsformen und Zugriffspfade zulassen.

Zur Beschreibung der internen Datensicht sieht das ANSI-Architekturmodell das Sprachmittel der Speicherbeschreibungssprache (SDL) vor. Eine SDL umfasst sprachliche Komponenten für u.a. folgende Zwecke:

Aufgaben einer Speicherbeschreibungssprache (SDL)

- Auswahl und Benennung des konkreten Speichermediums bzw. Geräts,
- Beschreibung der physischen Datendarstellung, d.h. des Typs und der Länge von Dateneinheiten,
- Festlegung von Zugriffspfaden, d.h. von Hilfsorganisationen wie Indizes, Zeigern usw.,
- Festlegung des Aufbaus und der Lage von Pufferbereichen im Arbeitsspeicher für Zwecke wie Datensuche und Datentransfer.

Beispiel 2.2

Betrachtet werde das in Beispiel 2.1 vorgestellte konzeptionelle Schema. Ein internes Schema für diese konzeptionelle Datensicht könnte beispielsweise wie folgt aussehen:

Beispiel eines internen Schemas

```
STORAGE MODULE FOR SCHEMA FIRMA;
    MITARBEITER, CONTINUOUS FILE, FILE NAME IS mitarbdatei,
        MEDIUM IS DISK, DEVICE IS IBM 3330;
        mitarbeiternr, FIXED DECIMAL (3,0), STORAGE KEY,
                SORT ASCENDING;
        name, STRING (20);
        adresse, STRING (30);
        alter, BIT(8), INDEX USING B-TREE;
        gehalt, FIXED DECIMAL (6,2), INDEX USING POINTER ARRAY;
        abteilung, STRING (12);
        beruf, STRING (8), INDEX USING B-TREE;
    END mitarbdatei;
END STORAGE MODULE.
```

Auch in diesem Beispiel werden Schlüsselwörter groß und Bezeichner für Dateneinheiten klein geschrieben. Zu dem Beispiel sei folgendes angemerkt:

Es wird eine Datei mit dem Namen *mitarbdatei* vereinbart und zwar mit der Speicherungsform CONTINUOUS FILE. Die Sätze der Datei werden also sequentiell abgelegt. Als Speichermedium dient ein Plattenspeicher IBM 3330.

Zwei Dateneinheiten werden als Festpunktzahlen vereinbart, nämlich *mitarbeiternr* mit 3 Stellen vor und Null Stellen nach dem Dezimalpunkt sowie *gehalt* mit einer Länge von 6 Stellen, davon 2 Stellen nach dem Dezimalpunkt. Bis auf die Dateneinheit *alter* werden die übrigen Dateneinheiten als Zeichenketten unterschiedlicher Länge definiert. Für die Dateneinheit *alter* ist schließlich eine Codierung als 8-stelliges Bitmuster vorgesehen.

Primärschlüssel ist, wie die Angabe "STORAGE KEY" erkennen lässt, die Dateneinheit *mitarbeiternr*. Die Sätze von *mitarbeiterdatei* werden sortiert nach aufsteigendem Primärschlüssel abgelegt; dies drückt die Angabe "SORT ASCENDING" aus.

Für einige Dateneinheiten sind spezielle Zugriffspfade vorgesehen; diese Dateneinheiten stellen also Sekundärschlüssel dar. Je eine Hilfsorganisation in Form eines B-Baumes wird für die Dateneinheiten *alter* und *beruf* vorgegeben. Ein Zeigerfeld soll dagegen einen schnellen Zugriff für den Sekundärschlüssel *gehalt* als Zugriffskriterium ermöglichen.

Datenbank-administrator

Als eigenständige personelle Instanz, die in einem Unternehmen für die Erstellung und die Pflege des internen Schemas zuständig ist, sieht der ANSI-Vorschlag den sogenannten Datenbankadministrator (engl. data base administrator) vor. Dem Datenbankadministrator obliegt es, das vom Unternehmensadministrator vorgegebene konzeptionelle Schema in das interne Schema umzusetzen. Angedeutet wurde bereits, dass hierbei anwendungsbezogene Aspekte zu berücksichtigen sind. Beispielsweise die sich aus den einzelnen Anwendungen ergebenden Mengengerüste, die Zugriffshäufigkeiten zu den Daten, die erwünschten Systemantwortzeiten bei interaktiven Anwendungen usw.

Übungsaufgabe 2.4

Betrachtet werde das konzeptionelle Schema, das als Lösung zu der Übungsaufgabe 2.2 am Ende dieser Kurseinheit angegeben wird. Entwickeln Sie für diese konzeptionelle Datensicht ein internes Schema. Orientieren Sie sich hierbei an dem in Beispiel 2.2 angegebenen internen Schema. Sehen Sie neben dem Primärschlüssel zumindest einen Sekundärschlüssel vor. Wählen Sie einfachheitshalber die in Beispiel 2.2 verwendete Speicherungsform der Datensätze und geben Sie für die einzelnen Dateneinheiten geeignete Typen und Längen vor.

Übungsaufgabe 2.5

In Beispiel 2.2 wird u.a. die Dateneinheit *gehalt* als Sekundärschlüssel vereinbart. Geben Sie einen Anwendungszusammenhang an, in dem der Sekundärschlüssel *gehalt* vorteilhaft genutzt werden kann.

d) Externes Schema

Begriff des externen Schemas

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.

Sprachmittel zur Beschreibung externer Schemata

Da externe Schemata Ausschnitte aus konzeptionellen Schemata darstellen, eignen sich zu ihrer Beschreibung grundsätzlich die Sprachmittel, die man für konzeptionelle Schemata verwendet. Sprachen zur Formulierung konzeptioneller und damit auch externer Datensichten bezeichnet man bekanntlich als Datenbeschreibungssprachen (DDL).

Externe Schemata stehen in einem unmittelbaren Bezug zu Anwendungsprogrammen. Viele konkrete Datenbanksysteme sehen daher ihre Beschreibung in einem bestimmten Bereich der Anwendungsprogramme vor. Beispielsweise in der sogenannten SCHEMA-SECTION der DATA-DIVISION von COBOL-Anwendungsprogrammen.

An einem Beispiel sei das Wesen externer Schemata nun weiter verdeutlicht.

Beispiel 2.3

Zugrundegelegt werde das in Beispiel 2.1 formulierte konzeptionelle Schema. Nun möge eine Benutzergruppe dadurch charakterisiert sein, dass ihre Mitglieder ausschließlich an den Adressen der Mitarbeiter des Unternehmens interessiert sind und ihre Informationswünsche durch Datenbankabfragen befriedigen. Das Subschema für diese Benutzergruppe könnte beispielsweise wie folgt aussehen:

Beispiel eines externen Schemas

```
SUBSCHEMA ADRESSBUCH OF SCHEMA FIRMA;
    RECORD TYPE ADRESSE OF RECORD TYPE MITARBEITER;
        ACCESS IS RETRIEVAL ONLY;
        DATA ITEM mitarbeiternr, KEY, CHARACTER (3);
        DATA ITEM name, CHARACTER (20);
        DATA ITEM adresse, ALPHANUMERIC (30);
        DATA ITEM beruf, CHARACTER (8);
        DATA ITEM abteilung, CHARACTER (12);
    END RECORD TYPE;
END SUBSCHEMA.
```

Das definierte Subschema ADRESSBUCH besteht aus dem Satztyp ADRESSE. Dieser Satztyp beinhaltet einen Ausschnitt aus dem konzeptionellen Schema bzw. dem Satztyp MITARBEITER. In ADRESSBUCH bleiben die Dateneinheiten verborgen, die nicht üblicher Bestandteil einer Mitarbeiteradresse sind - also *alter* und vor allem *gehalt*.

Mit der Angabe

ACCESS IS RETRIEVAL ONLY

werden die Manipulationsrechte der Benutzergruppe festgelegt. Gestattet ist nur das Lesen von Adressen, nicht aber das Eingeben, Ändern oder Löschen von Adressen.

Programmpuffer

Neben dem verwendeten Subschema muss in einem Anwendungsprogramm auch ein Pufferbereich, der sogenannte Programmpuffer, vereinbart werden. Der Programmpuffer stellt die Schnittstelle des Anwendungsprogramms zum Datenbanksystem dar. Über ihn wird der Datentransfer abgewickelt:

- Das Anwendungsprogramm kann auf Daten im Puffer zugreifen, nachdem diese der Datenbank entnommenen und im Programmpuffer bereitgestellt wurden.
- In der Datenbank abzuspeichernde Daten kann das Anwendungsprogramm in den Programmpuffer schreiben und so an das Datenbanksystem übergeben.

Auf die Vereinbarung eines Programmpuffers geht das folgende Beispiel ein.

Beispiel 2.4

Betrachtet werde ein nicht näher spezifiziertes Anwendungsprogramm, welches das in Beispiel 2.3 angegebene Subschema verwendet. In diesem Anwendungsprogramm soll nun ein bestimmter Bereich des Arbeitsspeichers als Programmpuffer vereinbart werden. Handelt es sich um ein PL/I-Anwendungsprogramm, so könnte die Puffervereinbarung beispielsweise wie folgt lauten:

Beispiel eines Programmpuffers in PL/I

```
DCL  1 ADRESSE,
       2 MITARBEITERNR, CHARACTER (3),
       2 NAME CHARACTER (20),
       2 ADRESSE ALPHANUMERIC (30),
       2 BERUF CHARACTER (8),
       2 ABTEILUNG CHARACTER (12);
```

Auch die Beschreibung des Programmpuffers findet auf der logischen Ebene statt. Die physische Darstellung der Daten, wie sie im internen Schema festgelegt wurde, bleibt hier unberücksichtigt.

Datenmanipulationssprache (DML)

Zur Formulierung von Datenbankzugriffen im Rahmen von Anwendungsprogrammen werden spezielle Sprachen verwendet, die sogenannten Datenmanipulationssprachen (DML). Häufig ist eine DML in eine andere Sprache, beispielsweise COBOL oder PL/I, eingebettet. Diese andere Sprache bezeichnet man dann als Gastsprache (engl. host language). Eine DML erweitert den Sprachumfang einer Gastsprache um Anweisungen zur Datenbankmanipulation, d.h. zum Lesen, Ändern, Löschen usw. von Daten der Datenbank. Die Einbettung von DML-Anweisungen in ein Anwendungsprogramm möge folgendes triviale Beispiel verdeutlichen.

Beispiel 2.5

Einbettung von DML-Anweisungen in ein Anwendungsprogramm

```
                    .
                    .
                    .
LIES ARTIKELSATZ, ARTIKELNR = 0418
ERHÖHE ARTIKELBESTAND UM 250
UPDATE ARTIKELSATZ
                    .
                    .
                    .
```

Die erste und die dritte Anweisung sind in einer hypothetischen DML formuliert, die zweite dagegen in einer hypothetischen Gastsprache. Die erste Anweisung bewirkt einen Zugriff auf einen bestimmten Artikelsatz. Der Satz wird mit der zweiten Anweisung aktualisiert und mit der dritten in die Datenbank zurückgeschrieben.

Anwendungsadministrator

Auch für die externen Schemata sieht der ANSI-Vorschlag eine spezielle personelle Instanz vor, den Anwendungsadministrator (engl. application administrator). Er ist für die Entwicklung und die Pflege der Subschemata der Benutzergruppen eines Unternehmens zuständig.

In der Praxis wird der Vorschlag ebenenbezogener Instanzen nur bedingt umgesetzt. Man begnügt sich in der Regel mit einer Instanz, dem Datenbankadministrator. Er vertritt die konzeptionelle und die interne Sicht und unterstützt die Benutzergruppen bei der Entwicklung externer Schemata.

Übungsaufgabe 2.6

Zugrundegelegt werde das in Beispiel 2.1 formulierte konzeptionelle Schema. Neben der in dem Beispiel 2.3 genannten Benutzergruppe existiere noch eine weitere Benutzergruppe, für deren Anwendungen ein Ausschnitt aus dem konzeptionellen Schema von Interesse ist. Diese Benutzergruppe sei für die Gehaltsabrechnung zuständig. Definieren Sie ein Subschema für diesen Anwendungszweck. Verwenden Sie hierbei die Zugriffsspezifikation:

ACCESS IS RETRIEVAL AND UPDATE,

da im Rahmen einer Gehaltsabrechnung auch eine Änderung von Daten, beispielsweise des Alters oder des Gehalts, auftreten kann. Wählen Sie außerdem für die Dateneinheiten, die Gegenstand von Vergleichen oder Berechnungen sein könnten und die nicht nur gelesen und angezeigt werden, die Festpunktdarstellung. Orientieren Sie sich bei der Formulierung des Subschemas an dem in Beispiel 2.3 vorgestellten Subschema.

2.2 Datenbankverwaltungssysteme

Datenbankverwaltungssysteme stellen die Verbindung zwischen Datenbanken und Datenbankbenutzern bzw. Anwendungsprogrammen her. Zu Beginn des Kapitels 2 wurden die Aufgaben, die sie hierbei erfüllen, grob umrissen. Zu den Aufgaben eines Datenbankverwaltungssystems gehören u.a.:

- Das Speichern von Daten und das Anlegen von Zugriffspfaden zu den gespeicherten Daten gemäß den Vorgaben des internen Schemas.
- Das Ausführen von Datenbankoperationen wie Lesen, Ändern, Löschen usw. von Datenobjekten gemäß den Manipulationsanweisungen in den Anwendungsprogrammen.
- Die Synchronisation von Benutzeraktivitäten, d.h. das Verhindern der gleichzeitigen Manipulation bestimmter Daten durch verschiedene Benutzer.

Aufgaben eines DBVS

- Das Prüfen der Berechtigung des Zugangs von Benutzern zur Datenbank und das Erteilen von Lese- und Schreiberlaubnissen während des Datenbankbetriebs.
- Das Protokollieren ein- und ausgehender Informationen und das Führen von Fehlerstatistiken.

Auf Synchronisations-, Datensicherheits- und Datenschutzprobleme geht das Kap. 6 ein. Hier stehen die anwendungsbezogenen Datenbankfunktionen und die prinzipielle Funktionsweise von Datenbankverwaltungssystemen im Vordergrund.

Anweisungen zur Datenbankmanipulation werden in Anwendungsprogrammen bekanntlich auf einer logischen Ebene, der Ebene externer Schemata, formuliert. Die Ausführung solcher Anweisungen ist mit einer Folge von Schematransformationen verbunden:

- Von den angesprochenen Objekten der externen Ebene ist zunächst zu den entsprechenden Objekten der konzeptionellen und schließlich zu den entsprechenden Objekten der internen Ebene überzugehen; nun erst können Datenbankzugriffe erfolgen.

Schematransformationen

- Umgekehrt sind bei der Bereitstellung von Daten aus der Datenbank die fraglichen Datenobjekte aus der internen Ebene zunächst in die konzeptionelle und dann in die externe Ebene zu transformieren.

Abgewickelt werden diese Schematransformationen von dem jeweiligen Datenbankverwaltungssystem. Es bedient sich hierbei sogenannter Transformationsregeln. Eine schematische Darstellung der prinzipiellen Zusammenhänge zeigt die Abb. 2.2.

Laut Abb. 2.2 treten zwei Transformationsstufen auf, eine Transformation zwischen externen Schemata und dem konzeptionellen Schema (ES/KS-Transformation) sowie eine Transformation zwischen dem konzeptionellen und dem internen Schema (KS/IS-Transformation). Die Regeln zur ES/KS-Transformation legen fest, wie Datenobjekte und Beziehungen zwischen den Datenobjekten der externen Ebene aus Objekten und Objektbeziehungen des konzeptionellen Modells zusammengesetzt sind. Und die Regeln zur KS/IS-Transformation beschreiben die Zusammenhänge zwischen den Datenobjekten und Objektbeziehungen des konzeptionellen Modells und den gemäß dem internen Schema zu realisierenden physischen Datenstrukturen.

zwei Stufen der Schematransformation

Schematransformationen mit Transformationsregeln

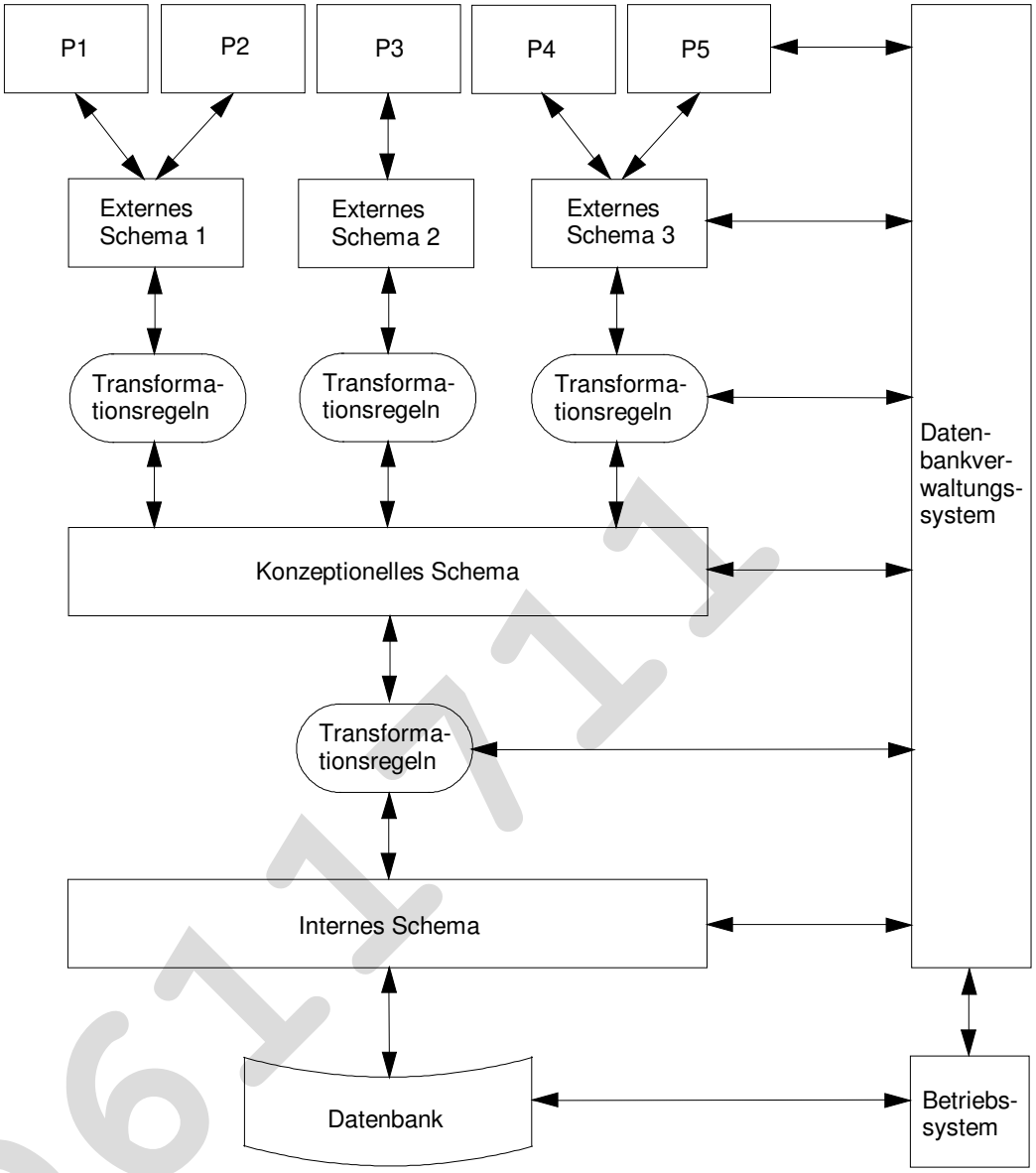


Abb. 2.2. Schematische Darstellung der Ebenenübergänge in einem Datenbanksystem.

Regeln zur Transformation von Datenobjekten

Transformationsregeln können impliziter oder expliziter Natur sein. Implizite Transformationsregeln müssen nicht erst formuliert werden. Sie ergeben sich aus den Schemabeschreibungen. So liegt eine implizite Transformationsbeziehung vor, wenn zwei einander entsprechende Datenobjekte benachbarter Ebenen den gleichen Namen aufweisen. Gehören beide Datenobjekte darüberhinaus dem gleichen Datentyp an, so ist die Transformationsregel eine Identitätsbeziehung. Andernfalls beinhaltet die Transformationsregel eine Typumwandlung. Explizite Transformationsregeln sind von den zuständigen Instanzen vorzugeben. Eine explizite KS/IS-Transformationsregel kann beispielsweise die Aufspaltung eines Datensatzes der konzeptionellen Ebene in einen häufig und einen selten benutzten Teilsatz der internen Ebene vorsehen. Auf diese Weise lässt sich der Zugriff zu Satzkomponenten beschleunigen.

Übungsaufgabe 2.7

Betrachtet seien das in Beispiel 2.1 angegebene konzeptionelle Schema sowie das daraus abgeleitete und in Beispiel 2.3 beschriebene Subschema. In diesen Schemata treten einander entsprechende Dateneinheiten auf, zwischen denen Identitätsbeziehungen bestehen, oder die lediglich mittels einer Typumwandlung ineinander umgesetzt werden können. Geben Sie je ein Beispiel für diese beiden Fälle an.

Die behandelten Schematransformationen vermitteln nur ein teilweises Bild der Funktionsweise von DBVS. Eine grobe Gesamtdarstellung der Funktionsweise eines DBVS findet sich in Abb. 2.3.

Funktionsweise eines DBVS

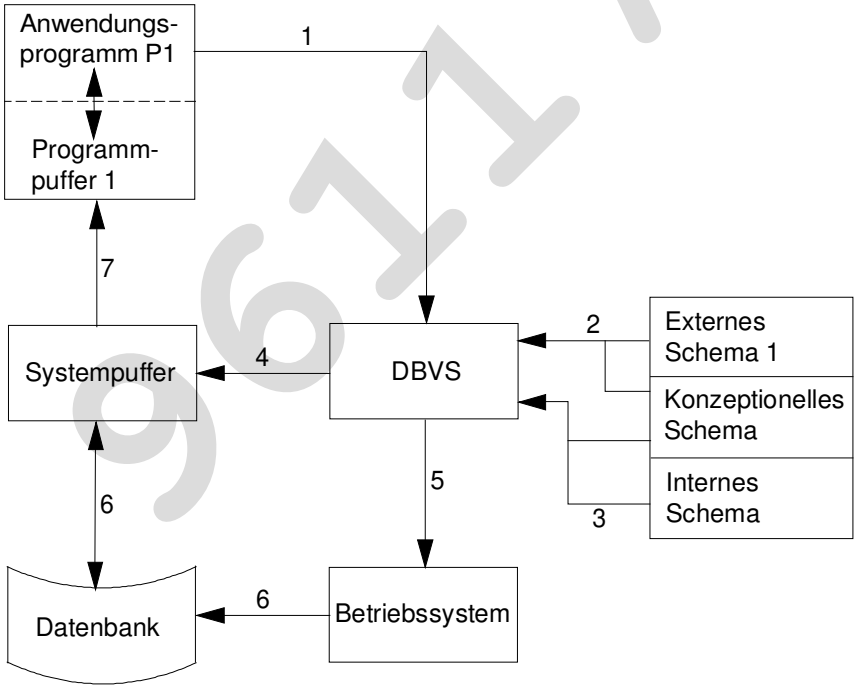


Abb. 2.3. Grobdarstellung der Funktionsweise eines DBVS.

Aus Abb. 2.3 geht insbesondere das Zusammenwirken eines DBVS mit einem Anwendungsprogramm, dem Betriebssystem und der Datenbank hervor. Die Schritte des Funktionsablaufs sind durch Ziffern gekennzeichnet, welche die Tätigkeitsabfolge bei einem Zugriff des Anwendungsprogramms P1 auf die Datenbank erkennen lassen. Zu den einzelnen Schritten sei folgendes bemerkt:

(1) Der Datenbankzugriff im Anwendungsprogramm P1 wird durch eine DML-Anweisung ausgelöst. Diese Anweisung möge wie folgt lauten:

LIES ARTIKELSATZ, ARTIKELNR = 0418

Die DML-Anweisung wird an das DBVS übergeben.

(2) Das DBVS interpretiert die übergebene DML-Anweisung und ermittelt die konzeptionelle Beschreibung des mit der Anweisung angesprochenen Datenobjekts. Letzteres geschieht mittels einer ES/KS-Transformation.

(3) Das DBVS transformiert die konzeptionelle Beschreibung des angesprochenen Datenobjekts in die Speicherstruktur, d.h. die interne Ebene, und ermittelt den Zugriffspfad für das gesuchte Datenobjekt bzw. den gesuchten Datensatz.

Erläuterungen zur Funktionsweise eines DBVS

(4) Das DBVS ermittelt die Seite (engl. page), in welcher der gesuchte Satz gespeichert ist und prüft, ob sich die Seite bereits im Systempuffer befindet. Ist dies der Fall, so erübrigt sich ein Datenbankzugriff und der Funktionsablauf wird mit dem Schritt (7) fortgesetzt. Andernfalls muss die Seite aus der Datenbank in den Systempuffer gebracht werden, in der sich der gesuchte Satz befindet. Dieser Seitentransfer wird in Schritt (5) initiiert.

(5) Das DBVS fordert zwei Betriebssystemdienste - hier zwei physische Datenbankzugriffe - an:

- Auslagern bzw. Schreiben der im Systempuffer zu ersetzenden Seite in die Datenbank.
- Lesen der Seite, die den gesuchten Satz enthält, in der Datenbank und Einlagern der Seite in den freigewordenen Platz des Systempuffers.

(6) Das Betriebssystem führt die beiden angeforderten Dienste aus. Für die Datenbankzugriffe verwendet es Standardroutinen, welche das Lesen und das Schreiben auf dem verwendeten externen Speichermedium gestatten.

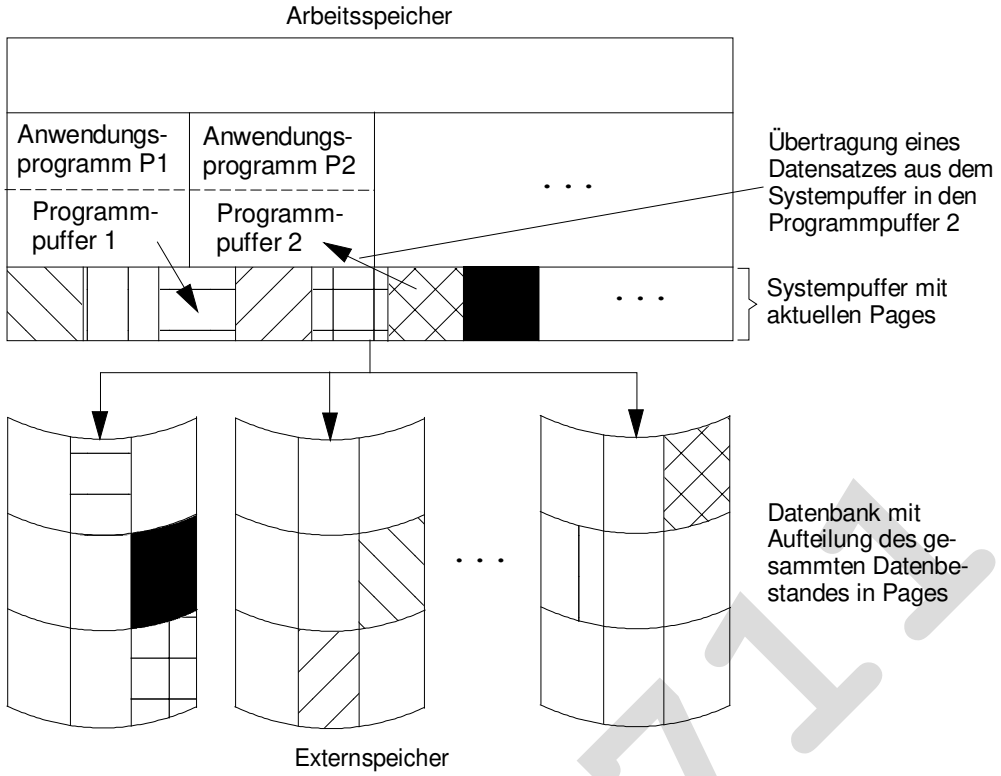
(7) Das DBVS greift auf den gesuchten Satz in der in den Systempuffer eingelagerten Seite zu, transformiert ihn in zwei Schritten in die durch das externe Schema 1 definierte Form und legt ihn dann in dem Programmpuffer 1 des Anwendungsprogramms P1 ab.

(8) Das Anwendungsprogramm P1 verarbeitet den Satz nun mit Anweisungen der Gastsprache. Eine Anweisung zur Manipulation des Satzes könnte beispielsweise lauten:

ERHÖHE ARTIKELBESTAND UM 250

Pagingkonzept

Stillschweigend wurde eben unterstellt, dass der Datentransfer zwischen Anwendungsprogrammen und Datenbank unter Verwendung eines Pagingkonzepts stattfindet. Bei diesem Konzept werden die zuletzt angesprochenen Teile der Datenbank in einem Bereich des Arbeitsspeichers, genannt Systempuffer, gehalten. Zwischen Datenbank und Systempuffer werden Speicherbereiche fester Größe, genannt Seiten oder Pages, ausgetauscht. Da sich Daten, auf die zugegriffen werden soll, in den aktuellen Pages des Systempuffers befinden können, ergeben sich auf diese Weise geringere mittlere Zugriffszeiten. Eine schematische Darstellung der auf dem Pagingkonzept beruhenden Speicherverbindungen zeigt die Abb. 2.4.



Speicherverbindungen in einem Datenbanksystem

Abb. 2.4. Schematische Darstellung der Speicherverbindungen in einem Datenbanksystem.

In Abb. 2.4 sind einige Seiten des Systempuffers und die dazugehörigen Seiten der Datenbank grafisch hervorgehoben. Die in den Systempuffer eingelagerten Bereiche der Datenbank werden also doppelt gespeichert, einerseits in der Datenbank und andererseits im Systempuffer.

Übungsaufgabe 2.8

Unter welchen Umständen kann es zwischen dem Inhalt des Systempuffers und den Inhalten der entsprechenden Datenbankbereiche zu Abweichungen kommen und wodurch werden diese inhaltlichen Diskrepanzen beseitigt?

Auswahl eines DBVS

Mit einigen Anmerkungen zur Auswahl von DBVS sei das vorliegende Kapitel abgeschlossen. Auf dem Softwaremarkt werden DBVS für Mainframes und Arbeitsplatzrechner angeboten. Auf die einzelnen Systeme kann hier nicht eingegangen werden. Hingewiesen werde jedoch auf einige Aspekte, die bei der Auswahl eines DBVS zu beachten sind:

- *Datenmodell*: Das DBVS sollte ein realitätsnahes, semantisch nicht armes Datenmodell unterstützen. Auf unterschiedliche Datenmodelle und den Vorgang der Datenmodellierung geht das nächste Kapitel ausführlich ein.
- *Datenkonsistenz*: Angesprochen ist hier die Widerspruchsfreiheit der gespeicherten Daten. Ein DBVS sollte zur Datenkonsistenz durch das Unterstützen von Konsistenzregeln beitragen. Konsistenzregeln garantieren die Widerspruchsfreiheit der Daten.
- *Datensicherheit*: Hier geht es um die Sicherheit der Daten im Falle von technischen Fehlern, Beschädigungen usw. Mit Hilfe geeigneter Routinen zur Rekonstruktion zerstörter Daten bzw. zum Wiederanlauf in Fehlersituationen sollte ein DBVS zur Datensicherheit beitragen.

Beurteilungskriterien

- *Datenschutz*: Dieser Begriff bezeichnet den Schutz personenbezogener Daten vor unbefugtem Zugriff und vor Missbrauch. Ein DBVS kann keinen hundertprozentigen Datenschutz gewährleisten. Doch kann es den unbefugten Zugriff zu Daten mit Hilfe von Routinen zur Vergabe und zum Prüfen von Zugriffsberechtigungen erheblich erschweren.
- *Vielfachzugriff*: Unter Vielfachzugriff ist der gleichzeitige Zugriff mehrerer Benutzer auf die gespeicherten Daten zu verstehen. Vielfachzugriffe sind nur im Falle von DBVS möglich, welche Routinen zur Synchronisation der Benutzeraktivitäten enthalten. Synchronisationsmechanismen verhindern beispielsweise, dass ein Benutzer auf Daten zugreift, die ein anderer Benutzer gerade verändert.
- *Effizienz*: Der Effizienzbegriff betrifft hier das Zeitverhalten des DBVS bei Zugriffen der Benutzer zur Datenbank. Ob ein DBVS über leistungsfähige Zugriffsroutinen verfügt, zeigt sich bei Zugriffen zu größeren Datenbeständen. Im Falle wiederkehrender interaktiver Abläufe sind allenfalls Systemantwortzeiten im Bereich von wenigen Sekunden tolerierbar.
- *Benutzerfreundlichkeit*: Angesprochen sind hier mehrere Teilaspekte. Zum einen sollte das DBVS dem Benutzer einfach zu erlernende und leistungsfähige Sprachen zur Datenmanipulation und zur Formulierung von Abfragen anbieten. Außerdem sollte das DBVS eine gut durchgestaltete Oberfläche aufweisen, welche dem Benutzer das Erschließen der vollen Systemfunktionalität leicht macht. Und schließlich trägt eine in das DBVS einbezogene Programmierunterstützung, beispielsweise in Form eines Maskengenerators, eines Listengenerators usw., zur Benutzerfreundlichkeit bei.

Inhalt des Kurses

Sämtliche angesprochenen Aspekte sind für die Entwicklung und den Betrieb von Datenbanksystemen von erheblicher Bedeutung. Im Laufe des Kurses werden sie daher wieder aufgegriffen und vertieft. So werden Datenmodelle und die Datenmodellierung eingehend in Kap. 3 behandelt. Auf Datenbanksprachen und insbesondere die Abfragesprache SQL geht das Kap. 4 ein. Unmittelbar die Effizienz berühren die in Kap. 5 angesprochenen Verfahren der physischen Datenorganisation und der die Aspekte Datenkonsistenz, Datensicherheit und Datenschutz umschließende Problemkreis der Datenintegrität ist Gegenstand des Kap. 6.

Lösungen zu den Übungsaufgaben

Übungsaufgabe 1.1

Das Band *Kundenumsätze-neu* enthält aktuelle, nichtkumulierte Umsatzdaten, während auf dem Band *Kundenumsatz-alt* kumulierte Umsatzdaten aufgezeichnet sind. Bei der Kundenumsatzfortschreibung geht es nun darum, die kumulierten Umsatzdaten um die aktuellen Kundenumsätze zu erhöhen. Dies geschieht dadurch, dass pro Kunde je ein Satz von *Kundenumsätze-alt* und *Kundenumsätze-neu* gelesen, die Kumulierung durchgeführt und das Ergebnis auf *Kundenumsätze* abgelegt wird. Die Datei *Kundenumsätze* spielt bei der nächsten Umsatzfortschreibung die Rolle von *Kundenumsätze-alt*. Benötigt werden also drei Laufwerke bzw. Dateien.

Übungsaufgabe 1.2

Sämtliche auf eine Datei zugreifenden Programme enthalten die vollständigen Informationen über den strukturellen Aufbau der Datei sowie die notwendigen Befehle zur Dateiverarbeitung. Verändert man z.B. in dem Programm *Fakturierung* in der Abb. 1.3 den Aufbau der Datei *Kundenumsätze-neu*, so müssen sämtliche weiteren Programme, welche die Datei *Kundenumsätze-neu* verarbeiten, an die neue Struktur angepasst werden. Im Fall des Beispiels muss also das Programm *Kundenumsatzfortschreibung* angepasst werden. Wird diese Änderung nicht durchgeführt, weil z.B. nicht alle auf die Datei zugreifenden Programme bekannt sind, so führt die Verarbeitung zu undefinierten Zuständen bzw. Fehlersituationen.

Übungsaufgabe 1.3

Im Falle der sequentiellen Verarbeitung mit Magnetbänden ist eine Kumulierung von Kundenumsätzen nur bei sortierter Verarbeitung möglich. Dazu ist es laut Abb. 1.3 notwendig, die bis dato kumulierten Kundenumsätze um die neuen Kundenumsätze zu erhöhen und in die Datei *Kundenumsätze* zu übernehmen Die beiden Eingabedateien *Kundenumsätze-alt* und *Kundenumsätze-neu* sind ebenso nach Kunden sortiert wie die Ausgabedatei *Kundenumsätze*, von der ausgehend die Umsatzstatistik erstellt wird. Bei beiden Verarbeitungsschritten, der Umsatzfortschreibung und der Statistikerstellung, müssen jeweils alle Sätze der beteiligten Dateien gelesen bzw. geschrieben werden. Außerdem werden, wie die Übungsaufgabe 1.1 zeigt, drei Bandlaufwerke bzw. Dateien benötigt.

Im Falle der Verwendung eines Dateiverwaltungssystems, welches Direktzugriffe auf die Sätze einer auf einem Direktzugriffsspeicher abgelegten Datei unterstützt, genügt dagegen nur eine Datei mit aktuellen kumulierten Kundenumsätzen. Jeder neue Kundenumsatz löst einen Aktualisierungsvorgang aus, bei dem jeweils nur auf den Datensatz des betroffenen Kunden zugegriffen wird. Der kumulierte Umsatz dieses Kunden wird um den neuen Kundenumsatz erhöht und das Kumulationsergebnis wird in die Datei zurückgeschrieben. Eine zusätzliche Datei ist daher ebenso wenig erforderlich wie das Sammeln und Sortieren neuer Kundenumsätze.

Übungsaufgabe 1.4

Redundanzen treten immer noch in einem erheblichen Umfang auf, weil

- viele Daten für verschiedene Verarbeitungszwecke benötigt und daher auch in verschiedene, auf diese Zwecke zugeschnittene Dateien einbezogen werden,
- und weil die für die verschiedenen Anwendungen zuständigen Benutzer häufig nicht hinreichend über die in anderen Anwendungsbereichen geführten Datenbestände informiert sind.

Physische Datenabhängigkeiten betreffen mehr Anwendungsprogramme als im Fall der separaten Datenverwaltung, weil beim Einsatz von Dateiverwaltungssystemen in Verbindung mit Direktzugriffen zu Daten tendenziell weniger Dateien auftreten, die nur von einem oder wenigen aufeinander aufbauenden Programmen benutzt werden. Mit der Anzahl von Programmen, die eine Datei benutzen, wächst aber das Ausmaß der physischen Datenabhängigkeit, da die Änderung einer Datei dann mehr Programme betrifft.

Inflexibilitäten resultieren daraus, dass für alle Programme, die eine Datei benutzen, prinzipiell die gleichen Daten bereitgestellt werden. Ob ein Anwendungsprogramm sämtliche in einem gelesenen Datensatz verzeichneten Daten benötigt oder nur einen Teil der Daten, wird durch das Dateiverwaltungssystem nicht berücksichtigt. Vielmehr werden lediglich komplette Datensätze bereitgestellt, aus denen das einzelne Programm die jeweils interessierenden Daten zu selektieren hat.

Übungsaufgabe 1.5

Hierunter kann man sich die Trennung zwischen dem *WAS* und dem *WIE* vorstellen. Der Benutzer formuliert *was* für Daten er bearbeiten möchte. Das DBVS setzt die Abfragen um und formuliert, *wie* aus dem physischen Datenbestand die gewünschten Daten zu generieren sind. Dabei transformiert das DBVS die auf der logischen Ebene formulierte Abfrage in eine auf den physischen Datenbestand zugreifende Suchoperation und transformiert dann die auf die Abfrage zutreffenden physischen Datenobjekte zurück in die logische Ebene, auf der auch die Antwort an den Benutzer formuliert wird. Bei der Suche bedient sich das DBVS einer dem Benutzer bzw. dem Anwendungsprogramm unbekannten und damit von den Anwendungsprogrammen unabhängigen Zugriffsorganisation. Unter den Schemaübergängen sind also die Übergänge zwischen der logischen Formulierung von Abfragen bzw. Antworten und ihrer physischen Darstellungsform zu verstehen.

Übungsaufgabe 1.6

Sämtliche existierenden Einzelsichten werden bei dem 3-Schichten-Konzept durch das konzeptionelle Modell repräsentiert. Kommt nun ein neues externes Datenmodell hinzu, so basiert es entweder ebenfalls auf dem konzeptionellen Modell oder es enthält Datenobjekte, die in dem konzeptionellen Modell bislang nicht auftreten. Anders als im ersten Fall muss das konzeptionelle Modell im zweiten Fall um die hinzukommenden Datenobjekte erweitert werden. Davon bleiben aber die bereits vorhandenen externen Datenmodelle unberührt, weil sie die hinzukommenden Datenobjekte nicht einschließen.

Übungsaufgabe 2.1

Normgebungen beziehen sich in der Regel auf komplexe Systeme, die Gegenstand wissenschaftlicher Untersuchungen und wirtschaftlicher Verwertungen sind. Sie dienen dazu, zwischen den Beteiligten Festlegungen zu treffen, welche einerseits die Nutzung der Systeme ermöglichen oder erleichtern, und andererseits die Hersteller der Systeme nicht mit zu starken Einschränkungen belegen. Mit dem ANSI-Architekturmodell wurden Herstellern und Anwendern Empfehlungen mit Normcharakter zum Aufbau von Datenbanken an die Hand gegeben. Dass man damit die Entwicklung und den Aufbau von Datenbanken in die richtige Richtung leitete, zeigt sich daran, dass das ANSI-Architekturmodell auch heute noch Anwendung findet.

Übungsaufgabe 2.2

```
SCHEMA FIRMA2;
    RECORD TYPE ARTIKEL;
        DATA ITEM artikelnr, INTEGER, KEY,
                ASSERT artikelnr > 0;
        DATA ITEM  bezeichnung, CHARACTER;
        DATA ITEM lagernr, INTEGER,
                ASSERT lagernr IS 0 THRU 5;
        DATA ITEM mindestbestand, INTEGER;
                ASSERT mindestbestand > 0;
        DATA ITEM bestand, INTEGER;
                ASSERT bestand > 0;
        DATA ITEM einkaufspreis, REAL, UNIT IS DM;
        DATA ITEM verkaufspreis, REAL, UNIT IS DM,
                ASSERT verkaufspreis > einkaufspreis;
    END RECORD TYPE;
END SCHEMA.
```


Übungsaufgabe 2.3

(1) Das Anschriftenformat der Kunden einer Firma ändert sich. Zusätzlich zu den Länderkennzeichen und der Postleitzahl muss noch eine Angabe zum Landesteil (Ost oder West) gespeichert werden. Hierzu ist es erforderlich, ein zusätzliches Feld in das konzeptionelle Schema aufzunehmen. Anwendungssysteme, welche auf Kundenanschriften zugreifen, sind entsprechend anzupassen. Nicht angepasste Anwendungssysteme verarbeiten die Anschriftendaten in der ursprünglichen Form. Die Nichtanpassung impliziert also keinen Programmfehler.

(2) Der in einer Firma bisher ausgegliederte Anwendungsbereich der Lohnbuchhaltung soll mit dem bereits für andere Anwendungen eingesetzten Datenbanksystem verwaltet werden. Das bestehende konzeptionelle Schema ist entsprechend den Anforderungen an die Lohnbuchhaltung zu erweitern. Die Verbindungen zu bestehenden Datenobjekten sind darzustellen. Auch in diesem Fall bleiben die bereits auf der Datenbank operierenden Anwendungssysteme weiterhin ohne Änderungen einsetzbar.

Übungsaufgabe 2.4

```
STORAGE MODULE FOR SCHEMA FIRMA2;
    ARTIKEL, CONTINUOUS FILE, FILE NAME IS artikeldatei,
    MEDIUM IS DISK, DEVICE IS IBM 3330;
        artikelnr, FIXED DECIMAL (6,0), STORAGE KEY,
                SORT ASCENDING;
        bezeichnung, STRING (20), INDEX USING B-TREE;
        lagernr,  FIXED DECIMAL (1,0);
        mindestbestand,  FIXED DECIMAL (8,2);
        bestand,  FIXED DECIMAL (9,2), INDEX USING POINTER ARRAY;
        einkaufspreis,  FIXED DECIMAL (8,2);
        verkaufspreis,  FIXED DECIMAL (8,2);
    END artikeldatei;
END STORAGE MODULE.
```

Übungsaufgabe 2.5

Beispielsweise können bei der Erstellung einer nach Gehaltsgruppen sortierten Gehaltsstatistik über den Sekundärschlüssel *gehalt* die Anzahl der Mitarbeiter der entsprechenden Gehaltsgruppe ermittelt werden, da der Sekundärschlüssel *gehalt* einen unmittelbaren Zugriff auf sämtliche Mitarbeiter einer Gehaltsgruppe gestattet.

Übungsaufgabe 2.6

```
SUBSCHEMA GEHALTSABRECHNUNG OF SCHEMA FIRMA;
    RECORD TYPE MITARBGEHALT OF RECORD TYPE MITARBEITER;
        ACCESS IS RETRIEVAL AND UPDATE;
        DATA ITEM mitarbeiternr, KEY, CHARACTER (3);
        DATA ITEM name, CHARACTER (20);
        DATA ITEM adresse, ALPHANUMERIC (30);
        DATA ITEM alter, FIXED DECIMAL (2,0), UNIT IS JAHRE;
        DATA ITEM gehalt, FIXED DECIMAL (6,2), UNIT IS DM;
    END RECORD TYPE;
END SUBSCHEMA.
```

Übungsaufgabe 2.7

Identitätsbeziehung: Die Dateneinheit *name* besitzt in beiden Schemata den gleichen Typ. Es wird in dem Subschema ADRESSBUCH lediglich zusätzlich noch die Länge auf 20 Stellen begrenzt.

Typumwandlung: Eine Typumwandlung findet für die Dateneinheit *mitarbeiternr* statt. Der Datentyp INTEGER ist in denTyp CHARACTER (3) umzuwandeln.

Übungsaufgabe 2.8

Inhaltliche Unterschiede zwischen dem Systempuffer und den referenzierten Datenbankbereichen entstehen immer dann, wenn Schreibzugriffe auf den Systempuffer erfolgen. Diese Unterschiede werden beseitigt, indem die geänderten Seiten des Systempuffers auf die referenzierten Seiten der Datenbank abgelegt werden.