

## 2 Grundlagen zu Datenbanken

### In diesem Kapitel erfahren Sie

- ✓ wie und warum Datenbanksysteme entwickelt wurden
- ✓ welche Datenbanktypen es gibt
- ✓ wie im 3-Ebenen-Modell die Sichten auf eine Datenbank beschrieben werden
- ✓ welche physischen Datenbankarchitekturen es gibt

### Voraussetzungen

- ✓ Grundlegende Kenntnisse im Arbeiten mit Dateien

### 2.1 Entwicklung der Datenbanken

Datenbanken spielen beim Einsatz von Computern häufig eine zentrale Rolle. Wenn Arbeitsabläufe computerunterstützt abgewickelt werden, ist meistens die Speicherung großer Datenmengen erforderlich. Als typische Beispiele dafür lassen sich die Personalverwaltung, die Lagerwirtschaft oder das Bestell- und Rechnungswesen großer Unternehmen nennen. Bei der Speicherung der Daten in einfachen Dateisystemen wurden unter anderem folgende Probleme erkannt:

#### ✓ Redundanzen

Viele Daten werden mehrfach gespeichert, was eine hohe Datenredundanz bedeutet. Dadurch werden Änderungen aufwendig (die gleichen Daten müssen mehrmals an verschiedenen Stellen geändert werden) und der Datenbestand ist fehleranfällig (wenn z. B. Änderungen von mehrfach gespeicherten Daten nur an einer Stelle vorgenommen werden).

#### ✓ Inkonsistenzen

Werden die Daten von mehreren Benutzern bzw. Programmen gleichzeitig geändert, kann es zu einem inkonsistenten Zustand der Daten kommen. Der Datenzugriff ist nicht synchronisiert.

Dazu kann es z. B. kommen, wenn ein Lagerbestand von zwei Benutzern gleichzeitig aktualisiert wird. Beide Benutzer lesen die aktuelle Anzahl eines Artikels (z. B. 8000). Der erste Benutzer fügt 100 Artikel hinzu und speichert den Wert (8100). Der zweite Benutzer entnimmt 500 Artikel und speichert den Wert (7500), wobei die Änderung des ersten Benutzers überschrieben wird. Die Daten sind inkonsistent, da der richtige Wert 7600 beträgt.

#### ✓ Datenschutzprobleme

Ein Lesezugriff auf die gesamten Daten ist zum Teil uneingeschränkt möglich. Er kann jedoch, abhängig vom verwendeten Betriebssystem, durch Lesesperren oder Verschlüsselung unterbunden werden.

#### ✓ Fehlende Datenunabhängigkeit

Eine komfortable Verwaltung der Daten ist nur durch die dafür konzipierte Software möglich.

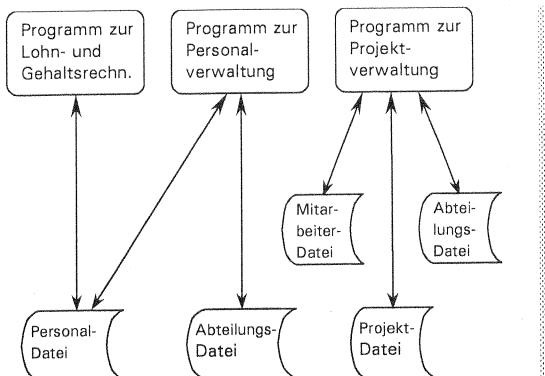
Ist eine Änderung der Struktur der Daten erforderlich, muss sowohl die Anwendungssoftware geändert als auch ein Programm zur Umstrukturierung der Dateien erstellt werden.

Sollen die gleichen Dateien in einer anderen Anwendung ausgewertet werden, muss für diese neue Anwendung ebenfalls eine eigene Datenverwaltung erstellt werden.

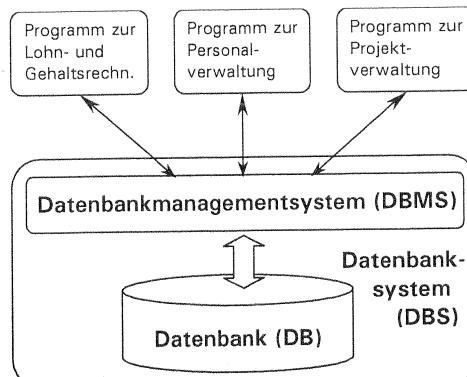
Durch die Entwicklung von Datenbanksystemen wurden diese Probleme nach und nach gelöst. In einem DBS werden die Daten in einer Datenbank zusammengefasst, die ausschließlich von dem Datenbankmanagementsystem (DBMS) der Datenbanksoftware verwaltet wird. Anwendungsprogramme greifen nun nicht mehr direkt auf die Daten zu, sondern stellen ihre Anforderungen nur noch an das Datenbankmanagementsystem. Die enge Verflechtung und Abhängigkeit von Daten und den damit arbeitenden Anwendungsprogrammen wird stark reduziert bzw. ganz aufgehoben.

## Grundlagen zu Datenbanken

Ein DBS besteht aus einer Anzahl von Datenbanken und dem DBMS. Die Datenbank ist die Sammlung von logisch zusammengehörigen Daten zu einem Sachgebiet. Das DBMS stellt die Schnittstelle zwischen der Datenbank und deren Benutzern (z. B. den Anwendungsprogrammen) her. Es gewährt einen effizienten und adäquaten (adäquat = angemessen) Zugriff auf die Daten und sorgt dabei für eine zentrale Steuerung und Kontrolle. Außerdem wird durch das DBMS ein Schutz gegen Hard- und Softwarefehler gewährleistet, sodass bei spielsweise bei Programm- oder Systemabstürzen die Daten nicht verloren gehen bzw. wiederhergestellt werden können.



Anwendung arbeitet direkt mit Dateien



Anwendungen greifen über die Datenbanksoftware auf die Daten zu, die in der Datenbank gespeichert sind

Eine Datenbank weist folgende Eigenschaften auf:

- ✓ In Datenbanken sind Daten entsprechend ihren natürlichen Zusammenhängen gespeichert. Dabei ist es nicht entscheidend, in welcher Form die Daten in Anwendungen benötigt werden. Die Daten der Datenbank bilden einen Ausschnitt aus der realen Welt ab.
- ✓ Auf die Daten einer Datenbank können viele Benutzer gleichzeitig zugreifen. Das Datenbankmanagementsystem verwaltet sowohl die Daten als auch die Zugriffe darauf und sorgt dafür, dass dieselben Daten nicht gleichzeitig von mehreren Benutzern bearbeitet werden können. Auch wenn mehrere Anwendungen mit einer gemeinsamen Datenbasis arbeiten, so wird in den Anwendungsprogrammen meist nur auf einen Teil davon zugegriffen (auf eine Sicht auf die Datenbank).

Das Datenbankmanagementsystem ist die Software, welche die Datenbanken verwaltet. Es ermöglicht ...

- ✓ das Anlegen von Datenbanken,
- ✓ die Speicherung, Änderung und Löschung der Daten,
- ✓ das Abfragen der Datenbank,
- ✓ die Verwaltung von Benutzern, Zugriffen und Zugriffsrechten.

Um von Anwendungsprogrammen bzw. Benutzern auf die Daten einer Datenbank zugreifen zu können, stellt das DBS ein Sprachkonzept zur Verfügung. Bei relationalen DBS ist das meist die Datenbanksprache SQL (Structured Query Language - strukturierte Abfragesprache).

## 2.2 Datenbankmodelle

Der Übergang von der Nutzung des Dateisystems zur Datenbank vollzog sich schrittweise.

- ✓ Zunächst wurden die Dateien auf Magnetbändern gespeichert (50er-Jahre).
- ✓ Diese wurden durch den Einsatz der schnelleren Magnetplatten abgelöst. Sie brachten den Vorteil des Direkt- und Mehrfachzugriffs (60er-Jahre).
- ✓ In den ersten Datenbanksystemen (1. Generation) wurde das hierarchische Datenmodell verwendet (70er-Jahre).
- ✓ Die Weiterentwicklung brachte das Netzwerkmodell hervor.

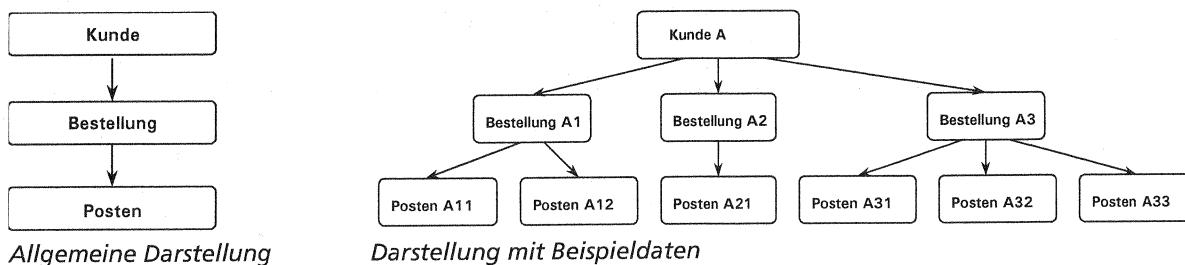
- ✓ Die nächste Datenbank-Generation wurde mit dem relationalen Modell eingeleitet. Dieses stammt bereits aus den 70er-Jahren. Im folgenden Jahrzehnt wurde die maßgebliche praktische Umsetzung realisiert.
- ✓ Parallel dazu entwickelten sich alternative Ansätze in Form der spalten- bzw. der dokumentenorientierten Datenbanken.
- ✓ Dem Paradigma objektorientierter Sprachen folgend wurden objektorientierte Datenbanken entwickelt (Anfang der 90er-Jahre).
- ✓ Der nachfolgenden Datenbank-Generation wurde das objektrelationale Datenmodell zugrunde gelegt, das die Vorteile der objektorientierten und der relationalen Datenmodelle vereinigt (Mitte der 90er-Jahre).
- ✓ Seit der Mitte des ersten Jahrzehnts des neuen Jahrhunderts ist ein verstärkter Trend zu den sogenannten NoSQL-Datenbanken zu verzeichnen.

## Hierarchische Datenbanken

Das hierarchische Datenmodell wurde entwickelt, um unterschiedlich lange Datensätze (zusammengehörige Informationen, eine bestimmte Sache betreffend) zu verarbeiten. Die Datensätze werden so aufgeteilt, dass gleichartige Daten zu kleineren Datengruppen zusammengefasst werden. Diese Gruppen bilden die Knoten der Hierarchie. So entsteht eine **baumartige Struktur**, die streng hierarchisch geordnet ist. Jeder untergeordnete Knoten ist von seinem übergeordneten Knoten abhängig. Die Struktur entspricht einer **Vater-Sohn-Beziehung**. Ein Vater kann mehrere Söhne haben, ein Sohn aber nur einen Vater. Die Struktur kann nicht ohne den Wurzelknoten existieren.

### Beispiel

Die Abbildung zeigt die hierarchische Struktur einer Kundenverwaltung. Jeder Kunde kann eine unterschiedliche Anzahl von Bestellungen mit einer bestimmten Anzahl von Posten aufgeben. So ist nicht vorhersehbar, wie lang ein bestimmter Datensatz ist. Die Daten der Kunden, Bestellungen und Posten bilden im hierarchischen Datenmodell die Knoten des Hierarchiebaums. Für jeden Knoten wird in der Datenbank ein Datensatz angelegt.



**i** Eine bekannte Datenbank mit hierarchischer Struktur ist IMS/DB von IBM.

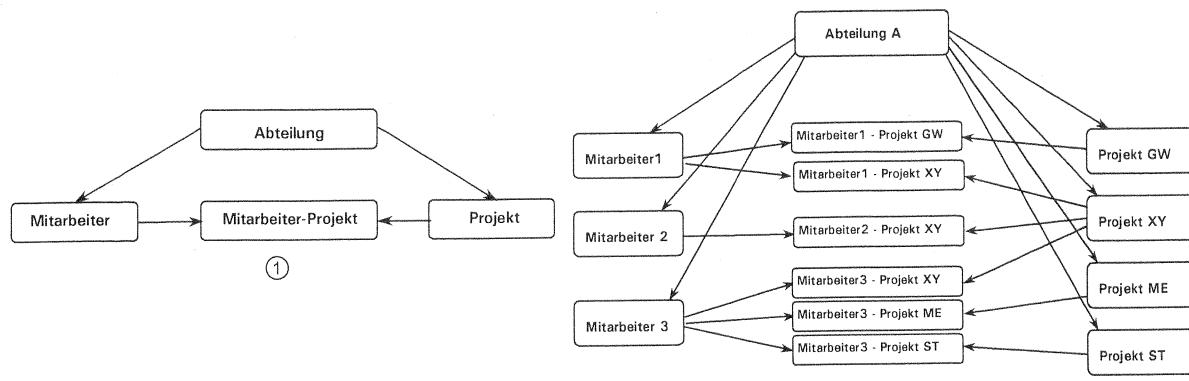
Nachdem die hierarchischen Datenbanken bis auf einzelne Nischen aus der praktischen Anwendung verdrängt wurden, finden heute mit der Verbreitung der erweiterbaren Auszeichnungssprache XML (Extensible Markup Language) ihre theoretischen Ansätze wieder eine Anwendung.

## Netzwerkdatenbanken

Beim Netzwerkmodell werden gleichartige Daten in **Recordsets** gespeichert, die miteinander in Beziehung stehen. Einem Record eines Recordsets können dabei mehrere Records eines anderen Recordsets zugeordnet werden, was als Pfeil in der grafischen Darstellung erscheint. Durch diese Beziehungen zwischen den Recordsets entsteht ein gerichteter Graph, der auch als Netzwerk bezeichnet wird. Die Beziehungen werden hier als **Sets** (Mengen) bezeichnet. Die Sets sind in der Datenbank fest definiert.

### Beispiel

Es wird die Projektverwaltung einer Abteilung betrachtet. Zu einer Abteilung gehören mehrere Mitarbeiter. In jeder Abteilung wird an mehreren Projekten gearbeitet. An jedem Projekt arbeiten mehrere Mitarbeiter der Abteilung mit, jeder Mitarbeiter kann aber auch an mehreren Projekten mitarbeiten. Diese Beziehungen sind in der Abbildung durch Pfeile dargestellt.



Allgemeine Darstellung

Darstellung mit Beispieldaten

Beim Netzwerkmodell darf eine Beziehung zwischen zwei Recordsets immer nur in eine Richtung zeigen. Zwischen den Recordsets *Mitarbeiter* und *Projekt* wäre eine beidseitige Beziehung notwendig (mehrere Mitarbeiter arbeiten an mehreren Projekten mit). Das Problem wird durch das Einfügen eines zusätzlichen Recordsets *Mitarbeiter-Projekt* gelöst ①.

Eine verbreitet zum Einsatz gekommene Netzwerkdatenbank ist u. a. die UDS (Universal Datenbank System) von Siemens.

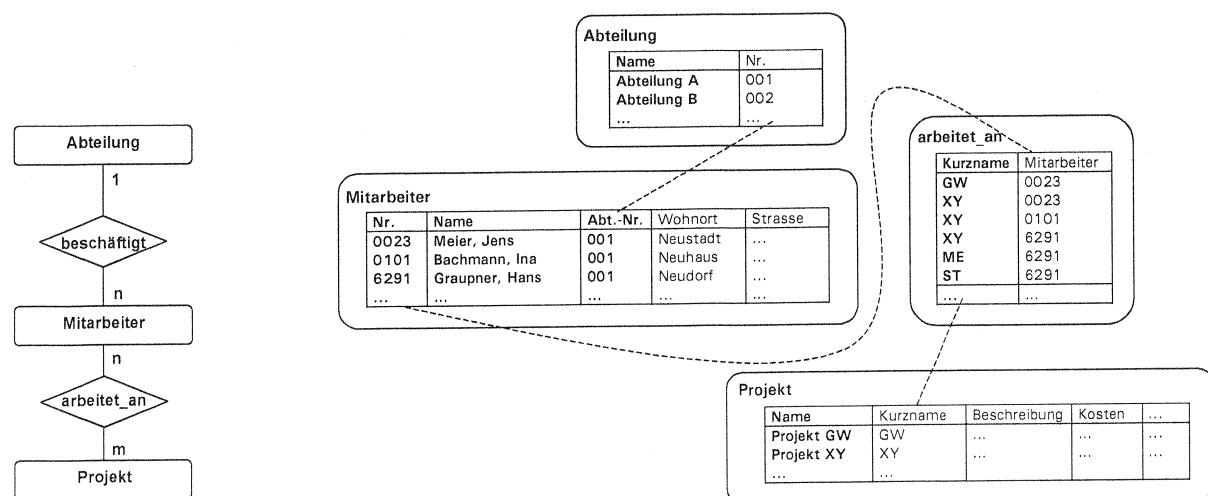


## Relationale Datenbanken

Relationale Datenbanken sind am weitesten verbreitet. Die Daten werden in Tabellenform gespeichert, in sogenannten Relationen (mathematische Ausdrucksweise). Zwischen den Relationen (Tabellen) können Beziehungen definiert werden. Es sind verschiedene Beziehungsarten möglich, die sich durch die Anzahl der miteinander in Beziehung stehenden Datensätze (Tabellenzeilen, Tupel) unterscheiden (einer mit einem 1 : 1, einer mit mehreren 1 : n, mehrere mit mehreren n : m). Die grafische Darstellung der Relationen und der zugehörigen Beziehungen erfolgt meist im Entity-Relationship-Modell (ERM). Die Beziehungen werden dort durch Linien dargestellt, die mit den Kardinalitäten (1, n, m) versehen sind. Über Abfragen ist es möglich, für die bestehenden Datenbanken unterschiedliche Auswertungen durchzuführen. Für die Abfrage und Auswertung der Daten hat sich die Abfragesprache SQL durchgesetzt.

## Beispiel

Die Projektverwaltung lässt sich im relationalen Modell wie folgt darstellen: Einer Abteilung gehören mehrere Mitarbeiter an, jeder Mitarbeiter ist aber nur in einer Abteilung beschäftigt. Zwischen den Relationen *Abteilung* und *Mitarbeiter* besteht eine 1:n-Beziehung. Jeder Mitarbeiter kann an mehreren Projekten mitarbeiten, ein Projekt kann wiederum von mehreren Mitarbeitern bearbeitet werden. Zwischen den Relationen *Mitarbeiter* und *Projekt* besteht eine n:m-Beziehung.



Darstellung im ER-Modell

Darstellung mit Beispieldaten

Die Daten der Tabellen werden physisch zeilenweise auf einem externen Speichermedium gesichert. Für die häufige Veränderung und Bearbeitung von Daten, die bei Geschäftsvorfällen notwendig ist, das Online Transaction Processing (OLTP), eignet sich diese Art der Speicherung sehr gut.

-  Es gibt eine Vielzahl relationaler Datenbanken, neben anderen Oracle, Microsoft SQL, DB2 von IBM, MySQL und PostgreSQL.

## Spaltenorientierte Datenbanken

Spaltenorientierte Datenbanken basieren ebenfalls auf dem architektonischen Ansatz der Relationen bzw. Tabellen. Im Gegensatz zu den relationalen Datenbanken erfolgt die physische Speicherung hier jedoch in einer anderen Form. Die Werte der Tabellen werden pro Spalte - also über ein Attribut aller Tupel hinweg - auf das externe Speichermedium geschrieben.

Anhand der externen Speicherung der Relation *arbeitet\_in* des Beispiels der Projektverwaltung zeigt sich der Unterschied folgendermaßen:

**Relationale Datenbank:**

GW, 0023; XY, 0023; XY, 0101; XY, 6291; ME, 6291; ST, 6291

**Spaltenorientierte Datenbank:**

GW, XY, XY, XY, ME, ST; 0023, 0023, 0101, 6291, 6291, 6291

arbeitet_an	
Kurzname	Mitarbeiter
GW	0023
XY	0023
XY	0101
XY	6291
ME	6291
ST	6291
...	...

Diese Speicherform bietet bei der reinen Auswertung von Daten, dem Online Analytical Processing (OLAP), deutliche Geschwindigkeitsvorteile, da hier häufig nur die Werte einzelner Spalten (Attribute) von Interesse sind. Der Leseprozess kann auf diese Spalten beschränkt werden. Damit entfällt das Lesen einer Vielzahl nicht benötigter Daten der einzelnen Tupel.

Bei der Bearbeitung von Daten kann diese Speicherform ebenfalls Vorteile besitzen. Dies ist immer dann der Fall, wenn nur ein Attribut über viele Tupel hinweg geändert werden muss.

-  Einer der ersten Vertreter der spaltenorientierten Datenbanken war Sybase IQ. Mittlerweile gibt es eine Vielzahl kommerzieller (z. B. Oracle Retail Predictive Application Server [RPAS]) und nicht kommerzieller (z. B. Apache Cassandra) Implementierungen.

## Dokumentorientierte Datenbanken

Nicht alle Informationen lassen sich gut in zweidimensionalen Relationen ablegen. Mengen strukturierter Informationen, die sich in den Ausprägungen der Einzelinformationen über die vorhandenen Attribute hinweg oder in der Menge der Werte der einzelnen Attribute unterscheiden, sind nicht effektiv in einem starren Relationenschema abbildbar.

Dieser Fakt führte zum Entwurf der dokumentenorientierten Datenbanken. Eine zusammengehörige Informationsmenge wird in einem Dokument gespeichert. Dieser Begriff steht dabei für eine strukturierte Information und hat nichts mit dem gleichen Begriff, wie er im Kontext der Textverarbeitung bekannt ist, zu tun. Die Daten innerhalb eines Dokuments werden in Feldname-Wert-Paaren abgespeichert. Dabei muss nicht jedes Feld in jedem Dokument vorhanden sein und einzelne Felder können in unterschiedlichen Dokumenten eine unterschiedliche Anzahl von Werten besitzen.

In Analogie zu dem Projektbeispiel könnte ein Ausschnitt aus einer dokumentenorientierten Datenbank beispielweise folgendermaßen aussehen:

```
{
(Vorname: Anton, Nachname: Schulze, AktProjekte: P1; P2; P3);
(Vorname: Maria, Nachname: Meier, Funktion: Projektleiter, AktProjekte: P2);
(Vorname: Matthias, Nachname: Schneider, Funktion: Mitarbeiter, AktProjekte: P4);
}
```

-  Ein typischer Vertreter dokumentenorientierter Datenbanken ist Lotus Notes Domino.

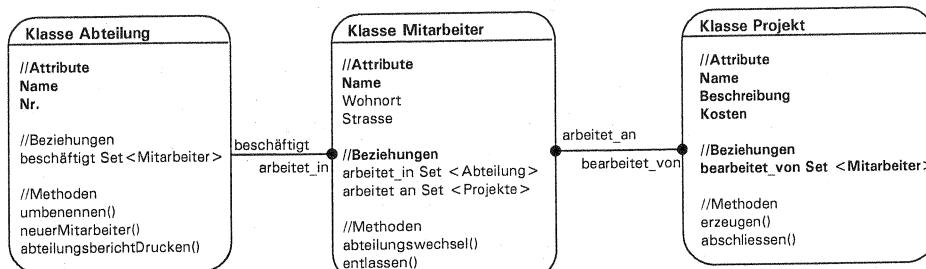
## Objektorientierte Datenbanken

Objektorientierte Datenbanken (OODB) sind nach dem Paradigma der objektorientierten Programmiersprachen entwickelt worden. Das Ziel der Entwicklung der objektorientierten DBS war es, ein DBS zu schaffen, in welchem Objekte unserer Umwelt mit ihren Eigenschaften und ihrem Verhalten nachgebildet (analog zu objektorientierten Programmiersprachen) und ohne großen Aufwand in einer Datenbank gespeichert und verwaltet werden können.

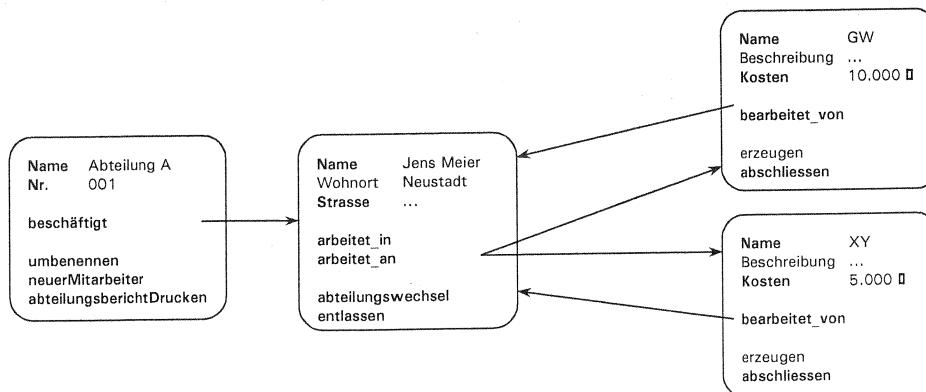
Jedes Objekt der Datenbank enthält Dateninformationen (Attribute), Verweise auf andere Objekte und Operationen (Methoden), die das Verhalten des Objekts widerspiegeln. Die Zusammenfassung von Daten und Operationen über diese Daten wird als Kapselung bezeichnet. Die Definition der Objekte (Daten, Verweise, Methoden) erfolgt über sogenannte Klassen. Durch die freie Beschreibung von Objekt-Klassen lassen sich selbst komplexeste Datenstrukturen in einer Datenbank verwalten, was in den flachen Tabellen relationaler DBS nur über mehrere in Beziehung stehende Tabellen möglich ist. Neben der Kapselung von Daten und Operationen sind weitere objektorientierte Konzepte in OODBS implementiert, wie z. B. die Vererbung, die Überladung von Methoden und dynamische Bindungen. Zusätzlich stehen in OODBS mehr Datentypen zur Verfügung als in den zuvor beschriebenen Datenbankmodellen (z. B. Arrays oder Klassentypen).

### Beispiel

Für die Projektverwaltung werden drei verschiedene Objekt-Klassen (Abteilung, Mitarbeiter und Projekte) definiert. Bestandteil der Klassendefinition sind auch die Verweise auf andere Objekte sowie die verschiedenen Operationen (Methoden). So hat ein Objekt der Klasse Abteilung einen Verweis auf eine unbestimmte Menge von Objekten der Klasse Mitarbeiter. Jeder Mitarbeiter besitzt einen eindeutigen Verweis auf seine Abteilung und einen Verweis auf eine unbestimmte Anzahl von Projekten. Jedes Projekt hat einen Verweis auf eine unbestimmte Anzahl von Mitarbeitern. Die Methoden der Klasse Abteilung könnten beispielsweise *neuerMitarbeiter*, *umbenennen* oder *abteilungsberichtDrucken* sein.



Darstellung im objektorientierten Datenbankschema



Darstellung mit Beispieldaten

Objektorientierte Datenbanken sind nach wie vor nur wenig verbreitet. Eine der wenigen verfügbaren Umsetzungen ist db4o.

## Objektrelationale Datenbanken

Objektrelationale Datenbanksysteme (ORDBS) wurden entwickelt, um die Nachteile relationaler DBS zu beseitigen. Solche Nachteile sind z. B. das "magere" Typsystem von relationalen DBS (umfangreiche Texte, Excel-Tabellen, XML-Dokumente, Audiodaten usw. können nur mit erhöhtem Aufwand oder gar nicht in der DB gespeichert werden) und Probleme bei der Abbildung der Objekte der realen Welt auf das relationale Modell, was als "impedance mismatch" bezeichnet wird. Das Ziel der Entwicklung war es, die Vorteile der Speicherung komplexer Objekte wie im OODBS zu nutzen und dabei die einheitliche Abfragesprache SQL beizubehalten. Es sollten also die Vorteile des relationalen und des objektorientierten Modells in dem neuen DB-Modell vereinigt werden. Dazu musste nicht alles von Grund auf neu entwickelt werden. Es wurde das vorhandene relationale Modell genutzt, das um die objektorientierten Konzepte erweitert wurde. Diese Erweiterungen werden verwendet, um die Objekte zu interpretieren und zu verwalten. Verschiedene Erweiterungen werden mit dem Datenbanksystem ausgeliefert, es existieren aber auch zusätzliche Erweiterungen von Fremdfirmen und Sie können auch eigene erstellen. Die Erweiterungen werden beispielsweise als DataBlades (Informix) oder Cartridges (Oracle) bezeichnet.

Anwendungsprogramme können über die erweiterte SQL-Sprache (Standard SQL:1999, früher als SQL-3 bezeichnet) auf objektrelationale Datenbanken zugreifen. Dieser Standard wurde von der International Standardization Organization (ISO) und dem American National Standards Institute (ANSI) 1999 verabschiedet.

 Eine am Markt weitverbreitete objektrelationale Datenbank ist PostgreSQL.

## NoSQL-Datenbanken

Mit den im Zuge der Entwicklung des Internets sprunghaft ansteigenden Volumen von Daten in einzelnen Datenbanken wurden die Bemühungen zur Entwicklung alternativer Architekturformen von Datenbanken wieder verstärkt. Ursache dafür waren die Probleme, die relationale Datenbanken bei Anwendungen mit sehr großen Datenmengen haben. Diese treten dann auf, wenn es zu einer extrem großen Anzahl gleichzeitiger Schreib- und Lesezugriffe oder Datenänderungen kommt. Ein weiterer Punkt ist die starke Abhängigkeit der relationalen Datenbanken von einem Schema. Schemaänderungen großer Datenbanken sind sehr zeitaufwendig und im Hinblick auf einen möglichen Datenverlust nicht ungefährlich. NoSQL-Datenbanken besitzen diese starke Abhängigkeit von einem Schema nicht. Typisch für NoSQL-Datenbanken ist die Unterstützung von verteilten Datenbanken auf mehreren Servern, bei der durchaus eine redundante Datenhaltung erfolgen kann.

Vorreiter der Bewegung der NoSQL-Datenbanken waren neben anderen die Big Player des Internets, wie beispielsweise Google, Amazon, eBay und Facebook.

Der Begriff NoSQL - im Sinne von "nicht SQL" - trat zuerst als Name einer kleinen Datenbank Ende der 90er-Jahre des letzten Jahrhunderts auf. Später wurde er als Sammelbegriff der Bewegung von Datenbanken, die ein anderes als das relationale Konzept verfolgen, im Sinne von "nicht nur (not only) SQL" verwendet. Dabei greift die Bewegung durchaus Konzepte auf, die bereits zu einem früheren Zeitpunkt betrachtet wurden. Exemplarisch hierfür stehen die dokumentenorientierten Datenbanken sowie die spaltenorientierten Datenbanken.

NoSQL-Datenbanken können in drei Gruppen unterteilt werden:

- ✓ **Dokumentenorientierte Datenbanken**
- ✓ **Graphen-Datenbanken**  
Dieser Typ spielt seine Stärken vor allem bei Anwendungen aus, die eine Vielzahl von Querverbindungen zwischen Daten verwalten müssen. Typisch ist dies zum Beispiel bei Webanwendungen wie Twitter (Wer folgt wem?). Architektonisch können Graphen-Datenbanken als Nachfolger der Netzwerkdatenbanken betrachtet werden. Die Daten werden als Knoten dargestellt und die Beziehungen zwischen ihnen durch die Verbindungen zwischen diesen Knoten.
- ✓ **Key/Value-Datenbanken** (zum Teil in Kombination mit spaltenorientierten Datenbanken)  
Key/Value-Datenbanken besitzen ein sehr einfaches Schema. Einem Key (Schlüssel), dargestellt durch eine willkürliche Zeichenkette, ist jeweils ein Value (Wert, Einzelwert, Liste oder Set) zugeordnet. Eine wichtige Unterteilung der Key/Value-Datenbank stellt der Ort der Speicherung der Daten dar - im RAM oder auf einem externen Medium.

 Eine moderne dokumentenorientierte Datenbank ist neben dem noch immer aktuellen Lotus Notes die CouchDB. Eine bekannte Graphen-Datenbanken ist Neo4J. Typische Vertreter des Key/Value-Datenbanktyps sind Big Table von Google, Cassandra und Redis.

## 2.3 Aufbau und Organisation einer Datenbank

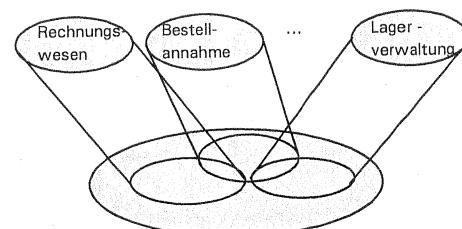
Eines der wichtigsten Ziele, welches ein DBS realisieren muss, ist die Datenunabhängigkeit. Diese wird durch die Trennung der physischen Speicherung der Daten und deren Verwaltung von den Anwendungsprogrammen erreicht. Zum einen sollte eine physische Datenunabhängigkeit bestehen, d. h., für Programme und Benutzer sollte die physische Organisation der Daten transparent sein. So kann die Struktur der gespeicherten Daten geändert werden, ohne dass die Anwendungsprogramme geändert werden müssen. Zum anderen ist auch die logische Datenunabhängigkeit eine wichtige Anforderung an ein Datenbanksystem. Damit kann zwischen einer logischen Gesamtstruktur der Datenbank und den anwenderspezifischen Sichten auf die Daten unterschieden werden. So können weitere Anwendungen und Sichten auf eine bestehende Datenbank erstellt werden, ohne dass bereits existierende Anwendungen dadurch beeinflusst werden.

Eine **Sicht** ist ein Ausschnitt einer Datenbank, der für eine Anwendung bzw. ein Problem relevanten Daten enthält.



### Beispiel

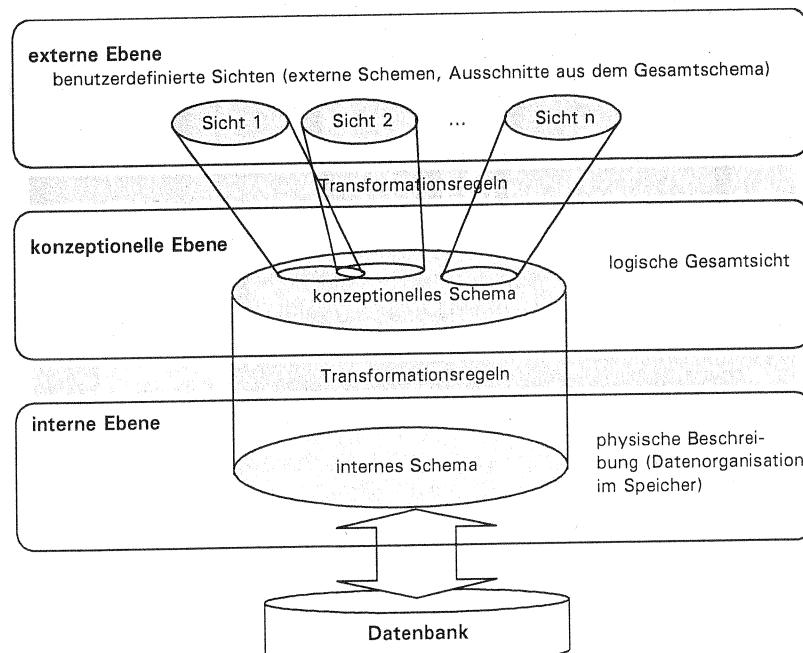
Ein Unternehmen des Großhandels besitzt eine große Datenbank, die die Artikeldaten, die Daten der Lagerverwaltung, die Kundendaten und die Daten der Bestellannahme und Rechnungslegung enthält. In den verschiedenen Abteilungen werden nur Teile der Datenbank, bestimmte Sichten, benötigt. So werden bei der Bestellannahme nur die Kunden- und Artikeldaten sowie die im Lager vorhandenen Artikel benötigt. Für die Verwaltung der Artikel im Lager wird nur auf die Artikel- und Lagerdaten zugegriffen. Das Rechnungswesen verwendet die Kundendaten sowie die Artikel- und Bestelldaten. Jede Abteilung verwendet nur die Daten, die sie auch benötigt. Für die Rechnungslegung wird beispielsweise keine Information zum Lagerbestand benötigt.



Logische Gesamtansicht auf die DB des Unternehmens

### 3-Ebenen-Modell

Am 3-Ebenen-Modell nach ANSI-SPARC 1978 (American National Standards Institute / Standards Planning and Requirements Committee) werden die unterschiedlichen Sichtweisen auf einen Datenbestand dargestellt. Das Prinzip der physischen und logischen Datenunabhängigkeit wird hier deutlich. Darunter versteht man die Möglichkeit des Datenzugriffs durch Programme oder Anwender, ohne dass diese über Kenntnisse der tatsächlichen technischen Umsetzung der physischen Speicherung und der Manipulationsprozesse der Daten verfügen.



Auf der **externen Ebene** erfolgt die Darstellung der Daten, wie sie in den einzelnen Anwendungen benötigt werden. In den Benutzersichten (kurz: Sichten) werden Teile der logischen Gesamtsicht so wiedergegeben, dass dem Benutzer nur die Daten zugänglich sind, mit denen er arbeiten darf. Die Typen der Daten und deren Beziehungen in den Sichten können dabei anders aufgebaut sein als im konzeptionellen Schema. So sind häufig nur Teile der Daten eines Datenobjekts für eine Anwendung relevant. Beispielsweise wird bei der Rechnungslegung der Lagerbestand der Artikel nicht benötigt, der zum Datenobjekt *Artikel* gehört. Die Benutzer sind mithilfe der Datenabfrage- und Datenmanipulationssprache (DQL - Data Query Language/DML - Data Manipulation Language) in der Lage, auf die Daten zuzugreifen (zu selektieren und zu lesen) und sie zu verändern (bzw. zu löschen und neue hinzuzufügen).

In der **konzeptionellen Ebene** werden alle Daten eines Anwendungsbereichs (z. B. Gesamtheit der Daten eines Unternehmens) zusammengefasst, die in der Datenbank gespeichert werden sollen. Auch die logischen Zusammenhänge sowie Änderungsvorschriften für die Daten müssen beschrieben werden. So entsteht eine logische Gesamtsicht, die auch als "Ausschnitt aus der realen Welt" bezeichnet wird. Die Beschreibung der Daten und ihrer Zusammenhänge erfolgt so, wie diese in der Realität vorkommen, und ist nicht auf die Bedürfnisse einzelner Anwendungen zugeschnitten. Beispielsweise werden alle Daten eines Unternehmens in der Datenbank abgelegt, die in den einzelnen Abteilungen ausgewertet bzw. verwaltet werden müssen. Das konzeptionelle Schema (konzeptionelle Modell) wird mithilfe einer geeigneten Datendefinitionssprache (DDL - Data Definition Language) beschrieben. Diese Aufgaben erledigt in der Regel der Datenbankadministrator.

Die **interne Ebene** beschreibt die Organisation der Daten auf den Speichermedien sowie die Zugriffsmöglichkeiten auf diese Daten. Vom Datenbankadministrator wird, vom konzeptionellen Modell ausgehend, eine physische Datenorganisation entwickelt, die für alle Benutzer einen optimalen Datenzugriff sicherstellt. Das interne Modell wird direkt in die Datenbank übertragen.

Zwischen den 3 Ebenen erfolgt eine Transformation der Schemen ineinander. Dafür besitzt das DBMS **Transformationsregeln**.

## Datenbankmanagementsystem (DBMS)

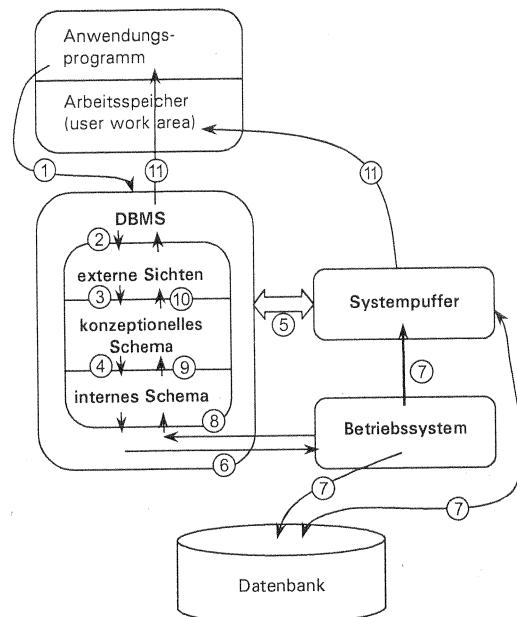
Das DBMS ist ein Softwarepaket, welches die Verwaltung der Datenbank übernimmt und alle Zugriffe darauf regelt. Als Blackbox betrachtet, nimmt das DBMS die Benutzeranfragen entgegen, ermittelt die angefragten Daten aus der Datenbank und liefert sie dem Benutzer bzw. dem Anwendungsprogramm zurück. Dabei vollzieht das DBMS folgende Arbeitsschritte:

- ✓ Das DBMS empfängt die Anfragen, in denen Daten einer bestimmten externen Sicht angefordert werden.
- ✓ Es liest die Definition der angeforderten Sicht und überprüft die Syntax der Anfrage.
- ✓ Nun überprüft es, ob der Benutzer die Rechte besitzt, auf die Daten zuzugreifen.
- ✓ Mithilfe der Transformationsregeln, die zwischen der externen Sicht und dem konzeptionellen Schema gelten, werden die benötigten Datenobjekte ermittelt.
- ✓ Über die Transformationsregeln, die zwischen dem konzeptionellen Schema und internen Schema angewandt werden, ermittelt das DBMS die physischen Datenobjekte und die Zugriffspfade.
- ✓ Das DBMS beauftragt das Betriebssystem zum Lesen der ermittelten Speicherbereiche.
- ✓ Das Betriebssystem legt diese gelesenen Blöcke im Systempuffer des DBMS ab.
- ✓ Die gelesenen Daten werden über die Anwendung der Transformationsregeln in entgegengesetzter Richtung umgewandelt. Dabei wird die gewünschte Auswahl der Daten zusammengestellt.
- ✓ Das DBMS stellt sicher, dass die übergebenen Daten für andere Benutzer so lange gesperrt sind, bis die Bearbeitung der Daten beendet wird.
- ✓ Die gesuchten Daten (im externen Format) werden an das Anwendungsprogramm bzw. den Benutzer übergeben.

Diese Schrittfolge wird in der folgenden Grafik dargestellt. Dabei werden auch technische Aspekte betrachtet.

## Grundlagen zu Datenbanken

- ① Eine Anfrage wird an das DBMS gestellt.
- ② Der Befehl wird analysiert und die zugehörige externe Sicht wird ermittelt.
- ③ Über die Transformationsregeln wird das konzeptionelle Schema ermittelt.
- ④ Über Transformationsregeln wird das interne Schema ermittelt.
- ⑤ Ein Teil der Daten wird im Systempuffer gehalten. Das DBMS prüft, ob sich die angeforderten Daten im Systempuffer befinden.
- ⑥ Sind die Daten nicht im Systempuffer, müssen sie über das Betriebssystem dorthin geladen werden.
- ⑦ Das Betriebssystem tauscht die vorhandenen Daten (Datenseiten - Pages) durch die angeforderten Daten aus und speichert gegebenenfalls geänderte Daten in der Datenbank.
- ⑧ Das Betriebssystem informiert das DBMS über die Bereitschaft der angeforderten Daten.
- ⑨ - ⑩ Die gewünschten Daten werden über die Transformationsregeln in das Format der betreffenden Sicht umgewandelt.
- ⑪ Das DBMS übergibt die angeforderten Daten und die Statusinformationen an die Anwendung.



Bearbeitung einer Datenbankabfrage

## Weitere Aufgaben des DBMS

### ✓ Sicherung der Integrität (in sich richtige und widerspruchsfreie Daten):

Durch die Anwendung der im konzeptionellen Schema vorgegebenen Integritätsbedingungen kann die logische Richtigkeit (Konsistenz) der Daten (entsprechend den Zusammenhängen in der Praxis) gewahrt werden.

Beispiel: In einer Bankanwendung wird von einem Konto A ein bestimmter Betrag abgebucht. Damit der Gesamtbetrag für alle Konten stimmt, muss dieser Betrag einem anderen Konto B gutgeschrieben werden. Die Änderung eines Kontos muss also mit der gleichzeitigen Änderung eines zweiten Kontos verbunden sein.

### ✓ Datensicherung (Recovery)

Das DBMS ist in der Lage, nach einem Systemabsturz, einem Absturz der Anwendung oder anderen Fehlern die Datenbank wieder in einen konsistenten Zustand zu überführen. Zu diesem Zweck verfügt das DBMS meist über ein internes Logbuch.

### ✓ Synchronisation

Meist arbeiten mehrere Benutzer gleichzeitig mit einer Datenbank. Das DBMS hat hier die Aufgabe, parallel ablaufende Transaktionen (Folge von Lese- und Schreib-Operationen) der Benutzer zu synchronisieren, d. h. die Zugriffe so zu verwalten, dass die Integrität der Datenbank gewahrt bleibt. Diese Aufgabe wird vom integrierten Transaktions-Manager übernommen.

Beispiel: Solange das Anwendungsprogramm des Personalbüros die Daten von Frau Maier bearbeitet, kann kein anderer Benutzer mit diesem Datensatz arbeiten. Erst wenn der Datensatz gespeichert wurde, können andere wieder darauf zugreifen. Es gibt aber Einstellungen für das DBS, die diesen Schutzmechanismus ausschalten bzw. dessen Reaktionen steuern.

### ✓ Datenschutz:

Einige Daten, wie beispielsweise die Gehälter der Angestellten, dürfen nur für bestimmte Personenkreise zugänglich sein. Das DBMS bietet die Mittel dafür, dass der Datenbankadministrator entsprechende Zugriffsrechte für jeden Benutzer bzw. für Benutzergruppen festlegen kann.

## Weitere Komponenten des DBMS

Meist besitzen Datenbanksysteme noch weitere Komponenten:

### ✓ Data Dictionary/Repositories

Das Data Dictionary (Datenlexikon, -wörterbuch; auch als Meta-Datenbank oder Katalog bezeichnet) dient der Speicherung von Informationen über die Daten der Datenbank und deren Verwaltung. Es werden darin beispielsweise das Datenbankschema, die Sichten und die Zugriffsrechte auf die Datenbank abgelegt. Der Anwender kann über das Dictionary Informationen über die Datenbank erhalten und Leistungsanalysen durchführen lassen.

In großen DBS werden Repositories verwendet, welche umfangreicher sind als Data Dictionaries. Sie enthalten zusätzlich zu den Informationen des Dictionarys noch Informationen über die Benutzer und die Anwendungsprogramme.

Der Inhalt des Data Dictionarys bzw. des Repositorys ist stark vom DBS-Hersteller abhängig.

### ✓ Logbuch

Datenbanksysteme verfügen über ein Logbuch, in welchem Informationen über die Transaktionsvorgänge verzeichnet sind, wie beispielsweise der Beginn und das Ende der Transaktion und der Zustand der Daten zu Beginn der Transaktion. Treten Systemfehler auf, werden die Informationen des Logbuchs zum Wiederherstellen der Datenbank verwendet.

Größere Datenbanksysteme bieten meist noch zusätzliche Komponenten, die den Anwender bzw. den Anwendungsprogrammierer bei seiner Arbeit unterstützen:

<b>Entwurfswerkzeuge zum Datenbankentwurf</b>	Entwurfswerkzeuge zum Datenbankentwurf unterstützen den Anwender beim Entwurf der Datenbank, sodass er nicht auf die Anwendung der Datendefinitionssprache (DDL) angewiesen ist.
<b>Abfrage-Generatoren</b>	Abfrage-Generatoren ermöglichen dem Anwender das Erzeugen von Datenbankabfragen auch ohne Kenntnisse der Datenbank-Abfragesprache.
<b>Report-Generatoren</b>	Report-Generatoren erzeugen Berichte über Datenbankinhalte in den verschiedenen Formen (z. B. Tabellen mit Kopf- und Fußzeilen und Zwischensummen).
<b>Tools zur Erstellung von Business-Grafiken</b>	Tools zur Erstellung von Business-Grafiken ermöglichen die grafische Darstellung von Daten der Datenbank in Diagramm-Form.
<b>CASE-Werkzeuge</b>	CASE-Werkzeuge (Computer Aided Software Engineering - computergestützter Softwareentwurf) dienen dem Entwurf von Datenbankanwendungen, wobei der Quellcode der Anwendung automatisch generiert wird.
<b>Utilities zur Fehleranalyse</b>	Utilities zur Fehleranalyse helfen dem Anwender, Fehler in der Datenbankstruktur aufzufinden und zu beseitigen.
<b>Funktionen zur Komprimierung und Reorganisation der Datenbank</b>	Funktionen zur Komprimierung und Reorganisation der Datenbank sind notwendig, wenn häufig Daten gelöscht und geändert wurden, da nicht mehr benötigter Speicherplatz nicht automatisch freigegeben wird. Bei der Ausführung der Funktionen wird die Datenbank reorganisiert und nicht benötigter Speicher freigegeben.
<b>Archivierungsfunktionen</b>	Archivierungsfunktionen werden für das Kopieren und Archivieren von Datenbeständen der Datenbank eingesetzt.

## 2.4 Physische Datenbankarchitektur

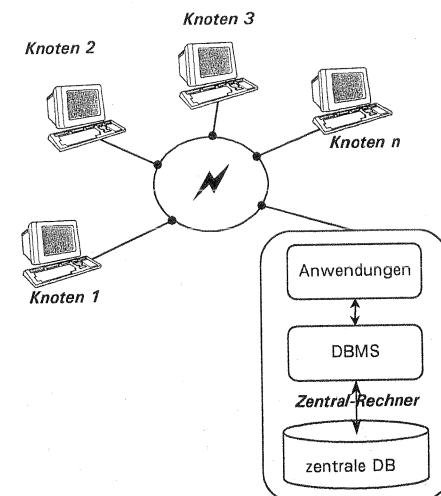
Die physischen Konzepte der Architektur von DBS ergeben sich aus dem logischen Konzept der Datenbank-Entwicklung (3-Ebenen-Architektur) in Verbindung mit der Rechnerumgebung. Dabei werden verschiedene Betrachtungsweisen angewendet. Zum einen werden **zentralisierte DBS** und **verteilte DBS** unterschieden, bei denen die lokale Anordnung der Datenbanken und der Rechner ausschlaggebend sind. Zum anderen führt die zentrale Speicherung der Daten auf einem Rechner (dem Server), von dem aus die Arbeitsplatzrechner (die Clients) auf die Daten zugreifen können, zum **Client/Server-Konzept**. Ein anderer Ansatz ist der Einsatz von Parallelrechnern und Multiprozessorsystemen, der auch von DBS genutzt werden kann (**parallele DBS**).

### Zentralisierte DBS

In einem zentralisierten DBS werden das gesamte DBMS und die Anwendungen auf einem Rechner abgelegt, der als zentraler Verwaltungsrechner bzw. Zentralrechner (auch Host oder Mainframe) bezeichnet wird. An den anderen Standorten befinden sich "dumme" Terminals, die nur der Ein- und Ausgabe dienen (wenig eigene Funktionalität). Von diesen Terminals aus haben alle Benutzer die gleichen Sichten auf die Datenbank, die von den auf dem Zentralrechner laufenden Anwendungen erzeugt wird.

Die Datenbank eines zentralisierten DBS ist im Vergleich zu verteilten Datenbanken relativ einfach zu administrieren. Bezuglich Antwortzeiten und Ausfallsicherheit kann es aber auf dem Zentralrechner zu Problemen kommen.

In modernen Rechnernetzen werden als Endgeräte "intelligente" Arbeitsplatzrechner eingesetzt. Die Datenbankanwendungen und die Client-Software der DBS können zusammen auf diesem Rechner laufen. Damit kann dem zentralisierten DBS ein Teil der Arbeit abgenommen werden, indem dort z. B. die Syntaxprüfung und die Optimierung der Abfragen durchgeführt werden.



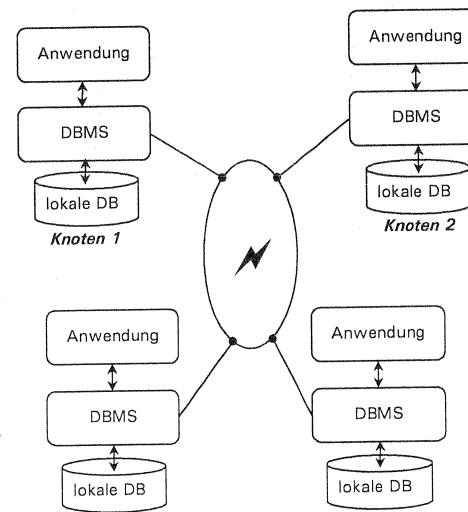
Netz mit Terminals und zentralisierter Datenbank

### Verteilte DBS

Um die Vorteile vernetzter Rechnersysteme (z. B. schneller Datenaustausch ohne externe Datenträger), die geografisch weit voneinander entfernt sein können, auch auf dem Gebiet der Datenbanken effektiv ausnutzen zu können, wurden und werden neue Methoden gesucht.

**Verteilte Datenbanken** sind eine Menge von mehreren logisch zusammengehörigen (Teil-)Datenbanken, die in einem Netz auf mehreren lokal getrennten Computern (z. B. in verschiedenen Städten) gespeichert sind. Ein **verteiltes DBMS** besitzt Mechanismen zur Zusammenführung und Abfrage der verteilten Datenbanken. Durch das DBMS wird die Verteilung der Daten vor dem Anwender verborgen. Dem Benutzer erscheint es wie ein zentralisiertes DBS, da er nur auf der externen Ebene arbeitet.

Beispielsweise besitzen die Filialen einer Bank jeweils die Kundendaten ihrer Kunden, die Zentrale kann aber auf alle Kundendaten zugreifen.



Verteiltes DBS

Verteilte DBS haben im Vergleich zu zentralisierten DBS entscheidende Vorteile:

<b>Lokale Autonomie</b>	Lokale Autonomie ermöglicht effektivere Anfragen, da die Daten dort gespeichert sind, wo sie gebraucht werden (besonders bedeutsam bei Unternehmen mit dezentraler Unternehmensstruktur).
<b>Zuverlässigkeit und Verfügbarkeit</b>	Die Verfügbarkeit wird verbessert, da der Ausfall eines Knotens nicht zum Ausfall des gesamten Systems führt. Gezielte Redundanz erhöht die Zuverlässigkeit.
<b>Leistung</b>	Die Leistung wird durch Parallelarbeit an verschiedenen Orten erhöht. Zugriffe können gleichzeitig durchgeführt und die Zugriffsposition genauer festgelegt werden, da die lokalen Datenbanken kleiner sind.
<b>Erweiterbarkeit</b>	Eine Erweiterbarkeit des Systems, wie z. B. das Hinzufügen eines neuen Knotens, wird auf relativ einfacher Weise ermöglicht.

Die Anwendung verteilter DBS bringt aber auch Nachteile mit sich:

<b>Mangel an praktischer Erfahrung</b>	Auch wenn umfangreiche theoretische Studien zu verteilten Datenbanken vorliegen, gibt es einen Mangel an praktischer Erfahrung auf diesem Gebiet.
<b>Komplexität</b>	Die Komplexität der Aufgaben (Synchronisation, Bearbeitung von Anfragen usw.) ist fast immer sehr hoch.
<b>Dezentrale Verwaltung</b>	Eine dezentrale Verwaltung bringt zusätzlichen Aufwand für die Verwaltung und die Synchronisation mit sich.
<b>Sicherheit</b>	Die Sicherheit ist zu gewährleisten, d. h. sowohl die Datensicherheit der lokalen Datenbanken als auch die Sicherheit im Netz (z. B. bei Datenübertragungen, Zugriffen auf Daten usw.).
<b>Kosten</b>	Es entstehen Kosten vor allem für die Software und die Kommunikation.
<b>Übergang von zentralisierten auf verteilte DBS</b>	Der Übergang von zentralisierten auf verteilte DBS verursacht Kosten durch den Umstieg auf neue Software. Auch im Bereich der Hardware können Kosten entstehen, z. B. für eine neue Kommunikationsinfrastruktur. Häufig ist neues Personal bzw. die Schulung des vorhandenen Personals erforderlich. Es werden gegebenenfalls Werkzeuge zur Überführung der vorhandenen Datenbanken in das neue System benötigt.

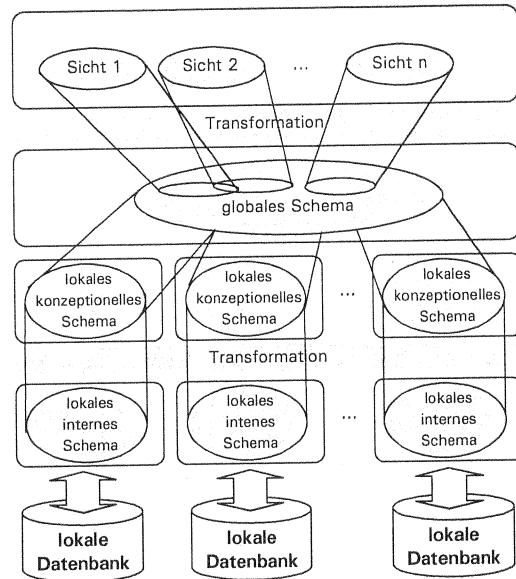
Die Verteilung eines Datenbanksystems kann unter verschiedenen Aspekten erfolgen:

- ✓ Ein DBS kann **homogen** verteilt sein, d. h., logisch zusammengehörige Datenbanken werden auf verschiedene Orte verteilt. Sie werden von derselben Datenbanksoftware verwaltet.
- ✓ Sollen unterschiedliche Datenbanken, z. B. aus verschiedenen Unternehmen, Unternehmensteilen oder Abteilungen, zusammen verwaltet und soll auf die Daten mehrerer unterschiedlicher Datenbanken gleichzeitig zugegriffen werden, ist das DBS **heterogen** verteilt. Die Datenbanken können auch verschiedene Datenmodelle besitzen. Heterogen verteilte DBS werden auch als Multidatenbanksysteme (MDBS) bezeichnet.
- ✓ Ein weiterer Punkt, nach dem eine Unterscheidung von verteilten DBS möglich ist, ist die Verwendung replizierter Daten (dieselben Daten befinden sich an verschiedenen Stellen und werden laufend abgeglichen).

Das 3-Ebenen-Modell muss für verteilte DBS erweitert werden. Die Aufteilung der Datenbanken auf die einzelnen Rechner muss darin deutlich werden, ebenso wie der Unterschied von homogen oder heterogen verteilten Datenbanken. Die Änderungen betreffen vor allem die konzeptionelle Ebene. Auf externer Ebene ist kein Unterschied zu zentralisierten DBS zu erkennen.

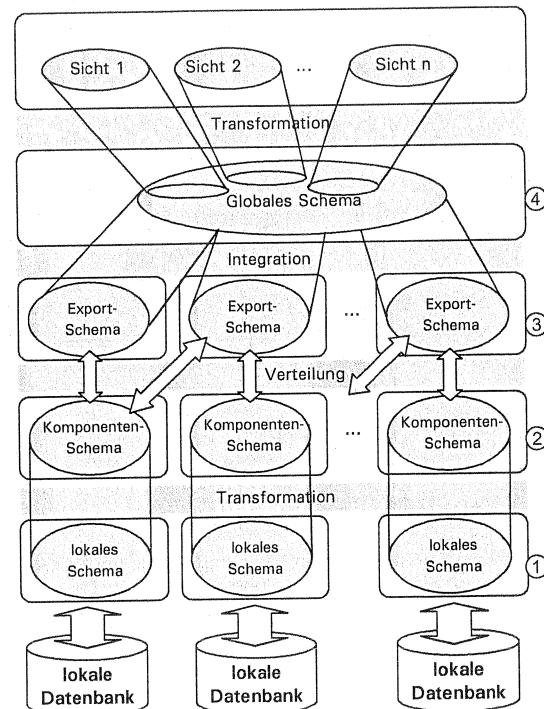
### ✓ Homogene Verteilung

In der konzeptionellen Ebene findet eine Aufteilung des konzeptionellen Schemas statt. Es wird ein gemeinsames konzeptionelles Schema für die gesamte Datenbank erstellt, auf dem die externen Sichten beruhen. Das Gesamtschema wird in lokale konzeptionelle Schemen unterteilt, für jede lokale Datenbank ein Schema. Für jedes lokale konzeptionelle Schema existiert dann ein lokales internes Schema.



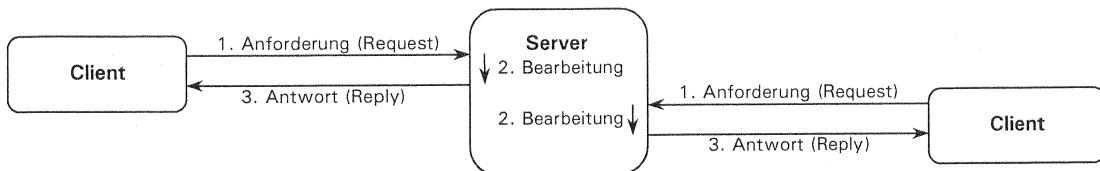
### ✓ Heterogene Verteilung

Die Zusammenführung mehrerer unterschiedlicher Datenbanken bringt einen höheren Aufwand mit sich. Die Daten aus den lokalen Schemen ① müssen zunächst in ein gemeinsames Datenbankschema transformiert werden. Ein transformiertes lokales Schema wird im Komponenten-Schema ② gespeichert. Die Daten aus den Komponenten-Schemen werden aufgeteilt und in den Exportschemen ③ abgelegt. Ein Exportschema kann sich aus Teilen eines oder mehrerer Komponenten-Schemen zusammensetzen. Aus einem Komponenten-Schema können aber auch mehrere Exportschemen erzeugt werden. Die Exportschemen werden wiederum im gemeinsamen globalen Schema ④ abgebildet.



## Client-Server DBS

Die meisten heutigen DBS arbeiten nach dem Client-Server-Konzept. Es realisiert ein funktional verteiltes System, in dem zwei unabhängige Prozesse über eine definierte Schnittstelle miteinander kommunizieren (als Prozess wird hier ein eigenständig laufendes Programm verstanden) - der Server und der Client. Die Kommunikation erfolgt über einen Anforderung-Antwort-Dialog. Der Client stellt eine Anforderung an den Server, der Server bearbeitet die Anforderung und gibt die gewünschte Antwort an den Client zurück. Der Server stellt also die Dienstleistungen zur Verfügung und der Client nimmt sie in Anspruch. Häufig werden die Dienste eines Servers von mehreren Clients - auch gleichzeitig - genutzt.



Die Initiative geht hierbei immer vom Client aus, der Server verharrt so lange in Warteposition, bis ein Client seine Anforderungen sendet.

- ✓ Client und Server können sich physisch sowohl auf dem gleichen Rechner befinden als auch auf verschiedenen Rechnern, die über ein Netzwerk verbunden sind.
- ✓ Ein Rechner (bzw. ein Programm) kann sowohl als Client als auch als Server arbeiten. Es ist von der momentanen Tätigkeit abhängig, ob gerade ein Dienst in Anspruch genommen oder bereitgestellt wird.

Bezogen auf ein DBS ist die einfachste Form eines Client-Server-Systems die, dass ein Datenbank-Anwendungsprogramm die Dienste eines Datenbank-Servers in Anspruch nimmt, z. B. durch eine Datenbankanfrage. Der Server führt die Anfrage über die Datenbank aus und sendet die selektierte Datenmenge an den Client, also das Anwendungsprogramm, zurück.

Bei relationalen DBS werden diese Abfragen meist in der Datenbank-Abfragesprache SQL formuliert. Der Server wird entsprechend als SQL-Server bezeichnet. Der SQL-Server (entspricht dem DBMS oder einem Teil davon) verwaltet die Datenbank und bearbeitet die SQL-Anfragen der Clients. Der Client ist beispielsweise eine Datenbankanwendung, welche die Anzeige und Bearbeitung der Daten ermöglicht, die ihm der SQL-Server liefert hat.

## Parallele DBS

Parallele Datenbanksysteme laufen auf Multiprozessorsystemen oder Parallelrechnern und nutzen gleichzeitig die Leistung mehrerer Prozessoren. Damit werden eine Leistungssteigerung und eine Verkürzung der Bearbeitungszeit bei Datenbankanfragen und Transaktionen erreicht. Bei großen Datenbanken mit vielen Benutzern verursacht die sequenzielle Verarbeitung von Abfragen zum Teil inakzeptable Antwortzeiten. Besonders beim Einsatz von objektorientierten Datenbanksystemen, die häufig mit komplex strukturierten Objekten arbeiten, ist ein erhöhter Aufwand an Rechenzeit notwendig, der Parallelverarbeitung erforderlich macht.

Unter einem parallelen DBS ist nicht zu verstehen, dass mehrere Benutzer gleichzeitig Anfragen an ein DBMS richten können und diese Anfragen dann zeitlich versetzt, aber gewissermaßen parallel abgearbeitet werden. Diese Arbeitsweise ist auch bei sequenziell arbeitenden DBS üblich. Eine echte Parallelarbeit wird durch den Einsatz von Parallelrechnern oder durch die Anwendung paralleler Algorithmen erreicht, die gleichzeitig auf mehreren Prozessoren ausgeführt werden.

In einem parallelen System sind mehrere Prozessoren, Platten- und Hauptspeicher über eine sehr schnelle Leitung (Hochgeschwindigkeitsnetz) miteinander verbunden. Die Arbeitsweise paralleler Datenbanksysteme hängt von der konkreten Rechnerarchitektur ab, die grundsätzliche Arbeitsweise ist aber die gleiche:

- ✓ Die Daten werden auf die verfügbaren Platten verteilt.
- ✓ Datenbankabfragen und Transaktionen werden so zerlegt, dass sie auf mehreren Prozessoren gleichzeitig abgearbeitet werden können.

## Grundlagen zu Datenbanken

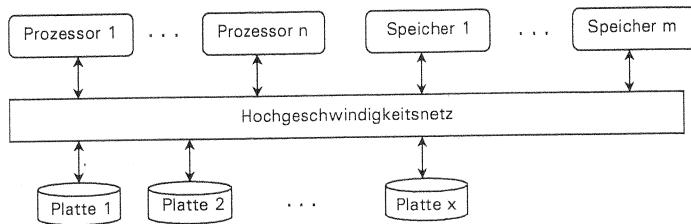
Prinzipiell gibt es drei verschiedene Architekturtypen für parallele Systeme:

✓ **Shared-Memory-Architektur**

(Shared-Everything-Architektur)

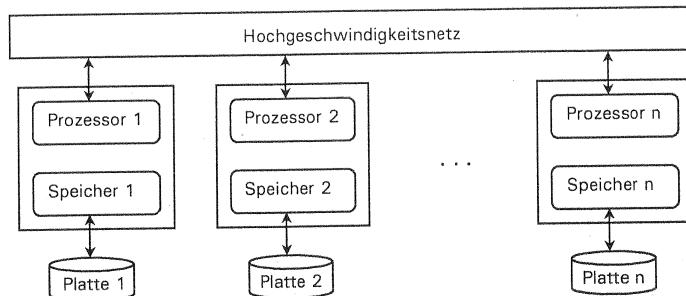
Alle Prozessoren des Systems können auf den gemeinsamen Speicher zugreifen und über diesen kommunizieren.

Die für die Ausführung einer Datenbankoperation benötigten Daten werden von den ausführenden Prozessoren über das Netzwerk angefordert und im Speicher bereitgestellt.



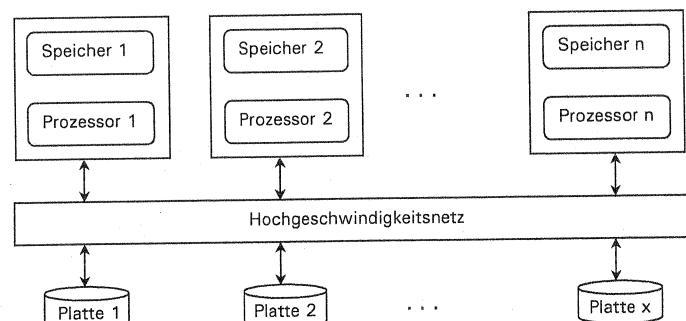
✓ **Shared-Nothing-Architektur**

Jedem Prozessor sind eigene Speichermedien (Haupt- und Plattenspeicher) zugeordnet, auf die er exklusiv zugreift. Bei diesem System verfügt jeder Prozessor über eine Kopie des DBMS. Eine wichtige Arbeitsweise dieser Architekturform ist das Weitergeben von Funktionen an einen anderen nicht ausgelasteten Prozessor.



✓ **Shared-Disk-Architektur**

Die Hauptspeicher sind den Prozessoren lokal zugeordnet, die Plattspeicher werden aber gemeinsam genutzt. Die Arbeitsweise ist ähnlich wie bei der Shared-Nothing-Architektur. Die Daten werden aber wie bei der Shared-Memory-Architektur über das Netzwerk angefordert.



## 2.5 Schnellübersicht

Was bedeutet ... ?	
<b>Client-Server-DBS</b>	Die meisten heutigen Datenbanksysteme arbeiten nach dem Client-Server-Konzept, bei dem der Datenbank-Server seinen Clients Dienste zur Verfügung stellt. Beispielsweise werden Datenbankanfragen an den Datenbank-Server gerichtet und dieser liefert die entsprechenden Daten an den Client zurück.
<b>Data Dictionary</b>	Speichert Informationen über die Datenbank (Sichten, Datenbankschema usw.) und deren Verwaltung (Zugriffsrechte usw.)
<b>Datenbank (DB)</b>	Sammlung logisch zusammengehöriger Daten, die physisch zusammenhängend auf einem externen permanenten Speichermedium abgelegt sind
<b>Datenbankmanagementsystem (DBMS)</b>	Datenbanksoftware, mit der die Daten der Datenbank und die Datenbank selbst erstellt, bearbeitet, verwaltet und gepflegt werden
<b>Datenbanksystem (DBS)</b>	Kombination aus den Datenbanken und dem Datenbankmanagementsystem
<b>Datenmodell</b>	Hilfsmittel zur Abstraktion der Daten aus der realen Welt. Es wird eine Struktur aus den relevanten Daten, deren Beziehungen und Bedingungen erzeugt.
<b>Datenunabhängigkeit</b>	Die Anwendungsprogramme und die physische Speicherung der verwendeten Daten sind voneinander unabhängig. Die Datenverwaltung wird von einem anderen Programm (z. B. einem DBMS) übernommen.

<b>Was bedeutet ...?</b>	
<b>hierarchisches DBS</b>	Die Daten werden nach streng hierarchischem Prinzip in einer baumartigen Struktur abgelegt.
<b>Integrität</b>	Liegt vor, wenn Daten in sich richtig (stimmig), widerspruchsfrei und vollständig sind (logische Integrität). Referenzielle Integrität → siehe Konsistenz
<b>Integritätsbedingungen</b>	Integritätsbedingungen sind Bestimmungen, die eingehalten werden müssen, um die Korrektheit und die logische Richtigkeit der Daten zu sichern.
<b>Konsistenz/Inkonsistenz</b>	Konsistenz (referenzielle Integrität) ist die Übereinstimmung von mehrfach gespeicherten Daten. Werden bei Änderungen nicht alle mehrfach gespeicherten Daten geändert, ist der Datenbestand inkonsistent, d. h., es existieren unterschiedliche Versionsstände der gleichen Daten.
<b>Logbuch</b>	Enthält Informationen über Transaktionen und wird für die Wiederherstellung der Datenbank nach Systemabstürzen verwendet
<b>Netzwerk-DBS</b>	Gleichartige Daten werden in Recordsets abgelegt, die Beziehung zwischen den Daten in Sets. In der grafischen Darstellung entsteht ein gerichteter Graph, der als Netzwerk bezeichnet wird.
<b>objektorientiertes DBS (OODBS)</b>	Die Daten und die Operationen, die auf den Daten ausgeführt werden können, werden in Form von Objekten gespeichert.
<b>objektrelationales DBS (ORDBS)</b>	Vereinigt Vorteile des relationalen und des objektorientierten DBMs. Daten werden als Objekte gespeichert, Zugriff erfolgt über die erweiterte SQL-Sprache.
<b>Dokumentorientierte Datenbank</b>	Die Daten werden in einzelnen Dokumenten gespeichert, die keinem festen Schema unterliegen.
<b>Spaltenorientierte Datenbank</b>	Durch die spaltenweise Speicherung der Daten der Relationen auf einem externen Speichermedium sind schnellere Leseoperationen bei der Datenauswertung möglich.
<b>NoSQL-Datenbank</b>	Unter NoSQL-Datenbanken werden alternative Ansätze der Datenspeicherung zur relationalen Datenbank verstanden. Typische Vertreter sind dokumentenorientierte Datenbanken, Graphen-Datenbanken und Key/Value-Datenbanken.
<b>Graphen-Datenbank</b>	Die Daten der Datenbank werden als Knoten dargestellt und die Beziehungen zwischen ihnen durch die Verbindungen zwischen diesen Knoten.
<b>Key/Value-Datenbank</b>	In Key/Value-Datenbanken werden die Daten in Paaren aus Schlüsseln und Werten gespeichert. Einem Schlüssel ist jeweils ein Wert (Einzelwert, Liste oder Set) zugeordnet.
<b>paralleles DBS</b>	Läuft auf Parallelrechnern oder Multiprozessorsystemen. Die (echte) Parallelarbeit wird durch den Einsatz von Parallelrechnern oder die Anwendung paralleler Algorithmen, die gleichzeitig auf mehreren Prozessoren ausgeführt werden (z. B. auf Multiprozessorsystemen), erreicht. Dadurch werden die Antwortzeiten bei Datenbankanfragen und Transaktionen verkürzt.
<b>Redundanz</b>	Mehrfache Speicherung von gleichen Daten. Dadurch erhöht sich das Risiko inkonsistenter Daten.
<b>relationales DBS (RDBS)</b>	Die Daten werden in Tabellenform gespeichert. Zwischen den Tabellen können Beziehungen (Relationen) definiert werden. Die meistverwendete Anfragesprache ist SQL.
<b>Repository</b>	Wie ein Data Dictionary aufgebaut, speichert aber noch zusätzliche Informationen, z. B. über Benutzer und Anwendungsprogramme
<b>Sicht</b>	Ausschnitt (Teilmenge) einer Datenbank, der die für eine Anwendung relevanten Daten enthält
<b>Transaktion</b>	Als Transaktion werden mehrere aufeinanderfolgende Lese- und Schreibzugriffe auf eine Datenbank bezeichnet, die in einem logischen Zusammenhang stehen. Diese werden entweder vollständig oder gar nicht ausgeführt.

<b>Was bedeutet ...?</b>	
<b>verteiltes DBS</b>	Die Datenbanken sind auf geografisch getrennt stehende Rechner verteilt. Auf jedem dieser Rechner läuft das DBMS, welches über Mechanismen zur Zusammenführung der verteilten Datenbank verfügt. Es gibt homogen und heterogen verteilte DBS.
<b>zentralisiertes DBS</b>	Die Datenbank, die Datenbanksoftware und die Datenbankanwendungen laufen auf einem zentralen Rechner. Die Terminals arbeiten über ein Netzwerk mit den Anwendungsprogrammen.
<b>3-Ebenen-Modell (auch 3-Ebenen-Architektur) nach ANSI-SPARC</b>	Dieses Modell besteht aus drei unterschiedlichen Abstraktionsebenen für die Darstellung des Datenbankschemas: der internen, der konzeptionellen und der externen Ebene. Auf jeder Ebene wird eine andere Sichtweise auf die Daten verwendet: die physische Datenorganisation, die logische Gesamtsicht und die logische Benzersicht der Daten.

## 2.6 Übung

### Fragen zur Datenbanktheorie

Übungsdatei: --

Ergebnisdatei: Antworten.doc

- ① Nennen Sie wichtige Gründe, die zur Entwicklung von Datenbanksystemen führten.
- ② Welche Datenbankmodelle kennen Sie? Wodurch sind sie gekennzeichnet?
- ③ Nennen Sie die Namen der 3 Ebenen des 3-Ebenen-Modells und geben Sie an, was in jeder Ebene dargestellt wird.
- ④ Was ist ein Datenbankmanagementsystem? Welche Aufgaben hat es?
- ⑤ Was ist ein Data Dictionary und wozu wird es benötigt?
- ⑥ Welche physischen Datenbankarchitekturen kennen Sie? Erläutern Sie jeweils kurz den Aufbau.