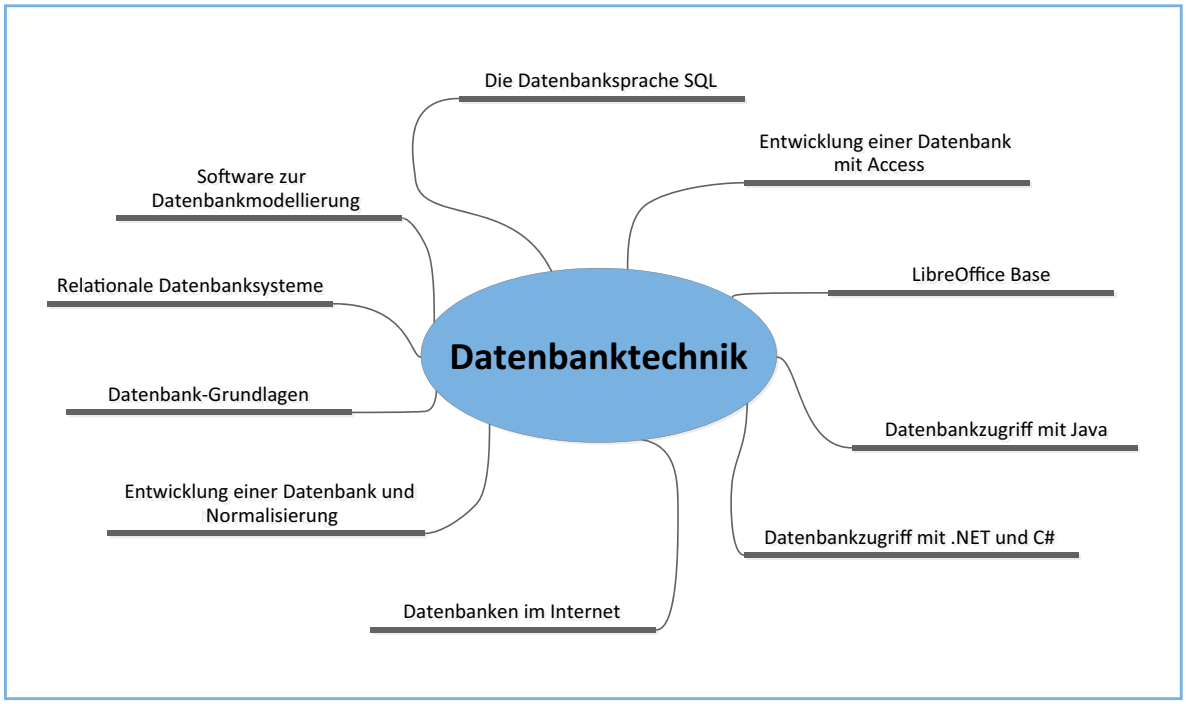
EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung



6. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 · 42781 Haan-Gruiten

Europa-Nr.: 36087

Verfasser:
Elmar Dehler, Laupheim
Dirk Hardy, Oberhausen
Hubert Troßmann, Günzburg

Unter Verwendung von Screenshots aus:

- Visual Studio Community Edition 2019 (Microsoft)
- ACCESS (Microsoft)
- Apache Netbeans IDE 12.6 (The Apache Software Foundation)
- SQLite-Datenbank (sqlite.org)

Kahoot! → kahoot.it
XAMPP → GPL
LearningApps
Firefox → Mozilla Community
mariadb-connector-odbc-3.x.x-win64.msi → MariaDB-Community

6. Auflage 2026

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Korrektur von Druckfehlern identisch sind.

ISBN 978-3-8085-3807-4

Bei Fragen zur Produktsicherheit wenden Sie sich bitte an produktsicherheit@europa-lehrmittel.de.

www.europa-lehrmittel.de
Satz: Typework Layoutsatz & Grafik GmbH, 86153 Augsburg (ab der 6. Auflage)
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagfotos: envfx – stock.adobe.com; Gina Sanders – stock.adobe.com
Druck: Akontext s. r. o., 141 00 Prag 4 (CZ)

Vorwort

Informatik und Informationstechnik beeinflussen nahezu alle gesellschaftlichen Bereiche unseres Lebens. Fast alle beruflichen Aktivitäten und Prozesse werden durch diese Technologien maßgeblich unterstützt oder vollzogen. Datenbanksysteme sind dabei ein zentraler Bestandteil, da von der Verfügbarkeit, Vollständigkeit und Richtigkeit der gespeicherten Daten die Aktionsfähigkeit eines Unternehmens abhängt.

Dieses Buch „**Datenbanken – Entwickeln, Programmieren, Anwenden**“ vermittelt die theoretischen und praktischen Grundlagen zur Planung, Realisierung und Programmierung von Datenbanken mit modernen Softwaresystemen. Großer Wert wird dabei auf die Klärung der Zusammenhänge gelegt.

Als grundlegende Einführung in das gesamte Fachgebiet der Datenbanktechnik ist dieses Buch geeignet für **Schüler** und **Studenten** an **beruflichen Schulen**, **Berufskollegs**, **Berufsakademien**, **Gymnasien**, **Fachhochschulen** und **Universitäten**.

Die einzelnen Kapitel enthalten neben zahlreichen Beispielen auch differenzierte Übungsaufgaben, die zur Erarbeitung und Vertiefung der Themengebiete dienen. Neu sind seit der 4. Auflage Übungsaufgaben und Wiederholungsaufgaben als digitale Inhalte. Die Lösungen zu den Übungsaufgaben können direkt über den Verlag angefordert werden und stehen kostenfrei zur Verfügung.

Die Autoren planen diese Inhalte sukzessive zu erweitern. Gerne können die Leserinnen und Leser hier auch eigene Inhalte auf Grundlage der Buchinhalte entwickeln und über die Lernplattformen weiteren Nutzern zur Verfügung stellen. Allen Nutzern hierbei viel Spaß und größtmöglichen Lernerfolg!

Die **6. Auflage** enthält zahlreiche überarbeitete Texte und Bilder. Das Thema digitale Inhalte mit Kahoot wollen die Autoren aktuell nicht weiter ausbauen, Learningapps werden sukzessive erweitert. Mit Learningapps stellt jeder Lernende den inviduellen Lernerfolg, im eigenen Tempo, sicher. Entgegen dem Gamification-Ansatz kann ohne Druck und Vergleiche mit anderen gelernt und wiederholt werden.

All unseren aufmerksamen Leserinnen und Lesern danken wir für die wertvollen Hinweise, die wir bereits in der **6. Auflage** dieses Buches berücksichtigt haben.

Ihre Meinung zu diesem Buch interessiert uns!

Anregungen und Kritik nehmen wir gerne unter
lektorat@europa-lehrmittel.de entgegen.

Frühjahr 2026 Autoren und Verlag

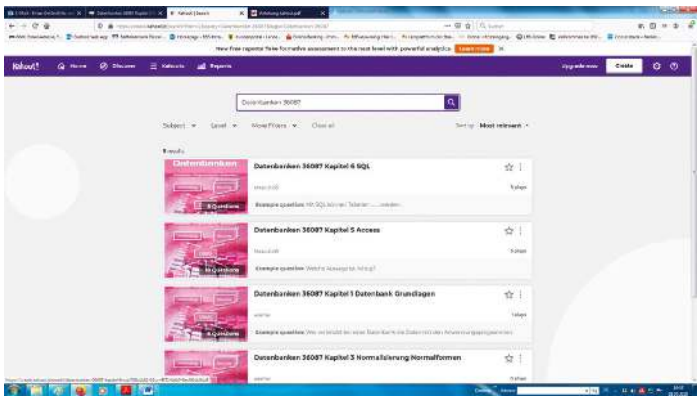
Digitale Inhalte mit Kahoot nutzen:

Kahoot! ist ein Audience Response System, mit dem Spiele oder Umfragen erstellt werden können. Diese werden alleine oder im Klassenverband mit PC oder Smartphone durchgeführt. Um kahoot! als Lehrer zu nutzen, muss eine Registrierung erfolgen. Für jedes Spiel erhält man einen Game-PIN (Zugangscode), der an die Lernenden weitergeben wird. Das Spiel oder die Umfrage wird im Klassenverband über einen Beamer oder Monitor für alle sichtbar eingeblendet. Geantwortet wird alleine oder in Gruppen unter Benutzung eines PCs oder eines Smartphones.

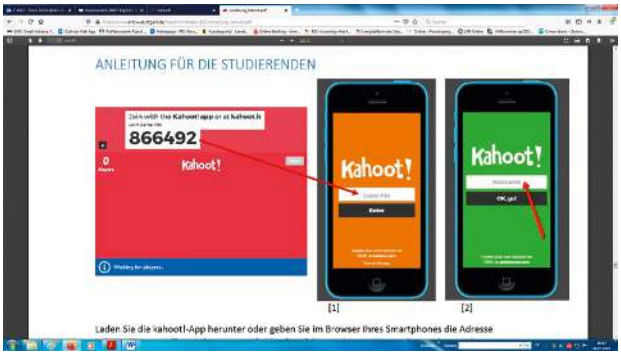
Eine Registrierung der Lernenden ist nicht notwendig.

KAHOOT-REGISTRIERUNG FÜR LEHRENDE

1. Öffnen Sie einen Browser und gehen Sie auf kahoot.com
2. Registrieren Sie sich, indem Sie auf den Button *Sign Up* klicken.
3. Wählen Sie *I want to use Kahoot! as a teacher*
4. Wählen Sie den gewünschten Weg der Registrierung und loggen Sie sich anschließend mit Ihren Zugangsdaten ein.



5. Nach Eingabe weiterer Angaben, erforderlich sind die Wahl eines Benutzernamens und die Angabe der Art der Einrichtung, für die Sie arbeiten, klicken Sie dann auf *Join Kahoot!.*
6. Wählen Sie die gewünschte Version von kahoot!, die Sie verwenden möchten. Die Gratisversion ist ein wenig versteckt.
7. Mit den Suchbegriffen Datenbanken oder der Europanummer 36087 gelangen Sie zum Angebot dieses Lehrwerkes.



Rufen Sie ein Spiel aus dem Angebot durch Anklicken auf und wählen Sie den Spielmodus (Classic = jeder gegen jeden, Team mode = die Mitspielerinnen und Mitspieler treten in Gruppen gegeneinander an). Nicht bei jedem Spieltyp stehen beide Modi zur Verfügung.

Unter Game Options können Sie zahlreiche Einstellungen vornehmen, z.B. Nicknames automatisch zuweisen.

Nach der Auswahl des gewünschten Modus bekommen Sie die Game PIN angezeigt, mit der sich die Mitspielerinnen und Mitspieler einloggen können.

Laden Sie die kahoot!-App herunter oder geben Sie im Browser Ihres Smartphones die Adresse kahoot.it ein. Es öffnet sich eine Seite, auf der sie die PIN des Spiels eingeben und Enter drücken müssen. Danach müssen sie einen Nickname wählen.

Sind alle Spielerinnen und Spieler eingeloggt, kann es losgehen: Durch Anklicken von Start beginnt das Spiel. Ist die Zeit abgelaufen, sehen alle die richtige Antwort und wie im Plenum abgestimmt wurde.

Die Spielerinnen und Spieler sehen auf dem eigenen Bildschirm ob sie richtig oder falsch abgestimmt haben, wie viele Punkte sie damit erreicht haben und auf dem wievielten Platz sie sich momentan befinden.

Die Fragen und Antworten können vom Lehrer jeweils besprochen werden. Über *Get Results* gelangen Sie am Ende des Spieles zur Bestenliste und im Anschluss können Sie Ihre Schüler noch um ein Feedback zum Spiel bitten.

1 Datenbank-Grundlagen

1.1 Einsatz von Datenbanken

Eine **Datenbank** ist eine Sammlung gespeicherter Daten, die untereinander in einer logischen Beziehung stehen und von einem **Datenbankverwaltungssystem** (**DBMS** von Database Management System) verwaltet werden. Die Daten werden z. B. von Anwendungsprogrammen und Benutzern eines Unternehmens verwendet.

Hinweis

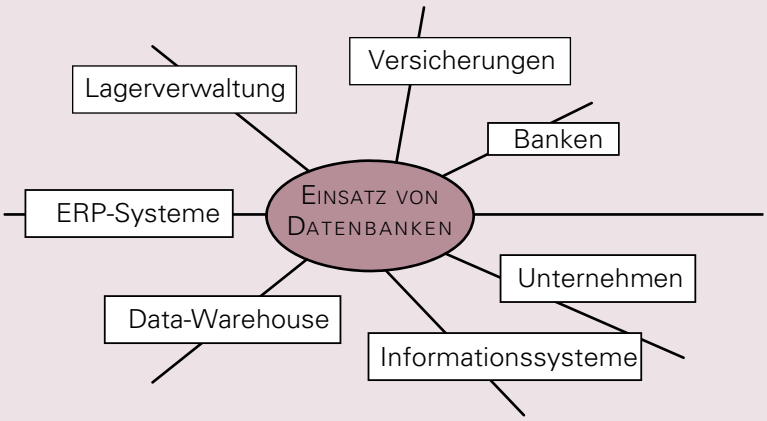
Datenbanken sind strukturierte, zusammengehörende Datenbestände. Diese werden effizient verwaltet und ausgewertet.

1.1.1 Beispiele für den Einsatz von Datenbanken

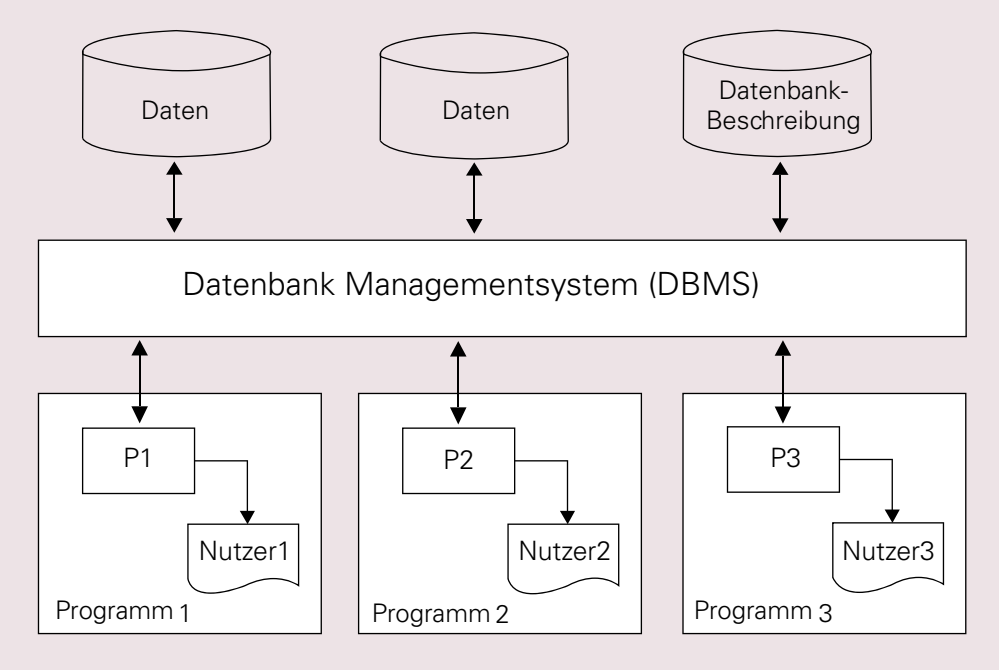
Datenbanken spielen beim Einsatz von Computern häufig eine zentrale Rolle. Die Speicherung von großen Datenmengen ist überall erforderlich, wo Arbeitsabläufe computerunterstützt abgewickelt werden.

Beispiele

- Banken und Versicherungen arbeiten mit Datenbanksystemen. In der Datenbank sind alle Informationen zu Konten, Buchungen und Kunden strukturiert abgelegt. Datenschutz und Datensicherheit haben höchste Priorität.
- Unternehmen jeglicher Größe und Branche arbeiten zur Ressourcenplanung mit ERP-Systemen (von Enterprise Resource Planning). Die Daten z. B. Kundendaten, Mitarbeiterdaten oder Artikeldaten, liegen gespeichert in Datenbanksystemen vor.
- Die automatisierte Lagerverwaltung macht den Einsatz von Datenbanken notwendig. Eine Lagerdatenbank enthält geordnete Informationen zu zahlreichen Lieferanten, Artikeln und deren Bestände.
- Informationssysteme im Internet (z. B. Wikipedia) verwalten ihre Artikel mithilfe von Datenbanken.
- Unternehmen speichern in Data Warehouses (Daten-Warenlager) Daten für die Datenanalyse zur betriebswirtschaftlichen Entscheidungshilfe. Ebenso speichern z. B. Marktforschungsinstitute eigene Daten und Fremddaten zur weiteren Verarbeitung.



Die Anwendungsprogramme, z. B. Software für Lagerhaltung oder Software zur Personalverwaltung, greifen über das DBMS auf die gemeinsamen Daten parallel zu.



1.1.2 Probleme bei der Datenspeicherung mit Datenbanken

Bei der Speicherung von Daten mithilfe von Datenbanken können zahlreiche Probleme auftreten:

- Redundanzen

  Daten werden mehrfach gespeichert, dadurch werden Änderungen aufwendig und der Datenbestand ist fehleranfällig. Dieselben Daten müssen mehrmals an verschiedenen Stellen geändert werden. Wenn z. B. Änderungen von mehrfach gespeicherten Daten nur an einer Stelle vorgenommen werden, sind die Datenbestände fehlerhaft.

- Inkonsistenzen

  Werden Daten von mehreren Benutzern bzw. Programmen zeitgleich bearbeitet und geändert, kann es zu einem inkonsistenten Zustand der Daten kommen. Der Datenzugriff ist nicht synchronisiert. Wird z. B. ein Girokonto von zwei Benutzern zeitgleich bearbeitet, sehen beide den aktuellen Kontostand von 2000 Euro. Hebt nun Benutzer A 1000 Euro ab und speichert diesen Vorgang, zeitgleich zahlt Benutzer B 500 Euro ein und speichert, dann sind im Datenbestand sowohl 1000 Euro als auch 2500 Euro inkonsistent und falsch.

- Datenschutz

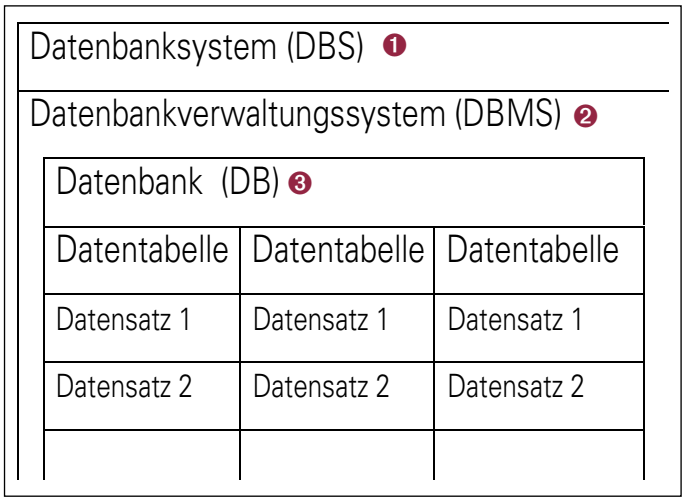
  Lesezugriffe und Schreibzugriffe auf den gesamten Datenbestand sind möglich. Datenschutz kann – abhängig vom verwendeten Betriebssystem – durch Zugriffsrechte oder Verschlüsselung realisiert werden.

- Fehlende Datenunabhängigkeit

  Die Verwaltung der Daten ist meist nur mit einer entsprechenden Anwendungssoftware möglich. Ist eine Änderung der Struktur der Daten erforderlich, muss sowohl die Anwendungssoftware geändert werden, als auch ein Programm zur Umstrukturierung der Dateien erstellt werden. Sollen die gleichen Dateien einer anderen Anwendung ausgewertet werden, muss für diese neue Anwendung ebenfalls eine eigene Datenverwaltung erstellt werden.

Damit der Anwender die Daten übersichtlich und einfach verwalten kann, benötigt er ein Datenbankverwaltungssystem DBMS. Ein **Datenbanksystem (DBS)** besteht somit aus der Kombination von Datenbank (DB) und einem Datenbankverwaltungssystem.

Typische DBMS sind z. B. Microsoft Access, LibreOffice Base, MariaDB/MySQL, Paradox, Oracle und MS SQL-Server.



❶ **DBS** = Datenbanksystem: Datenbanken + DBMS
❷ **DBMS** = Datenbank-Management-System: Software zur Verwaltung, Bearbeitung und Auswertung von Datenbanken
❸ **DB** = Datenbank: strukturierter, vom DBMS verwalteter Datenbestand

Hinweis:

Ein DBS speichert und organisiert die Daten redundanzfrei, mit der nötigen Datensicherheit und einem gewährleisteten Datenschutz. Das DBS ist unabhängig von den Anwendungen, die auf die Daten zugreifen.

Anwendungsprogramme greifen nicht direkt auf die Daten zu, sondern stellen ihre Anforderungen an das Datenbankmanagementsystem. Die Datenbank ist eine Sammlung logisch zusammengehöriger Daten zu einem Sachgebiet, z. B. Kundendaten und Auftragsdaten. Das DBMS stellt die Schnittstelle zwischen der Datenbank und deren Benutzern, z. B. den Anwendungsprogrammen, her. Es gewährt den Zugriff auf die Daten und sorgt dabei für eine zentrale Steuerung und Kontrolle. Das DBMS verwaltet die Benutzer, deren Zugriffe auf die Datenbank und die Zugriffsrechte der Benutzer. Außerdem wird durch das DBMS ein Schutz gegen Hard- und Softwarefehler gewährleistet, sodass beispielsweise bei Programm- oder Systemabstürzen die Daten nicht verloren gehen bzw. wiederhergestellt werden können. Änderungen an der Struktur der Datenbank bedeuten nicht, dass auch die Anwendungsprogramme geändert werden müssen.

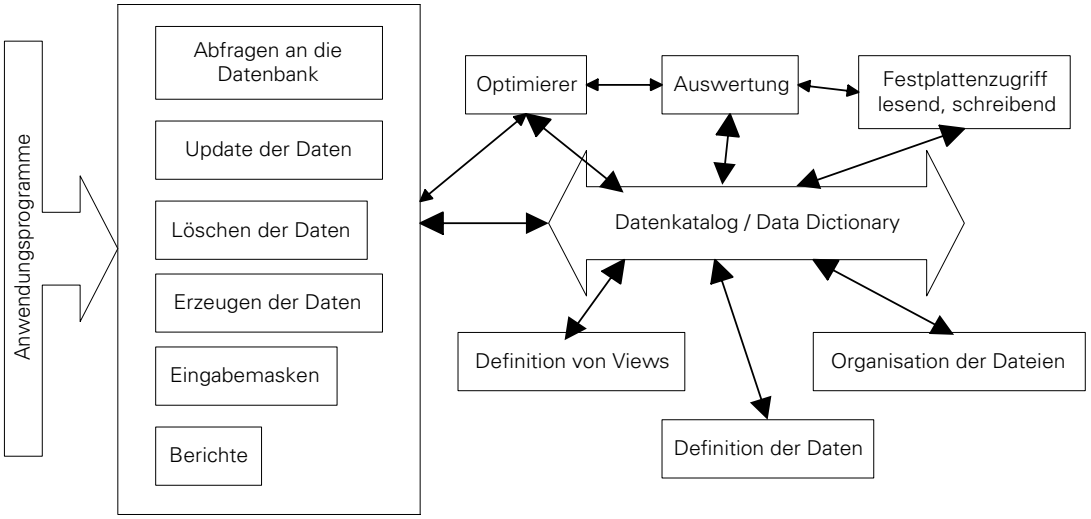
1.1.3 Aufgaben eines DBMS

E. F. Codd (brit. Mathematiker) fasste 1982 die Anforderung an ein DBMS in 9 Punkten zusammen:

1. Datenintegration = einheitliche Verwaltung aller benötigten Daten.
2. Datenoperationen = der Datenbestand ermöglicht das Suchen, das Ändern und das Abspeichern von Daten.
3. Datenkatalog = ein Datenkatalog (Data Dictionary) enthält die Beschreibungen der Datenbank.
4. Benutzersichten = jede Anwendung benötigt unterschiedliche Sichten (Views) auf den Datenbestand.
5. Konsistenzüberwachung = die Überwachung der Datenintegrität sichert die Korrektheit der Daten in der DB.
6. Zugriffskontrolle = Zugriffe auf den Datenbestand können kontrolliert werden und gegebenenfalls auch verhindert werden.
7. Transaktionen = Änderungen an der DB können als Einheiten zusammengefasst werden.
8. Synchronisation = gemeinsam benutzte Daten müssen bei konkurrierenden Transaktionen synchronisiert werden.
9. Datensicherung = ermöglicht die Wiederherstellung des Datenbestandes nach Konflikten, z. B. Systemabsturz.

Der Unterschied zwischen einem Datenbanksystem und einer Ansammlung einzelner Dateien besteht darin, dass in einem Datenbanksystem die Daten vom Datenbankverwal-

tungssystem DBMS zentral verwaltet werden. Die Anwendungsprogramme greifen über das DBMS auf die gemeinsamen Daten parallel zu.



Architektur eines DBMS

Das **Data Dictionary** (Datenkatalog) beschreibt, wie auf der internen Ebene die Datenspeicherung realisiert wird. Es ist der zentrale Katalog aller für die Datenhaltung wichtigen Informationen. Im Einzelnen ergeben sich folgende Komponenten:

- Definition der Dateiorganisationen, Definition der Zugriffspfade auf die Dateien,
- Konzeptuelle Datendefinition, Definition von Benutzersichten, Optimierung der Datenbankzugriffe,
- Auswertung der Abfragen und Änderungen und Steuerung der Festplattenzugriffe.

Einzelne **Transaktionen** sind in sich abgeschlossene Zugriffe auf den Datenbestand. Beispielsweise werden bei einem Buchungsvorgang von Konto A 100 € abgehoben und auf Konto B eingezahlt. Transaktionen werden in einem **Logbuch** abgespeichert. Das Logbuch enthält Informationen zum Beginn und Ende einer Transaktion und über die bearbeiteten Datenbestände vor und nach der Transaktion. Anhand des Logbuches können Transaktionen nachvollzogen und evtl. rückgängig gemacht werden.

Eine Transaktions-Managementsoftware ermöglicht den gleichzeitigen Zugriff auf Daten. Parallel ablaufende Transaktionen werden synchronisiert, um die Integrität des Datenbestandes zu gewährleisten.

Beispiel:

Im Linienflugzeug von Stuttgart nach Berlin wird ein Sitzplatz gebucht. Kunde A in Ulm loggt sich in die Buchungs-Software zeitgleich wie Kunde B in Stuttgart ein und bekommt denselben freien Platz angezeigt. Wenn nun Kunde A den Sitzplatz bucht, entspricht das Anklicken des Buchungsbuttons einem schreibenden Zugriff auf den Datensatz. In diesem Moment ist der Datensatz für Kunde A exklusiv für eine Transaktion reserviert. Findet die Buchung und damit die Transaktion einen erfolgreichen Abschluss, kann Kunde B denselben Platz nicht mehr buchen. Kunde B sieht den Sitzplatz anschließend als belegt.

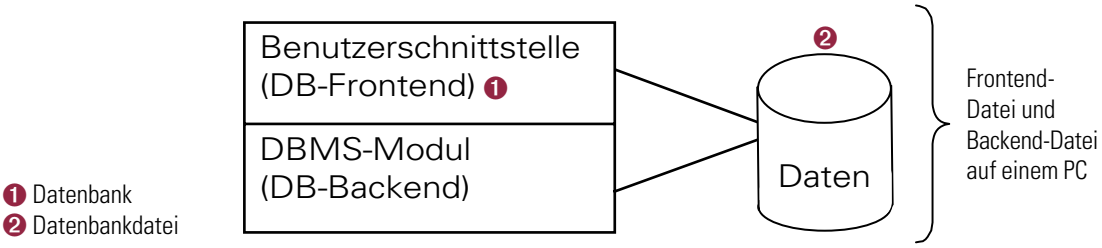
Es ergeben sich folgende Vorteile:

- Alle Programme arbeiten mit der gleichen Datenbasis, d. h., die Aktualität der Daten ist für alle dieselbe.
- Einmalige Speicherung der Daten für alle Anwendungen.
- Unabhängiger gleichzeitiger Zugriff auf gemeinsame Daten unter zentraler Verwaltung.

1.2 Systemarchitekturen

1.2.1 Desktop Datenbanken für einfache Anwendungen (Einbenutzerbetrieb)

Bei einer Desktop Datenbank läuft das Datenbankverwaltungssystem, z. B. Access oder Base, mit der jeweiligen Datenbank und der Datenbankanwendung auf dem PC des Anwenders.

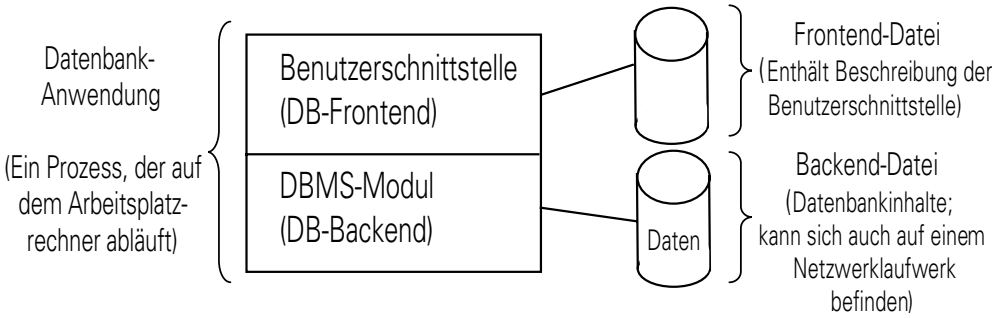


Desktop Datenbank für Einbenutzerbetrieb

1.2.2 Desktop Datenbanken für wenige Benutzer (Mehrbenutzerbetrieb)

Befinden sich die Datenbankinhalte (**Backend-Datei**) auf einem Netzlaufwerk im Intranet oder Internet, so können mehrere Benutzer parallel auf die Datenbankinhalte zugreifen.

Alle Daten müssen zur Verarbeitung, z. B. zum Suchen oder Sortieren, über das Netzwerk transportiert werden.

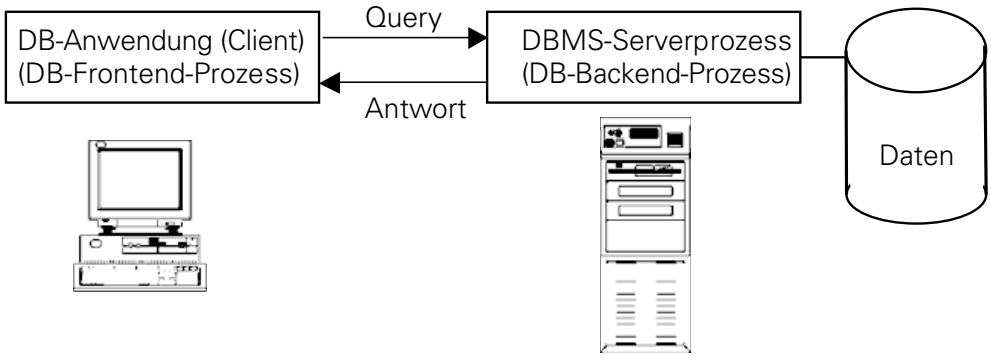


Desktop Datenbank für mehrere Benutzer

1.2.3 Client/Server-Datenbanken

Das Datenbankverwaltungssystem läuft auf einem Server-PC im Netzwerk und hat den exklusiven Zugriff auf die Datenbankdateien. Bei einem relationalen Datenbank-Server spricht man auch von einem **SQL-Server** (von **Structured Query Language** = strukturierte Abfragesprache). Client-Programme rufen Daten ab und speichern diese.

Nur die Abfrage (SQL-Anweisung, Query) und die Antwort müssen über das Netzwerk transportiert werden. Die Datenbankanwendung kann auf dem Client oder auf dem Server laufen oder sich auf beide verteilen.



Client / Server Datenbank

1.3 Datenbankmodelle

Man unterscheidet meist 5 **Datenbankmodelle**. Dies sind die relationalen Datenbanken, die objektorientierten, die hierarchischen Datenbanken, die netzwerkartigen Datenbanken und NoSQL-Datenbanken. Die Unterschiede dieser 5 Modelle liegen in der Art des logischen Aufbaus der Datenbank.

1.3.1 Relationale Datenbanken

Eine **relationale Datenbank** besteht ausschließlich aus Tabellen. Ein Zugriff erfolgt immer über diese Tabellen. Da leicht neue Tabellen hinzugefügt oder gelöscht werden können, sind spätere Änderungen des logischen Datenbankaufbaus relativ leicht möglich. Zugriffe auf Tabellen sind einfach zu programmieren, was zu der großen Beliebtheit dieses Datenbankmodells führte.

Die Zusammenhänge zwischen den einzelnen Tabellen werden über Beziehungen hergestellt. Diese Beziehungen sind in den Tabellen mit abgespeichert. Der Aufbau von Datenbeständen über Tabellen und ihre Beziehungen zueinander sind mathematisch fundiert (Relationenalgebra).

Die relationalen Datenbanken besitzen aber auch Nachteile: Zugriffe erfolgen oft über mehrere Tabellen, was längere Laufzeiten und eine hohe Anzahl von Ein-/Ausgaben zur Folge haben kann.

1.3.2 Objektorientierte Datenbanken

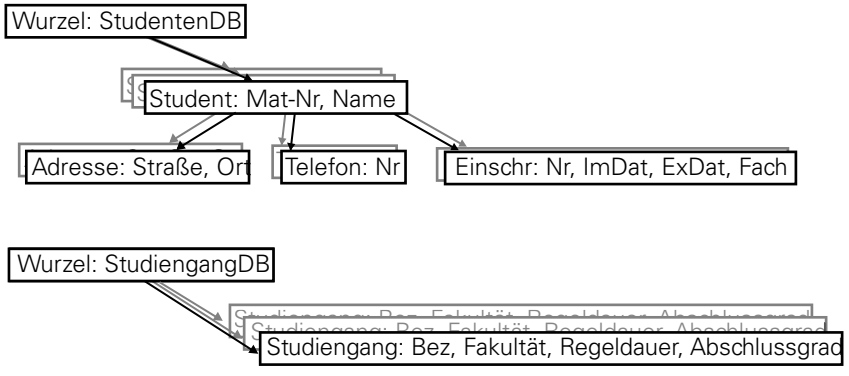
Eine **objektorientierte Datenbank** besteht ausschließlich aus Objekten. Ein Objekt ist entweder ein realer Gegenstand, z. B. ein Flugzeug, eine Person oder ganz allgemein ein abstrakter Gegenstand, etwa eine Adresse, eine Rechnung, ein Vorgang oder eine Abteilung einer Firma.

Da viele Objekte auch in Tabellenform gespeichert werden können, werden objektorientierte Datenbanken häufig als eine Erweiterung relationaler Datenbanken angesehen. Dies trifft allerdings nur teilweise zu. Schließlich gehören zu einer objektorientierten Datenbank auch objektorientierte Ansätze wie Klassen, Datenkapselungen oder Vererbungen.

Objektorientierte und objektrelationale Datenbanken haben einen komplexeren Aufbau als relationale Datenbanken (fast beliebige Objekte statt einfacher Tabellen). Als Konsequenz müssen Datenbank-Designer und Anwendungsprogrammierer einen höheren Aufwand in Entwurf und Programmierung investieren. Auch die interne Verwaltung der Datenbank ist umfangreicher. Als Vorteil erhält man insbesondere bei technischen und multimedialen Anwendungen einen anschaulicheren Aufbau (komplexe Objekte müssen nicht zwangsweise auf Tabellen abgebildet werden). Dies kann erhebliche Laufzeitvorteile nach sich ziehen.

1.3.3 Hierarchische und netzwerkartige Datenbanken

Die ältesten Datenbanken sind **hierarchische Datenbanken**, eine Weiterentwicklung der konventionellen Dateiorganisation, wie sie beim PC intern genutzt wird. Der logische Aufbau dieser Datenbanken entspricht einer umgedrehten Baumstruktur. Der Zugriff erfolgt immer über die Wurzel in Richtung des gewünschten Knotens. Ein Objekt kann dabei stets nur zu einer Wurzel zugeordnet werden, was als Monohierarchie bezeichnet wird. Dies gewährleistet geringste Redundanz, da direkt über die Baumstruktur zugegriffen wird, und garantiert kürzeste Zugriffszeiten.



Beispiel für eine hierarchische Datenbank

Hierarchische Datenbanken verknüpfen die Daten über feste Beziehungen, wobei ein Datensatz auf den nächsten verweist.

Die Aufgabe von Datenbanksystemen, die Realwelt zu modellieren, ist mit dem hierarchischen Modell nur sehr begrenzt möglich. Die Beschränkung auf die Darstellung von Ober- und Unterbegriffsbeziehungen ist für eine realistische Abbildung von Alltagssituationen, in denen oft nur wenige rein hierarchische Beziehungen existieren, nicht geeignet.

Beispiel für ein hierarchisches Datenbanksystem ist IMS von IBM.

Bei **netzwerkartigen Datenbanken** besteht der logische Aufbau aus Daten, die nicht mehr rein hierarchisch, sondern über ein beliebig aufgebautes Netz miteinander in Verbindung stehen. Dies erhöht die Flexibilität erheblich, allerdings erhöht sich die Komplexität des Aufbaus.

Beide Modelle genügen den heutigen Anforderungen nicht mehr.

Die wichtigsten Vertreter netzwerkartiger Datenbanken sind IDMS (Computer Associates) und UDS (Siemens-Nixdorf).

1.3.4 NoSQL-Datenbanken

NoSQL-Datenbanken (von: *Not only SQL* = nicht nur SQL) bezeichnet Datenbanken, die einen nicht-relationalen Ansatz verfolgen. NoSQL Datenbanken verwenden keine Tabellen und versuchen Verbindungen zwischen den Daten (Joins) weitestgehend zu vermeiden.

Typische NoSQL Datenbanken sind: Cassandra von Apache, MongoDB oder Redis.

NoSQL-Datenbanken wurden zuerst für einfache Open-Source-Datenbanken verwendet, die keine SQL-Zugriffsmöglichkeit bereitstellen. Nicht relationale, verteilte Datenspeichersysteme werden häufig unter dem Begriff NoSQL aufgeführt. Die Nachteile des relationalen Datenbankmodelles, z.B. Leistungsprobleme bei Indizierung großer Datenbestände oder Leistungsprobleme bei hohen Datenanforderungen/Datenänderungen können mit NoSQL-Systemen reduziert werden.

Die leistungsoptimierten NoSQL-Architekturen bieten meist nur geringe Konsistenzforderungen der Daten. Auch Transaktionen sind häufig nur auf wenige Datensätze eingeschränkt

Typische NoSQL-Datenbanksysteme unterstützen verteilte Datenbanken mit redundanter Datenhaltung auf vielen Servern. Die Systeme können so einfach erweitert werden und Ausfälle einzelner Server überstehen.

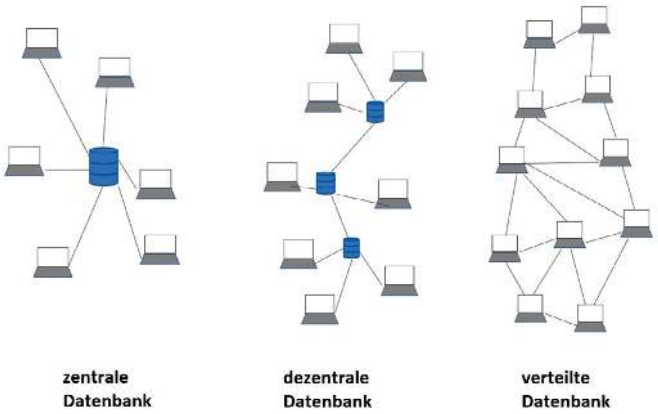
Vorteile und Nachteile von Datenbankmodellen

Modell	Vorteile	Nachteile
Relationale Datenbanken	leichte Änderbarkeit des Datenbankaufbaus, mathematisch fundiert, leicht programmierbar, einfache Verwaltung.	häufig viele Ein-/Ausgaben notwendig, erfordert bei großen Datenbeständen eine hohe Rechnerleistung.
Objektorientierte Datenbanken	universeller, objektorientierter Aufbau, noch relativ einfach programmierbar, einfach zu verwalten.	relativ viele Ein-/Ausgaben notwendig, komplexer Aufbau, erfordert eine relativ hohe Rechnerleistung.
Hierarchische und netzwerkartige Datenbanken	kurze Zugriffszeiten, geringe Redundanz.	Strukturänderung kaum möglich, komplexe Programmierung.
NoSQL-Datenbanken	Leistungsoptimiert für große Datenbestände und zahlreiche, zeitgleiche Zugriffe lesend und schreibend.	geringe oder keine Datenkonsistenz, hoher Aufwand für kontrollierte Transaktionen.

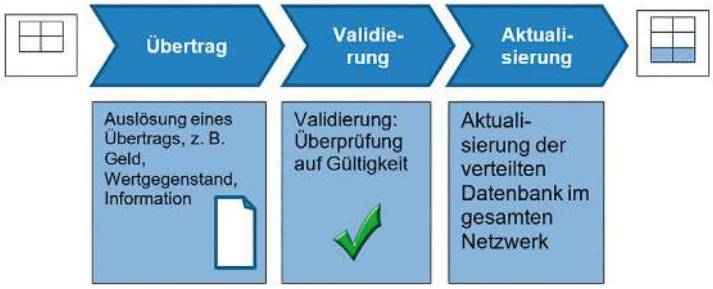
1.3.5 Distributed-Ledger-Technologie (DLT)

1.3.5.1 Grundlagen

Als Distributed Ledger (verteiltes Kontenbuch) wird eine verteilte Datenbank bezeichnet. In einer verteilten Datenbank sind die Daten auf jedem teilnehmenden Rechner im weltweiten Netz gespeichert.



Neue Datensätze können von jedem Teilnehmer selbst hinzugefügt werden, jeder Teilnehmer hat Schreibrechte, z. B. für die Bestätigung eines Wertübertrags. Ein anschließender Aktualisierungsvorgang stellt allen Teilnehmern den aktuellen Datenbestand zur Verfügung.



Erzeugt ein Teilnehmer durch eine neue Transaktion, z. B. Übertragung eines Geldbetrags, einen neuen Datensatz, so wird dieser zunächst auf seine Gültigkeit geprüft. Die Prüfung geschieht durch einen Konsensmechanismus.

Dieser Mechanismus legt fest, welche Bedingungen erfüllt werden müssen, um dem Kontenbuch (ledger) eine neue gültige Transaktion hinzuzufügen.

Konsensmechanismen zur Gültigkeitsprüfung

Name	Beschreibung
Proof-of-Work = Arbeitsnachweis	Arbeitsnachweis durch den Einsatz von Rechenaufwand eines Teilnehmers.
Proof-of-Stake = Anteilsnachweis	Anteilsnachweis durch Gewichtung der einzelnen Teilnehmer aus Teilnahmedauer und Vermögen („Stake")
PBFT (Practical Byzantine Fault Tolerance, = Praktische byzantinische Fehlertoleranz)	Einigung einer Mindestanzahl von Teilnehmern auf die Gültigkeit einer Transaktion

Mit **Proof-of-Work** wird Arbeitsaufwand eines Teilnehmers gewichtet, z. B. um ein Zufallswort, eine „Nonce", zu finden. Damit soll der Anwender abgehalten werden, einen Dienst übermäßig in Anspruch zu nehmen oder missbräuchlich zu verwenden.

Bei der Kryptowährung Bitcoin ist dieser Arbeitsnachweis das zeit- und energieintensive „Mining".

Proof-of-Stake gewichtet die Teilnahmedauer eines Teilnehmers am Netzwerk und sein Vermögen, Transaktionen durchzuführen.

PBFT erfordert eine Mindestanzahl an Teilnehmern für die Gültigkeit einer Transaktion.

Offene und geschlossene Distributed Ledgers

Offene und geschlossene Netzwerke von Distributed Ledgers unterscheiden sich durch die Möglichkeit des Zugangs der Teilnehmer. In offenen (unpermissioned = unregistrierten) Ledgers hat jeder nach einfacher Authentifizierung Zugang. In geschlossenen (permissioned = auf einen angemeldeten Teilnehmerkreis beschränkten) Netzwerken werden die Teilnehmer registriert und müssen bestimmte Voraussetzungen für den Zugang zum Kontenbuch erfüllen. Zum Beispiel kann ein geschlossenes Distributed Ledger für alle Kunden oder Lieferanten eingerichtet werden.

In offenen Netzwerken (unpermissioned) mit einem unbeschränkten Teilnehmerkreis ist der Konsensmechanismen Proof-of-Work üblich. Dies geschieht, indem der Teilnehmer eine bestimmte Rechenleistung oder eine Zeitverzögerung erlauben muss.

Proof-of-Work wird z. B. bei der Kryptowährung Bitcoin angewendet. Auch E-Mail-Provider verwenden teilweise dieses Konzept, indem sie den Versand einer E-Mail um wenige Sekunden verzögern. Dies stört einen normalen Anwender kaum, erschwert aber den Versand von Spam-Mails erheblich.

Proof-of-Stake wird bei permissioned ledgers angewendet (z. B. geplante Anwendung bei der Kryptowährung Ethereum). Dieser Mechanismus benötigt weniger Rechenkapazität, da die Teilnehmer bereits durch eine mehrstufige Anmeldung legitimiert sind.

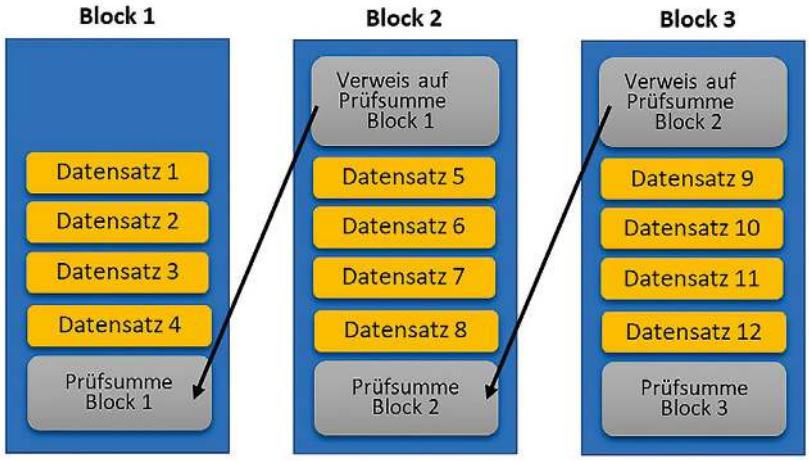
Vorteile der Distributed-Ledger-Technologie

Gegenüber herkömmlicher Datenverwaltung zeigt die DLT folgende Vorteile:

- Transparenz und Unveränderbarkeit: Jeder Teilnehmer kann im verteilten Kontenbuch die gesamten Datenveränderungen einsehen. Dadurch werden die Transaktionen für alle nachvollziehbar.
- Transaktionen können ohne vorgesetzte Zentralinstanzen abgewickelt werden (P2P = peer to peer).
- Sicherheit: auch bei Ausfall eines Teils des Netzwerks sind die gesamten Daten vorhanden.

1.3.5.2 Blockchain

Blockchain (= Blockkette) basiert auf dem Konzept der Distributed-Ledger-Technologie (DLT). Eine Blockchain ist eine stetig erweiterbare Liste von Datensätzen, die zu Blöcken verbunden werden. Diese Blöcke werden durch kryptografische Verfahren miteinander verkettet. Dazu wird am Ende eines jeden Blocks eine Prüfsumme gebildet. Der Beginn des nächsten Blocks enthält vor dem ersten Datensatz einen Verweis auf die Prüfsumme des vorhergehenden Blocks.



Bei einer Blockchain können anders als bei einer relationalen Datenbank (rDB) Daten nur hinzugefügt, aber nicht gelöscht werden.

Das Löschen von Daten würde aufgrund der blockweisen Prüfsummen durch andere Teilnehmer am Netzwerk sofort entdeckt und rückgängig gemacht werden.

Vergleich Relationale Datenbank und Blockchain

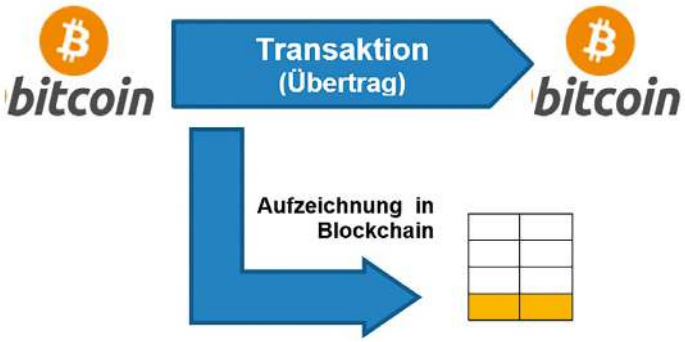
Relationale Datenbank (rDB)	Blockchain
Daten werden in Tabellen gespeichert.	Daten werden in Blöcken zusammengefasst.
Daten sind veränderbar.	Daten können nur hinzugefügt werden, nicht gelöscht werden.
Zentraler Server verwaltet den gesamten Datenbestand	Einträge werden von jedem Teilnehmer ausgeführt. Eintrag erfolgt nach zusätzlicher Validierung.
Kopie der Datenbank ist z. B. als Backup möglich.	Jeder Teilnehmer besitzt eine Kopie des gesamten Datenbestandes.
Prüfsummen werden für einzelne Datensätze generiert, nicht aber für die gesamte Datenbank.	Sowohl einzelne Datensätze als auch Datenblöcke werden mit kryptografischen Prüfsummen versehen. Verfälschungen der Daten sind nahezu unmöglich.

1.3.5.3 Kryptowährung Bitcoin

Mit Bitcoin (= digitale Münze) wird sowohl ein Zahlungssystem als auch eine Geldeinheit bezeichnet. Es wurde im Jahre 2009 auf der Grundlage des Blockchain-Konzepts entwickelt.

Der Besitz von Bitcoin wird durch den Besitz von kryptographischen Schlüsseln nachgewiesen, die auf dem Client in der Wallet (elektronische Geldbörse) aufbewahrt werden. Die persönliche Wallet muss gegen Verlust und Diebstahl durch Schadsoftware geschützt werden. Online-Dienste bieten die Verwaltung der persönlichen Wallet gegen Gebühren an.

Für Zahlungen (Transaktionen) ist eine Wallet-Software notwendig, z. B. Bitcoin Core. Transaktionen finden unter pseudonymen (verschleierten) Adressen statt, die von der Software generiert werden. Das Ergebnis der Transaktion wird von Netzwerk-Clients als Datensatz an die Blockchain angehängt.



Mining

Alle zehn Minuten werden im Blockchain-Netzwerk durch Clients neue Blöcke mit Daten der neuen Transaktionen geschaffen. Dazu ist erhebliche Rechnerleistung der Bitcoin-Teilnehmer aufzuwenden. Als Preis für diese Rechnerleistung erhalten diese Miners (= Schürfer) die Gebühren für die verbuchten Transaktionen als neue Bitcoin-Einheiten. Dieser Vorgang wird als Mining (= schürfen) bezeichnet.

Durch die stetig anwachsende Zahl an Buchungen sind die kryptographischen Berechnungen zunehmend energieintensiv. 2021 wurden dafür nach Schätzungen der Universität Cambridge bereits etwa 120 TWh aufgewendet (zum Vergleich: Schweden 2021 geschätzt 131 TWh).

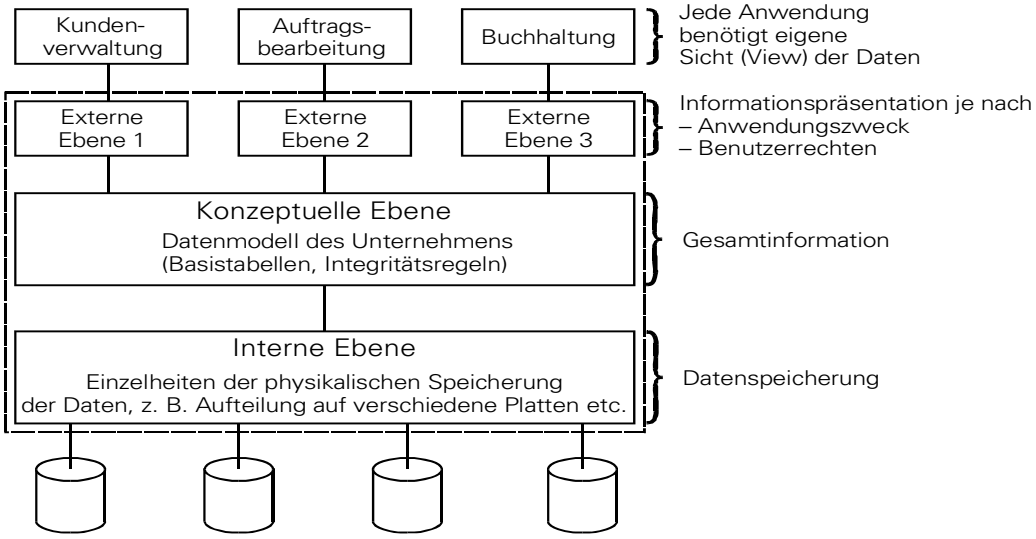
1.4 Architektur eines Datenbankmanagementsystems DBMS

Wichtige Merkmale des bisher beschriebenen Datenbankansatzes sind:

- Isolierung von Programmen und Daten (Programm/Daten- und Programm/Operationen-Unabhängigkeit),
- Unterstützung mehrerer Benutzersichten (Views) und
- Verwendung eines Katalogs zum Speichern der Datenbankbeschreibung (Schema).

1.4.1 Die Drei-Ebenen-Architektur (Drei-Schichten-Architektur)

Die Drei-Ebenen-Architektur trennt die Benutzeranwendungen und die Details der Speicherung (physische Eigenschaften) voneinander.



Folgende drei Ebenen sind definiert:

1. Interne Ebene
Die interne Ebene beschreibt die physikalischen Speicherstrukturen der Datenbank. Das interne Schema verwendet ein physisches Datenmodell und beschreibt die Details des Datenzugriffs bei der Datenspeicherung, Zugriffspfade für die Datenbank und die Dateiorganisation.

2. Konzeptuelle Ebene
Die konzeptuelle Ebene beschreibt die Struktur der gesamten Datenbank für alle Nutzer der Datenbank. Das konzeptuelle Schema verbirgt die Details der physischen Speicherstrukturen und konzentriert sich auf die Beschreibung von Entitäten, Datentypen, Beziehungen, Benutzeroperationen und Einschränkungen. Auf dieser Ebene wird ein von den Speicherstrukturen unabhängiges, logisches Datenmodell benutzt.

3. Externe Ebene oder View-Ebene
Die externe Ebene beinhaltet die externen Benutzersichten (Views). Jede View beschreibt den Teil der Datenbank, an dem eine bestimmte Benutzergruppe interessiert ist und verbirgt die übrigen Daten der Datenbank vor dieser Benutzergruppe. Auf dieser Ebene kann ebenfalls ein von den Speicherstrukturen abstrahierendes (logisches) Datenmodell benutzt werden.

Die Ebenen-Architektur, die praktisch allen modernen Datenbanksystemen zugrunde liegt, trägt wesentlich zur Unabhängigkeit zwischen den Anwendungsprogrammen und der internen Datenstruktur bei.

Änderungen, welche die physikalische Speicherung der Daten betreffen, werden durch das DBMS für die Schichten oberhalb des Internen Schemas weitgehend unsichtbar gemacht. Änderungen am konzeptuellen Schema (die in der Praxis möglichst selten vorgenommen werden sollten), können durch die Informationspräsentation mithilfe der externen Schemata vielfach ebenfalls für die Anwendungsprogramme transparent erfolgen.

Hinweis:

Die Basistabellen werden im Allgemeinen nur vom Datenbankadministrator bearbeitet.

Die externen Ebenen sorgen vor allem dafür, dass die einzelnen Anwendungen nur die Informationen bekommen, die sie haben müssen und haben dürfen.

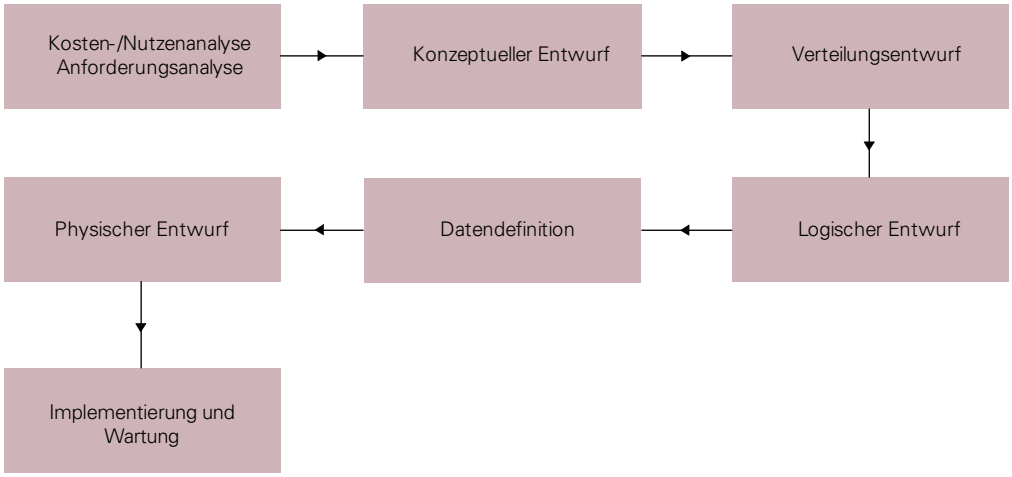
Hinweis:

Anwender bzw. Anwendungsprogramme sehen die DB-Inhalte immer nur aus dem Blickwinkel ihrer externen Ebene: „externe Sichtweise“ (external view).

1.5 Phasen des Datenbankentwurfs

Zur Realisierung einer Datenbank-Anwendung können verschiedene Phasen definiert werden. Die Phasen des Datenbankentwurfs werden auch in anderen Softwareentwicklungen durchlaufen.

1. Sammeln und Analysieren der Anforderungen an die neue Datenbank.
2. Systemunabhängiger Entwurf der Datenbank nach Anwendungsfunktionen.
3. Bei verteilten Datenbanken systemunabhängiger Entwurf des verteilten Systems.
4. Auswahl eines Datenbankmodells und Abbildung des konzeptuellen Entwurfs auf das Datenbankmodell.
5. Datendefinition, d. h. Codierung und Programmierung mithilfe eines DBMS, Definition der Benutzersichten.
6. Definition der Zugriffstrukturen im physischen Entwurf.
7. Installation der Datenbank-Anwendung, Anpassung, Testphase.



1.6 Aufgaben zu Kapitel 1

1. Erläutern Sie, was man unter einem Datenbankmanagementsystem versteht.
2. Was ist ein Datenbanksystem?
3. Nennen Sie Beispiele für den Einsatz von Datenbanken.
4. Welche Probleme treten beim Einsatz von Datenbanken auf?
5. Beschreiben Sie Inkonsistenzen bei Datenbanken am Beispiel Geldabhebung am Bankomat.
6. Welche Aufgaben hat ein DBMS?
7. Welche Aufgaben werden in der externen Ebene eines DBMS erfüllt?
8. Beschreiben Sie eine Desktop-Datenbank für den Einbenutzerbetrieb.
9. Beschreiben Sie eine Desktop-Datenbank für mehrere Benutzer.