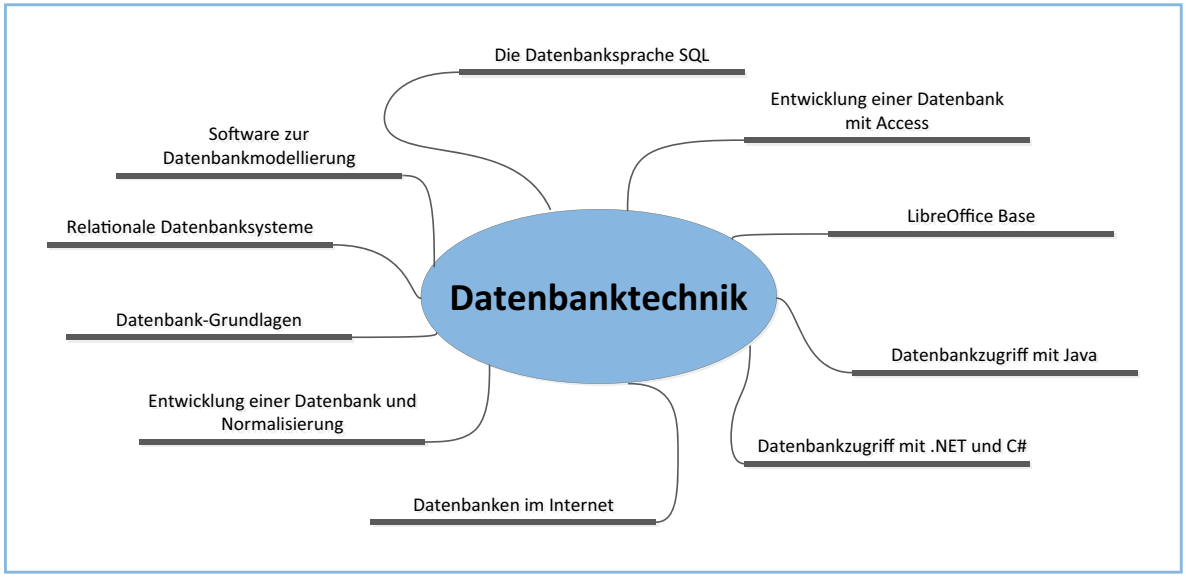
EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung



3. 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, Burtenbach

3. Auflage 2018

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-3689-6

Satz: Reemers Publishing Services GmbH, 47799 Krefeld
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagfotos: envfx-fotolia.com; Gina Sanders-fotolia.com
Druck: Medienhaus Plump GmbH, 53619 Rheinbreitbach

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.

All unseren aufmerksamen Leserinnen und Lesern danken wir für die wertvollen Hinweise, die wir in der **3. Auflage** dieses Buches berücksichtigt haben.

Ihre Meinung zu diesem Buch interessiert uns!

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

Herbst 2018 Autoren und Verlag

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

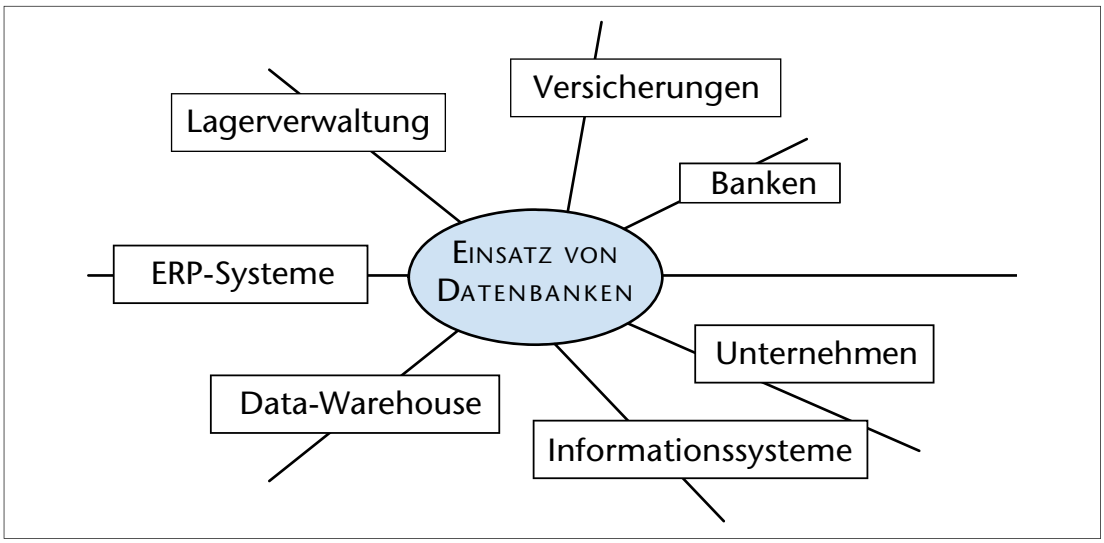
Datenbanken sind logisch zusammengehörende Datenbestände.

1.1.1 Beispiele für den Einsatz von Datenbanken

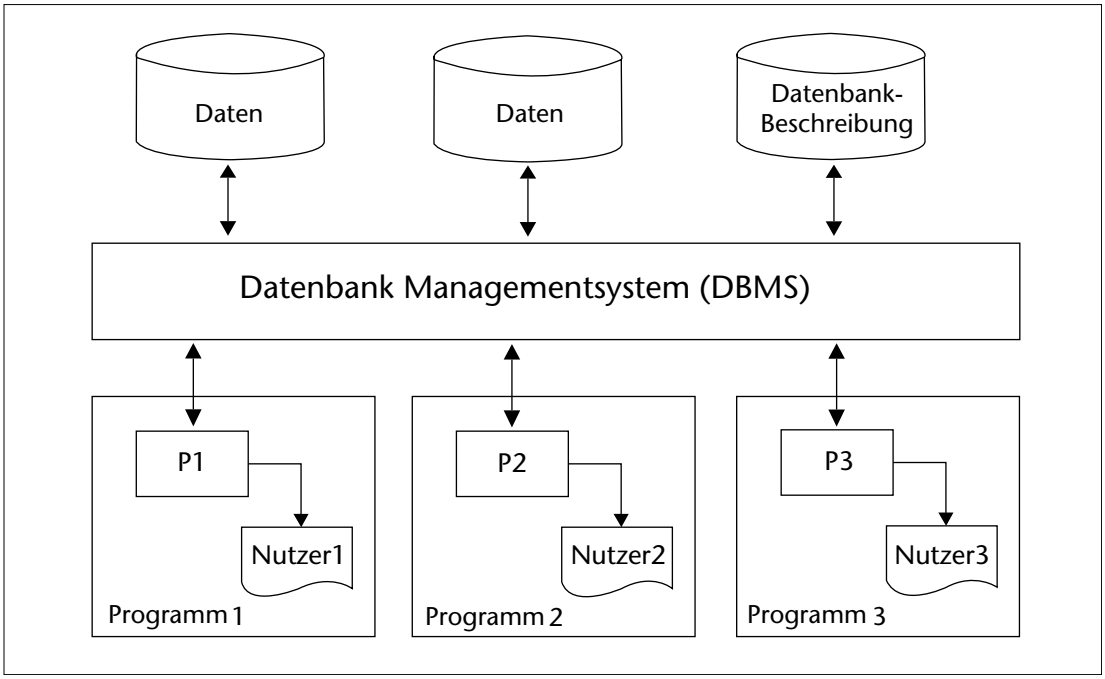
Datenbanken spielen beim Einsatz von Computern häufig eine zentrale Rolle. Die Speicherung von großen Datenmengen ist überall erforderlich, wo Arbeitsabläufe computerunterstützt abgewickelt werden.

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

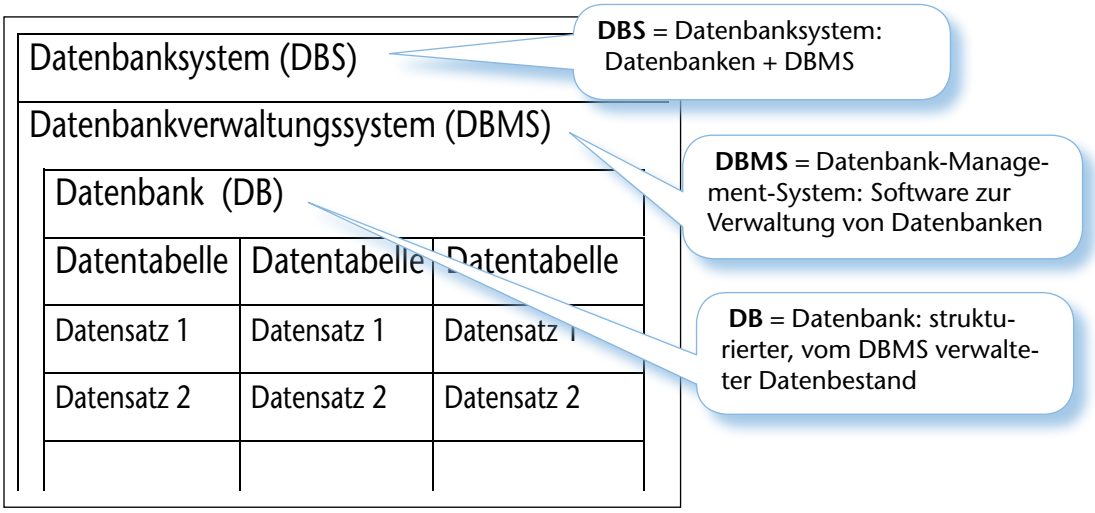
  Lesezugriffe und Schreibzugriffe auf den gesamten Datenbestand sind möglich. Datenschutz kann – abhängig vom verwendeten Betriebssystem – durch Zugriffsrechte oder Verschlüsselung realisiert werden.

- 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, MySQL, Paradox, Oracle und MS SQL-Server.



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.

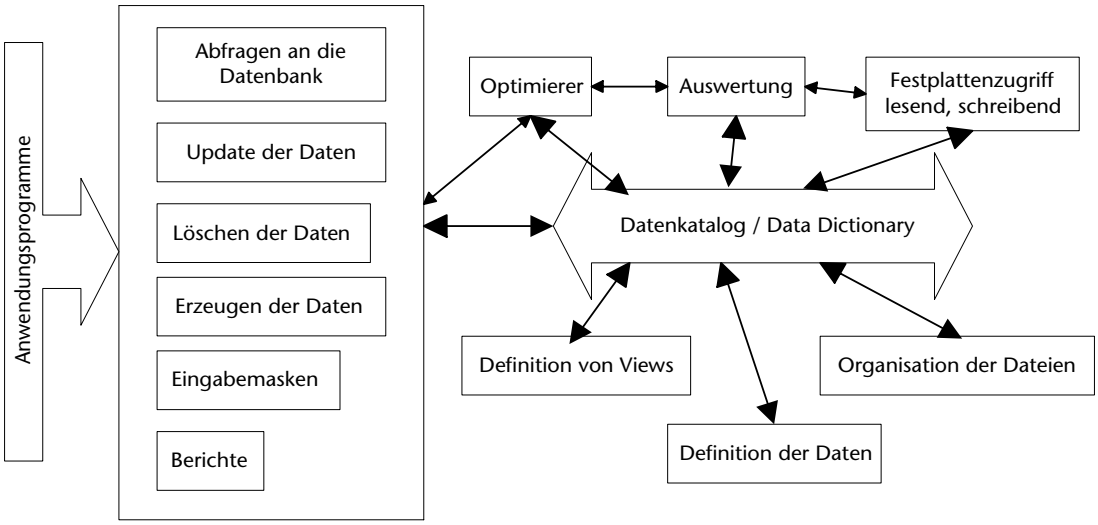
Anwendungsprogramme greifen nicht direkt auf die Daten zu, sondern stellen ihre Anforderungen an das Datenbankmanagementsystem. Die Datenbank ist eine Sammlung logisch zusammengehöriger Daten zu einem Sachgebiet, z. B. Kundendaten und Auftragsdaten. Das DBMS stellt die Schnittstelle zwischen der Datenbank 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 Datenbankverwaltungssystem 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.

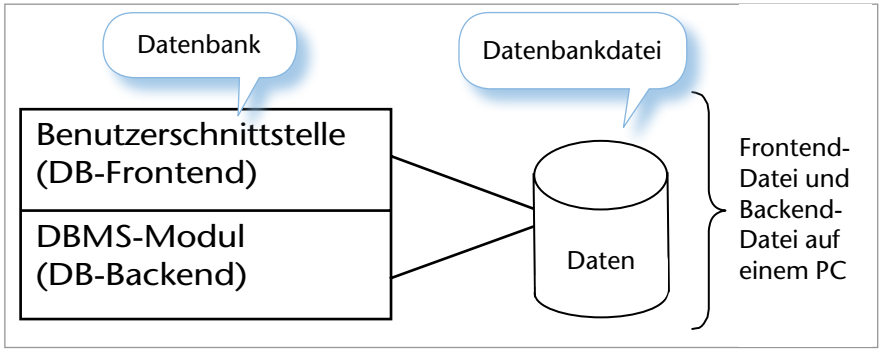
Es ergeben sich folgende Vorteile:

- 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.

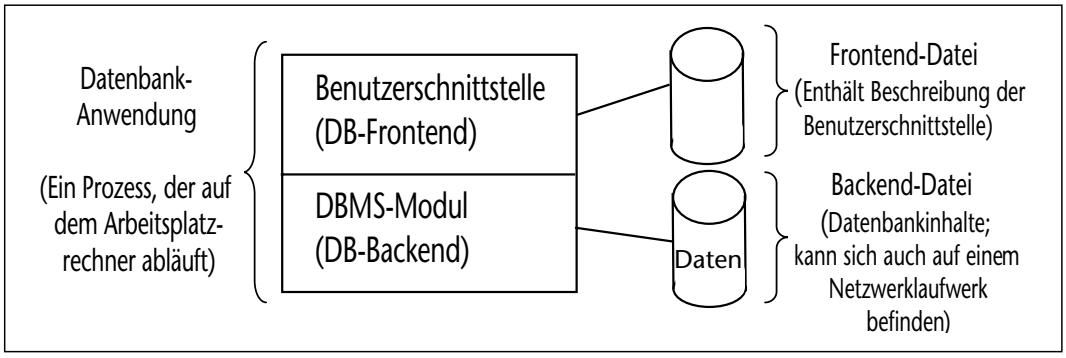


Desktop Datenbank für Einbenutzerbetrieb

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.

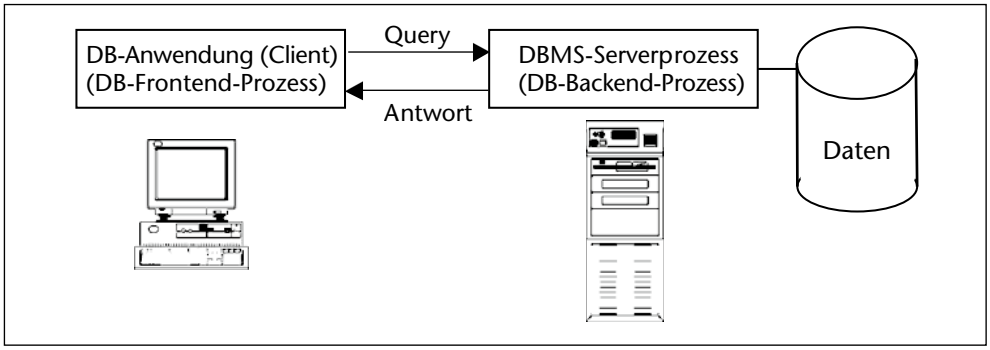


Desktop Datenbank für mehrere Benutzer

1.2.3 Client/Server-Datenbanken

Das Datenbankverwaltungssystem läuft auf einem Server-PC im Netz 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 vier **Datenbankmodelle**. Dies sind die relationalen Datenbanken, die objektorientierten, die hierarchischen und die netzwerkartigen Datenbanken. Die Unterschiede dieser vier 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

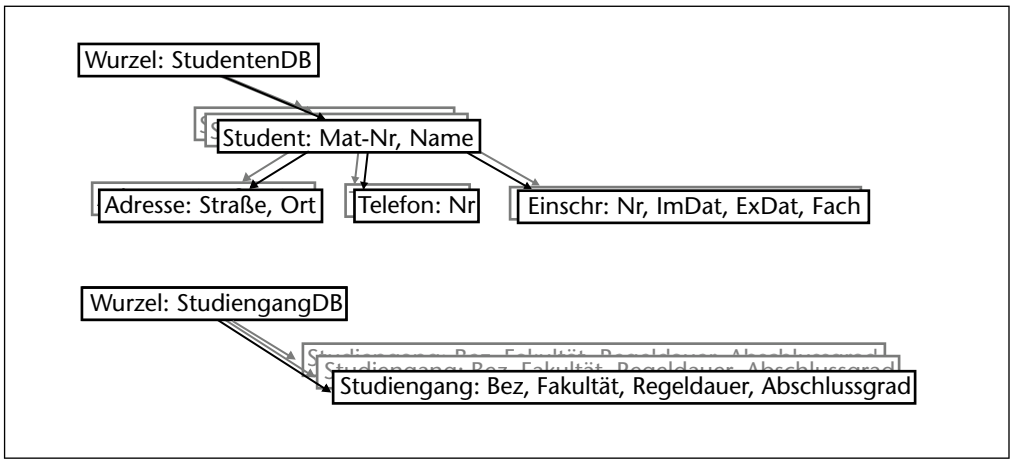
Eine **objektorientierte Datenbank** besteht ausschließlich aus Objekten. Ein Objekt ist entweder ein realer Gegenstand, z. B. ein Flugzeug, eine Person oder ganz allgemein ein abstrakter Gegenstand, etwa eine Adresse, eine Rechnung, ein Vorgang oder eine Abteilung einer Firma.

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).

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.

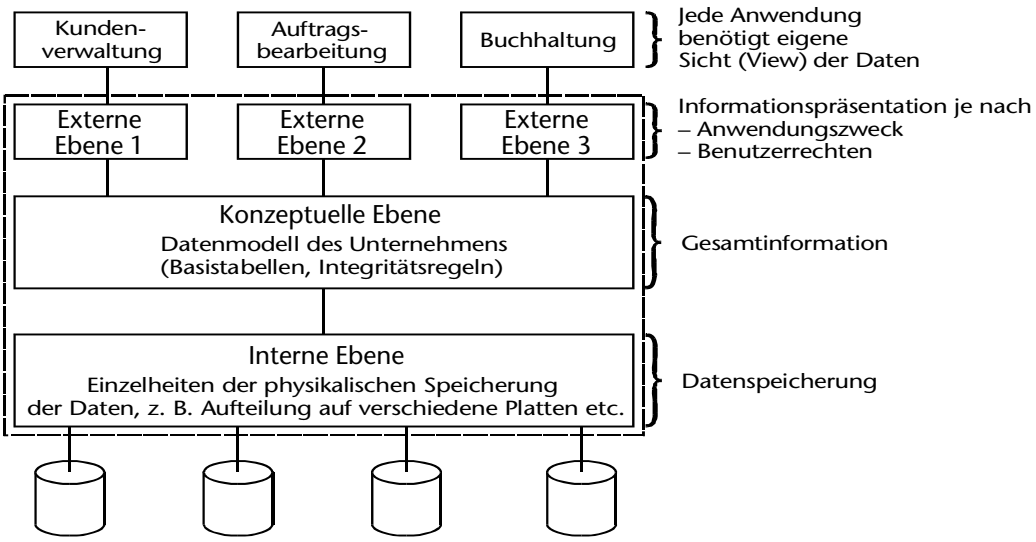
1.4 Architektur eines Datenbankmanagementsystems DBMS

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.

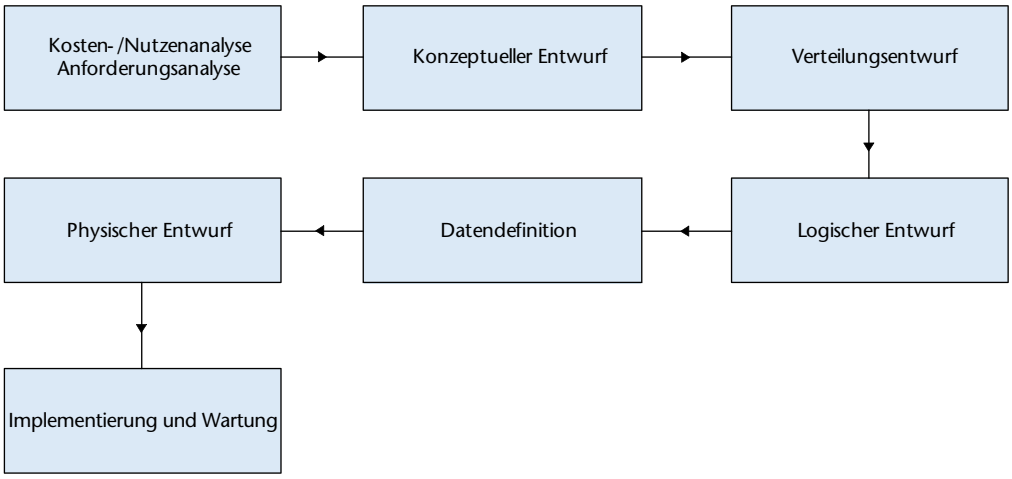
Hinweis:

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. Was versteht man unter einem Datenbankmanagementsystem?
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.
10. Beschreiben Sie eine Client/Server-Datenbank.
11. Nennen Sie verschiedene Datenbankmodelle.
12. Beschreiben Sie das relationale Modell.
13. Welchen Vorteil bieten hierarchische Datenbanken?
14. Beschreiben Sie die Drei-Ebenen-Architektur.
15. Geben Sie zu jeder Ebene der drei Schichten deren Aufgabe an.
16. Welchen Vorteil haben objektorientierte Datenbanken?
17. Welcher Unterschied besteht zwischen einem Datenbanksystem und der Datenspeicherung im PC?
18. Welche Aufgaben werden durch das Data Dictionary (Datenkatalog) gelöst?

2 Relationale Datenbanksysteme

2.1 Relationale Datenbanksysteme

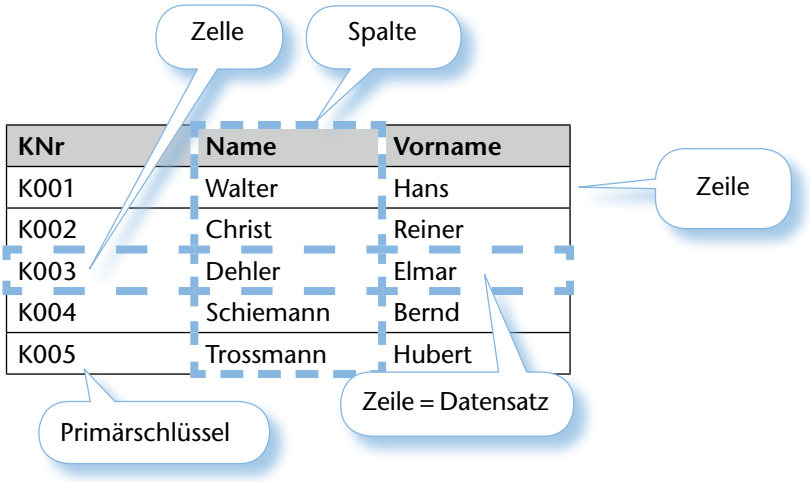
Die Daten einer relationalen Datenbank, z. B. Kundendaten oder Produktdaten, werden in Form von Tabellen verwaltet, die zueinander in Bezug stehen können.

2.1.1 Tabellen und Relationen

Tabellen, die zueinander in Beziehung stehen, werden auch als **Relationen** bezeichnet. Jede Zeile einer Tabelle enthält einen **Datensatz**.

Hinweis:

Relationale Datenbanksysteme verwalten Daten in Tabellen, die miteinander in Beziehung stehen.



Die Spalten der Tabellen enthalten vergleichbare Daten der einzelnen Datensätze, z. B. den Namen eines Kunden. Die Werte in einer Spalte sind jeweils in einem festzulegenden **Felddatentyp** (Datentyp) abgespeichert. Datenbanksysteme verfügen über verschiedene Datentypen.

Datentypen von Datenbanksystemen		
Datentyp	**Beschreibung**	**Beispiel**
Integer	Ganzzahl	5
Numeric (x.y)	Dezimalzahl mit x Stellen und y Nachkommastellen	53.27
Decimal (x.y)	Dezimalzahl mit mindestens x Stellen und y Nachkommastellen	53.2768
Float	Gleitkommazahl	8.3E13
Character	Zeichenkette	CH80653
Date	Datum	31.05.1966
Time	Uhrzeit	17:55

2.1.2 Schlüssel und Beziehungen

Die einzelnen Zeilen einer Tabelle enthalten **Datensätze**. Die Datensätze beschreiben beispielsweise eine Person, einen Gegenstand oder ein Ereignis. Datensätze nennt man auch **Entitäten** (von entity = Wesen) oder Objekte.

Hinweis:

Entität, Objekt, Datensatz entsprechen einer Zeile in einer Datenbanktabelle.

Objekte könnten mit weiteren Objekten innerhalb der Tabelle verwechselt werden, wenn sie denselben Namen haben. In der Tabelle Kunden sind dies zwei Kunden, für die das zutrifft. Um Verwechslungen auszuschließen, ist es notwendig, für jeden Datensatz eine Kennung durch eindeutige Felder festzulegen. Diese Felder werden als **Primärschlüssel** (**Primary Key**) bezeichnet. Sie ermöglichen die eindeutige Identifikation des jeweiligen Datensatzes. In der Tabelle Kunden ist dies meist die Kundennummer.

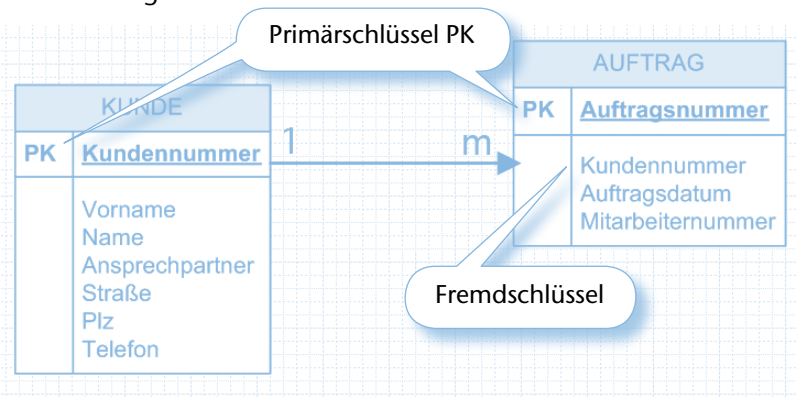
Hinweis:

Ein Primärschlüssel dient zur eindeutigen Kennzeichnung und Identifizierung eines Objektes in einer Tabelle.

1:m-Beziehung

Kauft ein Kunde einen Artikel, ist es nicht sinnvoll sämtliche Daten des Kunden erneut in einer weiteren Tabelle Kunde einzugeben. Es wird nur noch die `Kundennummer` in der Tabelle `Auftrag` gespeichert.

Bei der grafischen Darstellung weist eine Beziehungslinie von z. B. einer Tabelle mit Kundendaten zu einer Tabelle mit Auftragsdaten.



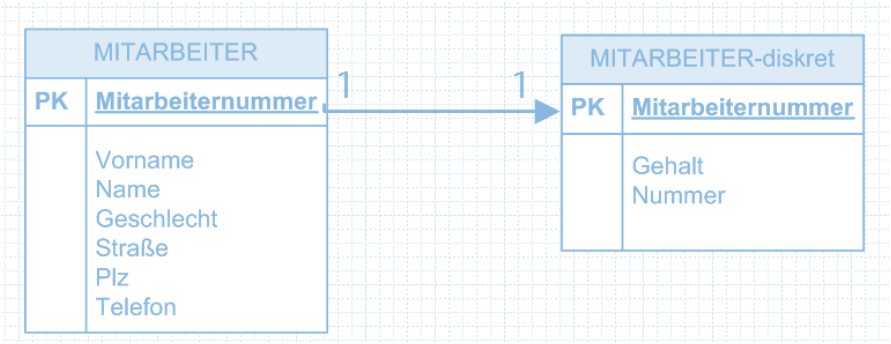
Die Tabellen Kunde und `Auftrag` stehen über das Primärschlüsselfeld der Tabelle Kunde in Beziehung zueinander. Jeder Datensatz der Tabelle `Auftrag` enthält eine `Kundennummer`, die genau einem Kunden „1“ aus der Tabelle Kunde zugeordnet ist. Umgekehrt kann jeder Kunde aus der Tabelle Kunde in mehreren „m“ Zeilen der Tabelle `Auftrag` angesprochen werden. Die Tabellen Kunde und `Auftrag` besitzen somit eine 1:m-Beziehung. Die Datensätze, die in der Tabelle Kunde durch den Primärschlüssel `Kundennummer` eindeutig identifiziert werden, sind auch in der Tabelle `Auftrag` durch diese `Kundennummer` festgelegt. Das Feld `Kundennummer` verweist auf das entsprechende Feld der Tabelle Kunde und wird deshalb als **Bezugsschlüssel** oder **Fremdschlüssel** bezeichnet.

Hinweis:

Ein Fremdschlüssel ist ein Feld, welches auf ein Primärschlüsselfeld einer anderen Tabelle verweist.

1:1-Beziehung

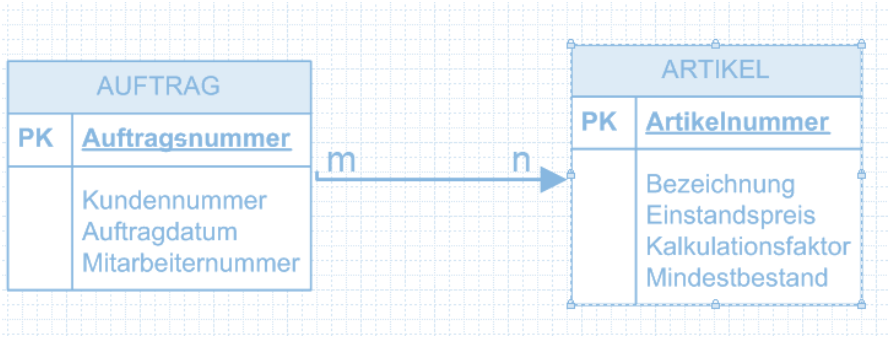
Aus einer Personaltabelle können Daten, z. B. das Gehalt, ausgelagert werden, die nicht jedem Benutzer zur Einsicht offen sein sollen. So entsteht eine neue Tabelle `Mitarbeiter-diskret`. Da auch hier die Datensätze über den Primärschlüssel `Mitarbeiternummer` gefunden werden, steht einem Datensatz der ersten Tabelle genau ein Datensatz der zweiten Tabelle gegenüber. Eine solche Beziehung wird 1:1-Beziehung genannt.



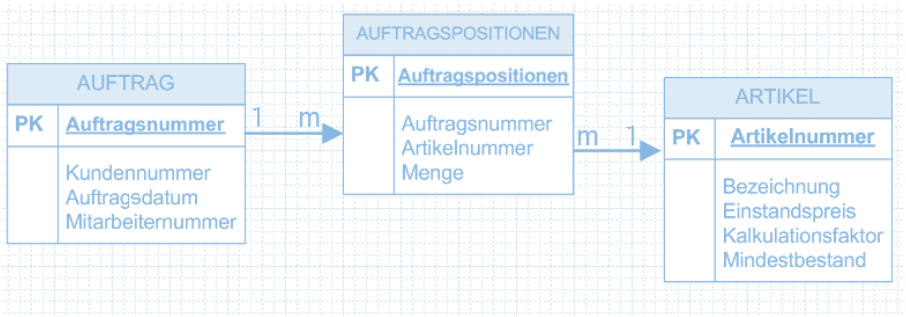
1:1-Beziehungen kommen in der Praxis selten vor.

m:n-Beziehung

Betrachtet man die Tabellen `Auftrag` und `Artikel`, so kann ein Auftrag sicher viele „m“, „n“ oder „ ∞“ Artikel enthalten, andererseits kann ein Artikel auch in beliebig vielen „n“, „m“ oder „∞“ Aufträgen enthalten sein (∞ bedeutet unendlich).



Durch die Tabelle `Auftragspositionen` werden zwei 1:n-Beziehungen hergestellt.



Ein Artikel, der in einem Datensatz Auftragspositionen genannt wird, kann eindeutig in der Tabelle `Artikel` identifiziert werden. Umgekehrt kann ein Artikel in mehreren Auftragspositionen angesprochen werden. Somit liegt zwischen den Tabellen `Artikel` und `Auftragspositionen` eine 1:m-Beziehung vor. Auch die Tabellen `Auftragspositionen` und `Auftrag` sind über eine 1:m-Beziehung verbunden. Ein Auftrag hat zwar beliebig viele Artikelpositionen, aber jede Artikelposition ist genau einem Auftrag zuzuordnen.

Man nennt die beiden äußeren Tabellen Mastertabellen (ebenso: starke Entität oder strong entity) und die mittlere Tabelle Childtabelle (Verbindungstabelle, schwache Entität oder weak entity).

Hinweis:

Eine m:n-Beziehung muss in einer relationalen Datenbankumgebung in je eine 1:m- und eine m:1-Beziehung über eine geeignete Childtabelle aufgelöst werden. In der Childtabelle müssen die Primärschlüssel der Mastertabellen als Fremdschlüssel eingetragen werden.

Die Kardinalität beschreibt die Art der Beziehung der Datensätze zueinander. Man kann 16 mögliche Kardinalitäten unterscheiden.