

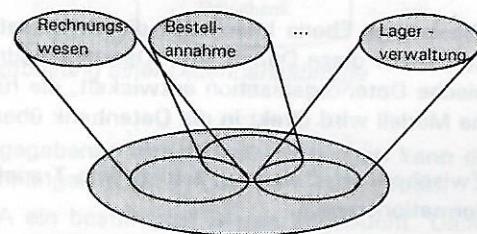
## 1.4 Aufbau und Organisation einer Datenbank

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 die für eine Anwendung bzw. ein Problem relevanten Daten enthält.

### Beispiel

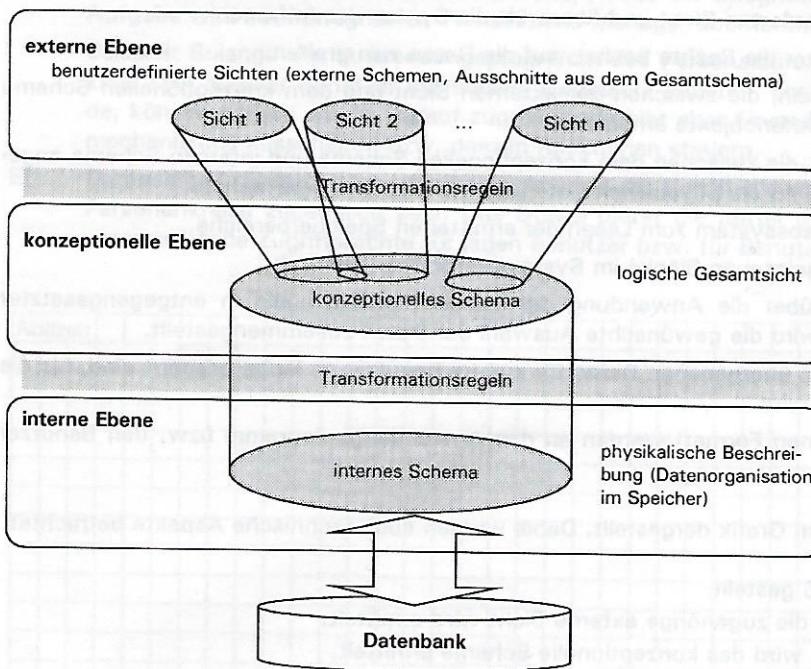
Ein Unternehmen des Großhandels besitzt eine große Datenbank, die die Artikeldaten, die Daten der Lagerverwaltung, die Kunden- und die Daten der Bestellannahme und Rechnungslegung enthält. In den verschiedenen Abteilungen werden nur Teile der Datenbank, bestimmte Sichten, benötigt. So werden bei der 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 Gesamtsicht auf die DB des Unternehmens

### 1.4.1 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.



Auf der **externen Ebene** erfolgt die Darstellung der Daten, wie sie in den einzelnen Anwendungen benötigt werden. In den Benutzersichten (kurz: Sichten) werden Teile auf die logische 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 sind nicht auf die Belange 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) 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**.

## 1.4.2 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 gefragten Daten aus der Datenbank und liefert sie dem Benutzer bzw. dem Anwendungsprogramm zurück. Dabei vollzieht das DBMS folgende Arbeitsschritte:

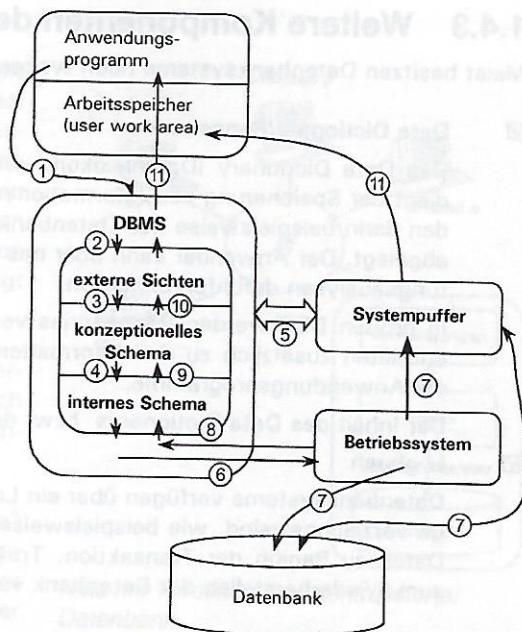
- Das DBMS empfängt die Anfragen, in denen Daten einer bestimmten externen Sicht angefordert werden.
- Es liest die Definition der angeforderten Sicht und überprüft die Syntax der Anfrage.
- Nun überprüft es, ob der Benutzer die Rechte besitzt, auf die Daten zuzugreifen.
- Mithilfe der Transformationsregeln, 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.

- 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.

Durch das DBMS wird dabei noch eine Reihe von weiteren Aufgaben ausgeführt.



Bearbeitung einer Datenbankabfrage

**Integrität** (in sich richtige und widerspruchsfreie Daten):

Durch die Anwendung der im konzeptionellen Schema vorgegebenen Integritätsbedingungen kann die logische Richtigkeit 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 dann 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.

Notizen

### 1.4.3 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 Datenbank-Schema, 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 Datenbank-Entwurf</b>	Entwurfswerkzeuge zum Datenbank-Entwurf 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 Datenbank-Abfragen auch ohne Kenntnisse der Datenbank-Abfragesprache (DQL/DML).
<b>Report-Generatoren</b>	Report-Generatoren erzeugen Berichte über Datenbankinhalte in den verschiedensten 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 Datenbank-Anwendungen, wobei der Quellcode der Anwendung automatisch generiert wird.
<b>Utilities zur Fehleranalyse</b>	Utilities zur Fehleranalyse helfen dem Anwender, Fehler in der Datenbank-Struktur 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.

## 1.5 Physische Datenbankarchitektur

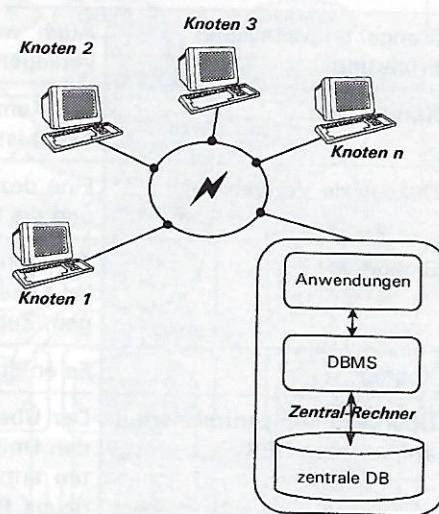
Die physischen Konzepte der Architektur von DBS ergeben sich aus dem logischen Konzept der Datenbank-Entwicklung (3-Ebenen-Architektur) in Verbindung mit der Rechnerumgebung. Dabei werden verschiedene Beobachtungsweisen angewendet. Zum einen werden **zentralisierte DBS** und **verteilte DBS** unterschieden, 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 Aspekt ist der Einsatz von Parallelrechnern und Multiprozessorsystemen, der auch von DBS ausgenutzt werden kann (**parallele DBS**).

### 1.5.1 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 Datenbank-Anwendungen 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.



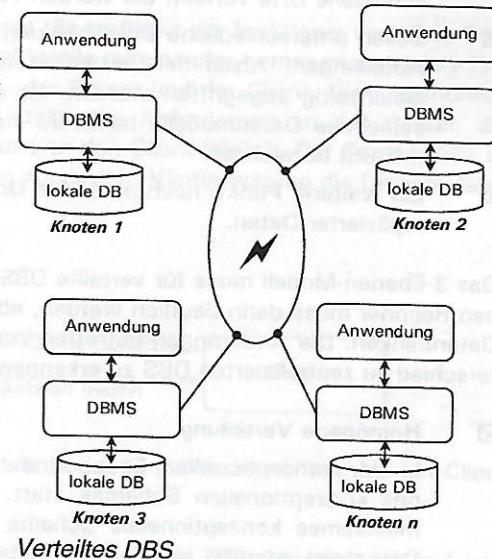
Netz mit Terminals und zentralisierter Datenbank

### 1.5.2 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 die Kundendaten ihrer Kunden, die Zentrale kann aber auf alle Kundendaten zugreifen.



Verteiltes DBS

Verteilte DBS haben im Vergleich zu zentralisierten DBS entscheidende Vorteile:

Lokale Autonomie	Lokale Autonomie ermöglicht effektive Anfragen, da die Daten dort gespeichert sind, wo sie gebraucht werden (besonders bedeutsam bei Unternehmen mit dezentraler Unternehmensstruktur).
Zuverlässigkeit und Verfügbarkeit	Die Verfügbarkeit wird verbessert, da der Ausfall eines Knotens nicht zum Ausfall des gesamten Systems führt. Gezielte Redundanz erhöht die Zuverlässigkeit.
Leistung	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.
Erweiterbarkeit	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.

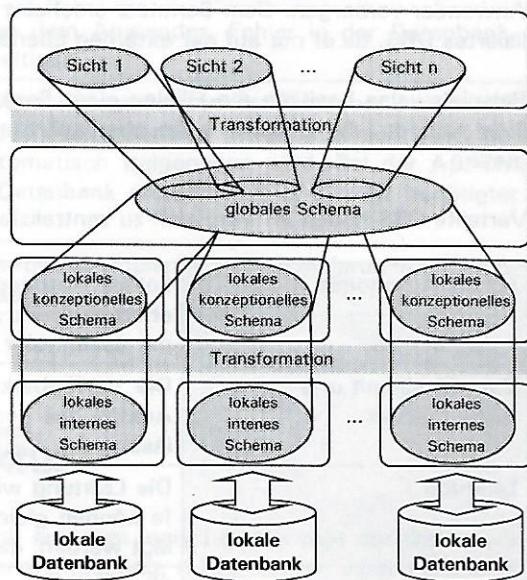
Die Verteilung eines Datenbanksystems kann unter verschiedenen Aspekten erfolgen:

- 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.

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.

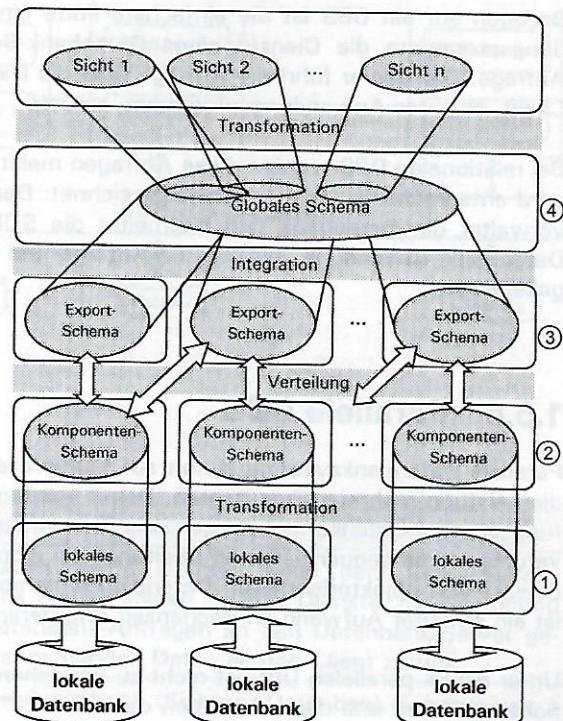
#### Homogene Verteilung

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.



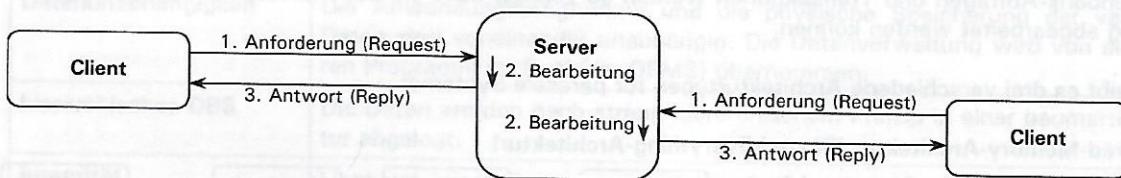
**Heterogene Verteilung**

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.



### 1.5.3 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. Programm) kann im Prinzip als Client und auch als Server arbeiten. Es ist von der momentanen Tätigkeit abhängig, ob er gerade einen Dienst für einen anderen Prozess ausführt oder selbst Dienste in Anspruch nimmt.

Notizen



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 Datenbank-Anfrage. 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 definiert. 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 Datenbank-Anwendung, welche die Anzeige und Bearbeitung der Daten ermöglicht, die ihm der SQL-Server geliefert hat.

## 1.5.4 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 Datenbank-Anfragen und Transaktionen erreicht. Bei großen Datenbanken mit vielen Benutzern verursacht die sequentielle 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 erforderlich, 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 Leistung (Hochgeschwindigkeitsnetz) miteinander verbunden. Die Arbeitsweise paralleler Datenbanksysteme hängt von der konkreten Rechnerarchitektur ab, die grundsätzliche Arbeitsweise ist aber die gleiche.

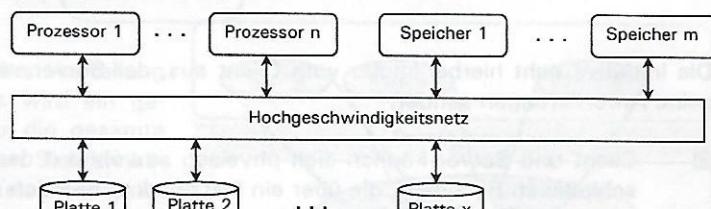
- Die Daten werden auf die verfügbaren Platten verteilt.
  - Datenbank-Abfragen und Transaktionen werden so zerlegt, dass sie auf mehreren Prozessoren gleichzeitig abgearbeitet werden können.

Prinzipiell gibt es drei verschiedene Architekturtypen für parallele Systeme:

- ### Shared-Memory-Architektur (Shared-Everything-Architektur)

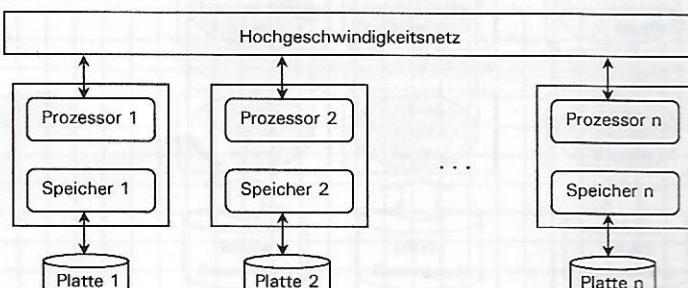
Alle Prozessoren des Systems können auf den gemeinsamen Speicher zugreifen und über diesen kommunizieren.

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



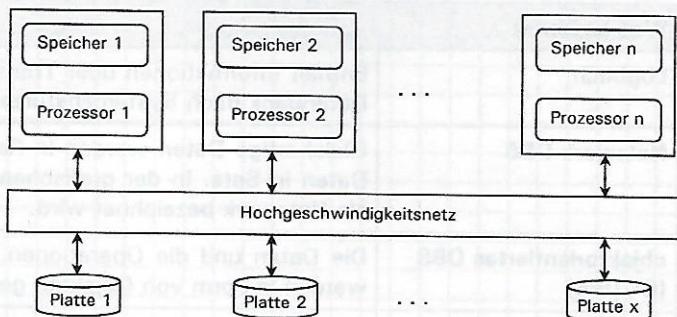
- #### Shared-Nothing-Architektur

Jedem Prozessor sind eigene Speichermedien (Haupt- und Plattspeicher) 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.



## 1.6 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 Datenbank-Anfragen 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, Datenbank-Schema 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>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). Referentielle 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 (referentielle 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.

### Notizen

Was bedeutet...	
<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>objektrelational DBS (ORDBS)</b>	Vereinigt Vorteile des relationalen und des objektorientierten DBMs. Daten werden als Objekte gespeichert, Zugriff erfolgt über die erweiterte SQL-Sprache.
<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 Datenbank-Anfragen 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 aufeinander folgende Lese- und Schreibzugriffe auf eine Datenbank bezeichnet, die in einem logischen Zusammenhang stehen. Diese werden entweder vollständig oder gar nicht ausgeführt.
<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 Datenbank-Anwendungen 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 Benutzersicht der Daten.

## 1.7 Übung

- 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.