HANSER

Taschenbuch Datenbanken

Herausgegeben von Thomas Kudraß

ISBN-10: 3-446-40944-0
ISBN-13: 978-3-446-40944-6

Leseprobe

1 Datenbanken: Grundlagen und Überblick

Thomas Kudraß

In diesem Kapitel werden die wichtigsten Grundbegriffe und Konzepte im Fachgebiet Datenbanken eingeführt und zueinander in Beziehung gesetzt (→ 1.1 bis 1.6). Darauf aufbauend, erfolgt eine Einordnung des Fachgebiets Datenbanken in die Informatik sowie ein Überblick über die zentralen Themen des Faches, die den Gegenstand der Kapitel 2 bis 17 dieses Buches bilden (→ 1.7). Die Historie (→ 1.8) und ein Blick auf aktuelle Trends in der Forschung (→ 1.9) ermöglichen eine zeitliche Einordnung des erreichten Entwicklungsstandes von Datenbanksystemen.

1.1 Dateien und Dateisysteme

Informationen werden durch Daten repräsentiert, die in einem Kontext interpretiert werden. Die Information wird dazu in Zeichen bzw. Zeichenketten codiert, was auf der Grundlage von Regeln erfolgt, die man als Syntax bezeichnet. Die Codierung geschieht auf der Grundlage von Codierregeln.

Daten sind maschinell lesbar und verarbeitbar. Sie beinhalten die ursprünglichen Benutzerdaten sowie andere zusätzliche Elemente, die für eine maschinelle Verarbeitung erforderlich sind.

❑ *Beispiel:* 04275 kann eine Postleitzahl sein und damit zu den Daten gehören. Wenn dazu der Name des Ortes genannt wird, so wird daraus eine Information. Der Wert der Postleitzahl kann durch eine Folge von 5 Zeichen repräsentiert werden: „04275“. Die Codierung dieser Zeichen erfolgt auf der Grundlage des ASCII-Codes. Somit wird diese Zeichenkette hexadezimal als Bytefolge X'3034323735' repräsentiert.

Man unterscheidet

- strukturierte Daten, z. B. Telefonbücher, Adresslisten,
- unstrukturierte Daten, z. B. Textdokumente, digitale Bilder,
- semistrukturierte Daten, z. B. Produktkataloge, Patientenakten.

Eine **Datei** (file) ist ein Bestand inhaltlich zusammengehöriger Daten, der auf einem Datenträger persistent abgelegt ist. Daten sind *persistent*, wenn sie über die Laufzeit des Programmes hinaus, in dem sie verarbeitet werden, weiter existieren. Anderenfalls spricht man von *transienten* Daten.

Dateien werden in den meisten Betriebssystemen über Dateisysteme verwaltet und können auf unterschiedliche Art und Weise organisiert sein (→ 6.3, 8.1). Logisch lässt sich eine strukturierte Datei als eine Folge von **Datensätzen** (records) betrachten. Ein Datensatz besteht aus mehreren Feldern (fields), deren Anzahl und Struktur in jedem Datensatz dieser Datei gleich ist.

❑ *Beispiel:* Die Datei `Stadt` besteht aus Datensätzen mit den Feldern `PLZ` und `Ort`: `04275,Leipzig; 64289,Darmstadt; 99425,Weimar`.

Zur Arbeit mit Dateien stehen verschiedene Operationen zur Verfügung:

- Anlegen (create) und Löschen (drop) von Dateien.
- Öffnen (open) und Schließen (close) von Dateien. Das Öffnen einer Datei ist die Voraussetzung, um auf diese zuzugreifen.
- Lesen (read) und Schreiben (write) von Dateien.

Bei der herkömmlichen Dateiverarbeitung definiert und implementiert jeder Benutzer die Dateien, die für eine bestimmte Anwendung benötigt werden, als Teil der Anwendungsentwicklung.

❑ *Beispiel:* So könnte ein Handelsunternehmen die Adressdaten seiner Kunden für Marketingmaßnahmen in einer Datei verwalten. In einer anderen Software zur Abwicklung der Bestellungen werden ebenfalls die Adressen der Kunden erfasst.

Die herkömmliche Arbeit mit Dateien führt zu dem Problem der **Redundanz**, d. h. dem mehrfachen Vorhandensein ein und derselben Information, die in unterschiedlichen Dateien gespeichert werden kann.

Nachteile redundanter Datenhaltung:

- erhöhter Speicherplatzbedarf für die Mehrfachspeicherung.
- erhöhter Aufwand zur Synchronisation von Datenänderungen. Dabei muss ein geänderter Datensatz mit anderen Datensätzen, die die gleiche Information beinhalten, abgeglichen werden.

Redundante Speicherung birgt immer die Gefahr von *Inkonsistenz*, d. h. die widersprüchliche Speicherung gleicher Informationen.

1.2 Terminologie

Eine **Datenbank** (DB) ist eine Sammlung von Daten, die einen Ausschnitt der realen Welt beschreiben.

Der übliche Gebrauch des Begriffs Datenbank wird in vielen Fällen aber eingeschänkt. Demnach hat eine Datenbank folgende Eigenschaften:

- Eine Datenbank stellt Aspekte der realen Welt dar. Dieser Weltausschnitt wird auch als Diskursbereich (universe of discourse) bezeichnet.
- Eine Datenbank ist eine integrierte, d. h. logisch zusammenhängende Sammlung von Daten für einen festgelegten Zweck. Sie wird von einer bestimmten Benutzergruppe in Anwendungen eingesetzt.

Ein **Datenbankmanagementsystem (DBMS)** ist ein Softwaresystem (d. h. eine Sammlung von Programmen), das dem Benutzer das Erstellen und die Pflege einer Datenbank ermöglicht. Dies umfasst die Definition, die Erzeugung und Manipulation von Datenbanken.

Das DBMS legt das *Datenmodell* der Datenbank (*Datenbankmodell*) fest. Die Auswahl des DBMS ist eine kritische Entscheidung. Hierbei ist zu unterscheiden zwischen kommerzieller und freier Software.

▶ *Hinweis:* Für ein DBMS wird selten auch der Begriff Datenbankverwaltungssystem (DBVS) oder Datenbankbetriebssystem verwendet.

Ein DBMS zusammen mit einer oder mehreren Datenbanken wird als **Datenbanksystem (DBS)** bezeichnet.

Ein **Datenmodell** enthält drei Bestandteile:
- *Datenstrukturen*: Objekte und deren Beziehungen (statische Eigenschaften),
- *Operationen* und Beziehungen zwischen Operationen (dynamische Eigenschaften),
- *Integritätsbedingungen* auf Objekten und Beziehungen. Dies sind Regeln, die die Menge der erlaubten Zustände bzw. Zustandsübergänge definieren. Sie können modellinhärent sein oder explizit definiert werden.

❑ *Beispiel:* Objekte sind dabei etwa Kunden, Artikel, Lieferanten, aber auch Ereignisse wie Bestellungen. Eine Beziehung zwischen diesen Objekten ist beispielsweise die Bestellung von Artikeln durch Kunden. Operationen auf Objekten sind das Erfassen von Neukunden und die Pflege der Artikeldaten. Weiterhin kann man Integritätsbedingungen an Kundenobjekte formulieren, z. B. dass ihre Kundennummer jeweils unterschiedlich ist (Schlüsselbedingung) und dass eine Bestellung genau einem Kunden zugeordnet werden muss.

Ein **Datenbankmodell** ist ein Datenmodell für ein Datenbanksystem. Es bestimmt, auf welche Art und Weise Daten prinzipiell gespeichert werden und wie man die Daten manipulieren kann.

▶ *Hinweis:* Datenbankmodelle werden im Kontext von Datenbanksystemen oft einfach nur als Datenmodelle bezeichnet.

Ein Datenmodell kann auf Basis eines Datenbankmodells implementiert werden oder selbst als Datenbankmodell dienen.

❑ *Beispiel 1:* Das mehrdimensionale Datenmodell kann relational in einer Datenbank umgesetzt werden, was als ROLAP (Relational Online Analytical Processing) bezeichnet wird (→ 13.3).

❑ *Beispiel 2:* XML kann in einer relationalen Datenbank (→ 11.3) oder in einer reinen XML-Datenbank (nativ) gespeichert werden (→ 11.4).

Daneben gibt es abstrakte Datenbankmodelle, die im Datenbankentwurf eingesetzt werden, aber keine Entsprechung in einem konkreten DBMS haben. Solche Modelle werden oft als **semantische Datenmodelle** oder **Informationsmodelle** bezeichnet, ein typisches Beispiel ist das *Entity-Relationship-Modell* (*ERM*) (→ 2).

Die ersten (prärelationalen) Datenbankmodelle waren satzorientiert, es sind dies das **hierarchische** und das **Netzwerk-Datenmodell**, angelehnt an die Datenstrukturen von kommerziellen Programmiersprachen. Bei hierarchischen Datenmodellen können Datensätze (record types) nur in hierarchischen Beziehungen ($1:N$) stehen, beim Netzwerkmodell auch in vernetzten $M:N$-Beziehungen. Das hierarchische Datenmodell erlebt eine Renaissance durch das Aufkommen von XML-Datenbanken und die Nutzung von XML als Datenmodell.

Das **relationale Datenmodell** (RDM) beruht auf dem Begriff der Relation, der im Wesentlichen ein mathematisches Modell für eine Tabelle beschreibt. Die Daten werden in Form von zweidimensionalen Tabellen verwaltet, die über Schlüssel (Primärschlüssel, Fremdschlüssel) miteinander verknüpft werden können (→ 3). Die meisten in der Praxis eingesetzten Datenbanksysteme beruhen auf einem relationalen DBMS (RDBMS).

Während das relationale Datenmodell eine klare mathematische Grundlage hat, gibt es eine Vielzahl objektorientierter Datenbankmodelle, die sich oft an objektorientierte Programmiersprachen (z. B. C++) angelehnt haben.

Demgegenüber haben die großen DBMS-Hersteller ihren relationalen Systemen objektorientierte Eigenschaften hinzugefügt und diese zu **objektrelationalen** Systemen weiterentwickelt.

▶ *Hinweis:* Allgemeine Lehrbücher über Datenbanken sind u.a. /1.1/, /1.2/, /1.4/, /1.5/, /1.9/.

1.3 Merkmale eines DBMS

1.3.1 Aufgaben eines DBMS

Gemäß seiner Definition muss ein DBMS folgende Funktionalitäten bieten:

Integrierte Datenhaltung. Das DBMS ermöglicht die *einheitliche* Verwaltung *aller* von den Anwendungen benötigten Daten. Somit wird jedes logische Datenelement, wie beispielsweise der Name eines Kunden, *an nur einer* Stelle in der Datenbank gespeichert. Dabei muss ein DBMS die Möglichkeit bieten, eine Vielzahl komplexer Beziehungen zwischen den Daten zu definieren sowie zusammenhängende Daten schnell und effizient miteinander zu verknüpfen. In manchen Fällen kann eine *kontrollierte Redundanz* allerdings nützlich sein, um die Effizienz der Verarbeitung von Anfragen zu verbessern (→ 9.3.1.3).

Sprache. Das DBMS stellt an seiner Schnittstelle eine Datenbanksprache (*query language*) für die folgenden Zwecke zur Verfügung:

- Datenanfrage (retrieval),
- Datenmanipulation (Data Manipulation Language, DML),
- Verwaltung der Datenbank (Data Definition Language, DDL),
- Berechtigungssteuerung (Data Control Language, DCL).

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, Programmierschnittstellen und grafische Benutzeroberflächen (*Graphical User Interface, GUI*). Die Möglichkeit eines Webzugriffs ist heute eine Grundvoraussetzung für den Einsatz eines DBMS.

Katalog. Der Katalog (*data dictionary*) ermöglicht Zugriffe auf die Datenbeschreibungen der Datenbank, die auch als Metadaten bezeichnet werden.

❑ *Beispiele:* Eigentümer und Erzeugungszeit von Datenbankobjekten (Tabellen), Beschreibung von Tabellen und ihren Spalten (Datentyp, Länge).

Benutzersichten. Für unterschiedliche Klassen von Benutzern sind verschiedene Sichten (*views*) erforderlich, die bestimmte Ausschnitte aus dem Datenbestand beinhalten oder diesen in einer für die jeweilige Anwendung benötigten Weise strukturieren. Die Sichten sind im externen Schema der Datenbank definiert (→ 1.4.2.2).

Konsistenzkontrolle. Die Konsistenzkontrolle, auch als *Integritätssicherung* bezeichnet, übernimmt die Gewährleistung der Korrektheit von Datenbankinhalten und der korrekten Ausführung von Änderungen. Ein korrekter Datenbankzustand wird durch benutzerdefinierte Integritätsbedingungen (*constraints*) im DBMS definiert, die während der Laufzeit der Anwendungen vom System kontrolliert werden. Daneben wird aber auch die physische Integrität sichergestellt, d. h. die Gewährleistung intakter Speicherstrukturen und Inhalte (Speicherkonsistenz).

Datenzugriffskontrolle. Durch die Festlegung von Regeln kann der unautorisierte Zugriff auf die in der Datenbank gespeicherten Daten verhindert werden (*access control*). Dabei kann es sich um personenbezogene Daten handeln, die datenschutzrechtlich relevant sind, oder um firmenspezifische Datenbestände. Rechte lassen sich auch auf Sichten definieren.

Transaktionen. Mehrere Datenbankänderungen, die logisch eine Einheit bilden, lassen sich zu Transaktionen zusammenfassen, die als Ganzes ausgeführt werden sollen (Atomarität) und deren Effekt bei Erfolg permanent in der Datenbank gespeichert werden soll (Dauerhaftigkeit).

Mehrbenutzerfähigkeit. Konkurrierende Transaktionen mehrerer Benutzer müssen synchronisiert werden, um gegenseitige Beeinflussung, z. B. bei Schreibkonflikten auf gemeinsam genutzten Daten, zu vermeiden (*concurrency control*). Dem Benutzer erscheinen die Daten so, als ob nur eine Anwendung darauf zugreift (Isolation).

Datensicherung. Das DBMS muss in der Lage sein, bei auftretenden Hard- oder Softwarefehlern wieder einen korrekten Datenbankzustand herzustellen (*Recovery*).

1.3.2 Vorteile des Datenbankeinsatzes

Zusätzlich zu den genannten Merkmalen ergibt sich bei Einsatz eines Datenbanksystems eine Reihe von Vorteilen:

Nutzung von Standards. Der Einsatz einer Datenbank erleichtert Einführung und Umsetzung zentraler Standards in der Datenorganisation. Dies umfasst Namen und Formate von Datenelementen, Schlüssel, Fachbegriffe.

Effizienter Datenzugriff. Ein DBMS gebraucht eine Vielzahl komplizierter Techniken zum effizienten Speichern und Wiederauslesen (retrieval) großer Mengen von Daten.

Kürzere Softwareentwicklungszeiten. Ein DBMS bietet viele wichtige Funktionen, die allen Anwendungen, die auf eine Datenbank zugreifen wollen, gemeinsam ist. In Verbindung mit den angebotenen Datenbanksprachen wird somit eine schnellere Anwendungsentwicklung ermöglicht, da der Programmierer von vielen Routineaufgaben entlastet wird.

Hohe Flexibilität. Die Struktur der Datenbank kann bei sich ändernden Anforderungen ohne große Konsequenzen für die bestehenden Daten und die vorhandenen Anwendungen modifiziert werden (Datenunabhängigkeit, → 1.4.3).

Hohe Verfügbarkeit. Viele transaktionsintensive Anwendungen (z. B. Reservierungssysteme, Online-Banking) haben hohe Verfügbarkeitsanforderungen. Ein DBMS stellt die Datenbank allen Benutzern dank der Synchronisationseigenschaften *gleichzeitig* zur Verfügung. Änderungen werden nach Transaktionsende sofort sichtbar.

Große Wirtschaftlichkeit. Die durch den Einsatz eines DBMS erzwungene Zentralisierung in einem Unternehmen erlaubt die Investition in leistungsstärkere Hardware, statt jede Abteilung mit einem eigenen (schwächeren) Rechner auszustatten. Somit reduziert der Einsatz eines DBMS die Betriebs- und Verwaltungskosten.

1.3.3 Nachteile von Datenbanksystemen

Trotz der Vorteile eines DBMS gibt es auch Situationen, in denen ein solches System unnötig hohe Zusatzkosten im Vergleich zur traditionellen Dateiverarbeitung mit sich bringen würde:

- Hohe Anfangsinvestitionen für Hardware und Datenbanksoftware.
- Ein DBMS ist Allzweck-Software, somit weniger effizient für spezialisierte Anwendungen.
- Bei konkurrierenden Anforderungen kann das Datenbanksystem nur für einen Teil der Anwendungsprogramme optimiert werden.
- Mehrkosten für die Bereitstellung von Datensicherheit, Mehrbenutzer-Synchronisation und Konsistenzkontrolle.
- Hochqualifiziertes Personal erforderlich, z. B. Datenbankdesigner, Datenbankadministrator.
- Verwundbarkeit durch Zentralisierung (Ausweg Verteilung).

Unter bestimmten Umständen ist der Einsatz regulärer Dateien sinnvoll:

- Ein gleichzeitiger Zugriff auf die Daten durch mehrere Benutzer ist nicht erforderlich.
- Für die Anwendung bestehen feste Echtzeitanforderungen, die von einem DBMS nicht ohne weiteres erfüllt werden können.
- Es handelt sich um Daten und Anwendungen, die einfach und wohldefiniert sind und keinen Änderungen unterliegen werden.

1.3.4 Produkte

Tabelle 1.1 gibt einen Überblick über heute verfügbare DBMS verschiedener Hersteller und das zugrunde liegende Datenbankmodell (kommerzielle DBMS sind mit „$“ gekennzeichnet). Bei Anbietern objektrelationaler DBMS gibt es Unterschiede bei den objektorientierten Merkmalen.

Tabelle 1.1: DBMS-Produkte und -Hersteller

DBMS	$?	Hersteller	Modell/Charakteristik
Adabas	$	Software AG	NF^2-Modell (nicht normalisiert)
Cache	$	InterSystems	hierarchisch, „postrelational“
DB2	$	IBM	objektrelational
Firebird		–	relational, basierend auf InterBase
IMS	$	IBM	hierarchisch, Mainframe-DBMS
Informix	$	IBM	objektrelational
InterBase	$	Borland	relational
MS Access	$	Microsoft	relational, Desktop-System
MS SQL Server	$	Microsoft	objektrelational
MySQL		MySQL AB	relational
Oracle	$	ORACLE	objektrelational
PostgreSQL		–	objektrelational, hervorgegangen aus Ingres und Postgres
Sybase ASE	$	Sybase	relational
Versant	$	Versant	objektorientiert
Visual FoxPro	$	Microsoft	relational, Desktop-System
Teradata	$	NCR Teradata	relationales Hochleistungs-DBMS, speziell für Data Warehouses

1.4 Architektur eines Datenbanksystems

1.4.1 Architekturen

Der Begriff **Architektur** kann bei Datenbankanwendungen einen unterschiedlichen Kontext haben. Hierbei lassen sich nennen:

- *Systemarchitektur* eines DBMS: Es werden die Komponenten eines DBMS beschrieben, die sich in einem Schichtenmodell anordnen lassen. In Standardisierungsvorschlägen werden die Schnittstellen zwischen den DBMS-Komponenten genormt, nicht jedoch diese selbst.
- *Schemaarchitektur* eines DBS: Diese unterscheidet zwischen verschiedenen Abstraktionsebenen, in denen die Daten einer Datenbank repräsentiert werden (→ 1.4.2).
- *Anwendungsarchitektur*: In datenbankbasierten Anwendungen erfolgt die Aufteilung der Funktionalität in verschiedene logische Schichten (tiers), z. B. unter Performance-Gesichtspunkten. Dabei wird die Datenhaltung zumeist einem DBMS zugeordnet (Back-End) sowie das Zusammenwirken der Benutzerkomponenten bei der Abarbeitung einer DB-Anwendung beschrieben.
- *Verteilte Architektur*: Aspekte der Verteilung können alle drei der oben genannten Architekturen betreffen. Sie bilden somit eine eigenständige Dimension von Architektur. So kann ein DBS als verteiltes DBS auf mehreren Knoten eines Netzwerks operieren oder als föderiertes DBS aus mehreren (heterogenen) DBS bestehen.

▶ *Hinweis:* Architektur und Komponenten eines Datenbanksystems werden vertiefend in /1.3/, /1.6/, /1.7/ beschrieben.

1.4.2 Schemaarchitektur

1.4.2.1 Datenbankschema

Für jedes Datenmodell ist es wichtig, zwischen der Beschreibung der Struktur und dem Inhalt der Datenbank zu unterscheiden. Die strukturelle Beschreibung einer Datenbank auf Basis eines Datenmodells wird als **Datenbankschema** bezeichnet, das im Verlauf des Datenbankentwurfs jeweils in einer bestimmten Sprache (DDL) spezifiziert wird. Ein grafisch dargestelltes Schema wird Schemadiagramm genannt.

❑ *Beispiel:* Für die Speicherung von Lieferantendaten gibt es je nach Datenmodell (Text, Relationenmodell, XML) unterschiedliche Schemadefinitionen.

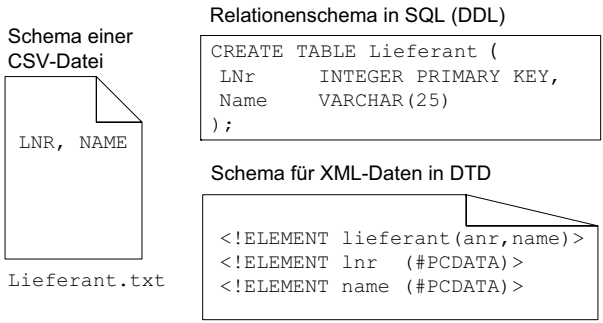

Bild 1.1: Schemadefinitionen im Vergleich

Ein Datenbankschema kann spezifisch für eine Anwendung definiert werden. Daneben gibt es aber auch vordefinierte Schemata für bestimmte Kategorien von Anwendungen, auch als *Referenzmodell* bezeichnet. Diese unterliegen einer Standardisierung, erleichtern den Datenaustausch und verkürzen die Entwicklungszeit von DB-Anwendungen.

❑ *Beispiel:* Für geografische Objekte in Geodatenbanken steht das ISO 19107 Spatial Schema als Referenzmodell zur Verfügung (→ 16.2.2).

Die in der Datenbank zu einem bestimmten Zeitpunkt gespeicherten Daten werden als Datenbankzustand bezeichnet. Jeder Datensatz stellt eine **Instanz** dar und entspricht einem Objekt (entity). Mit jeder Veränderung der Datenbank erhalten wir einen anderen Datenbankzustand. Das Schema wird auch als **Intension**, der Datenbankzustand als **Extension** bezeichnet. Die Veränderung des Schemas heißt **Schemaevolution**.

1.4.2.2 Drei-Ebenen-Architektur

Die Drei-Ebenen-Architektur (oder auch Drei-Schema-Architektur) beschreibt den grundlegenden Aufbau eines Datenbanksystems. Die Architektur wurde 1975 vom Standards Planning and Requirements Committee (SPARC) des American National Standards Institute (ANSI) entwickelt und formuliert das Prinzip der *Separation of Concerns* für Datenbanken /1.25/.

Die drei Ebenen umfassen Folgendes:

1. Auf der internen Ebene wird ein **internes Schema** definiert. Das physische Schema beschreibt die Details der physischen Speicherung der Daten (→ 9.3.3) und die Zugriffspfade wie z. B. Indexe (→ 8).