Gliederung

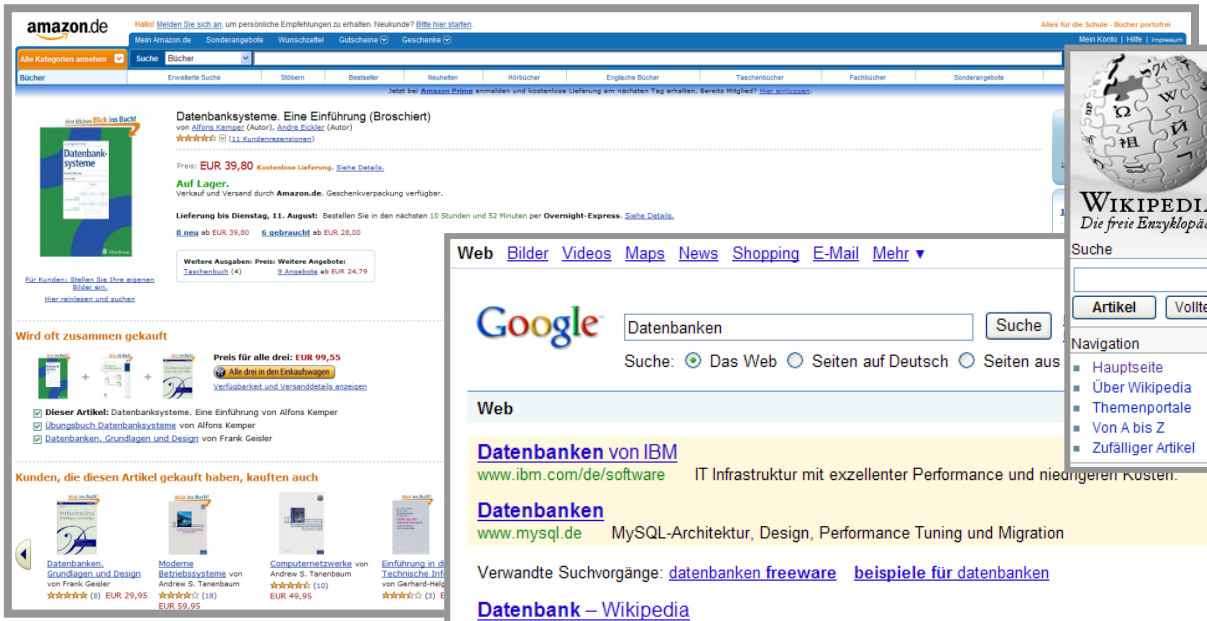
1. Einführung
2. Datenbankentwurf
3. Datenbankimplementierung
4. Physische Datenorganisation
5. Anfrageoptimierung
6. Transaktionsverwaltung
7. Datensicherheit und Wiederherstellung
8. Business Intelligence

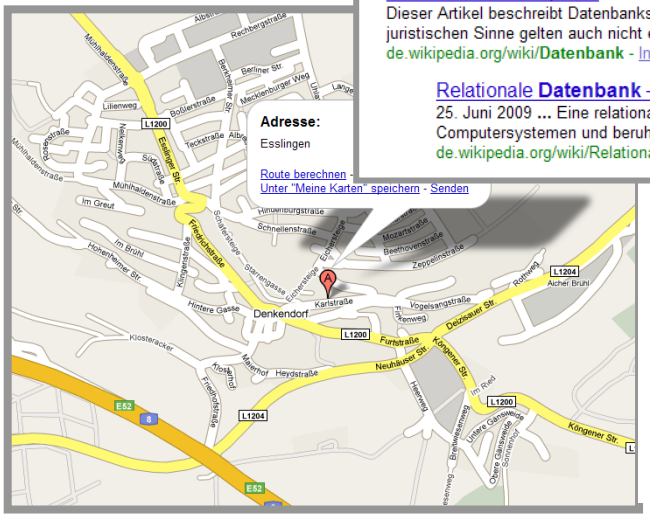
Gliederung

1. Einführung

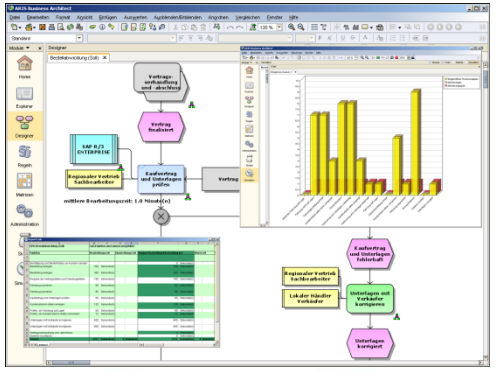
- **Gründe für die Einführung eines Datenbanksystems**
- Definition Datenbanksysteme
- Architekturen von Datenbanksystemen
- Entwicklungsgeschichte der Datenbanksysteme
- Arten von Datenbanksystemen
- Verbreitete Datenbanksysteme

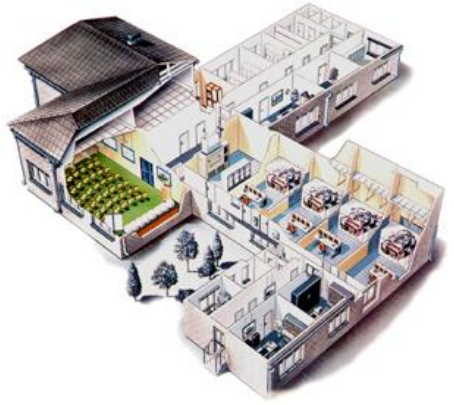
Anwendungsbeispiele aus dem Alltag

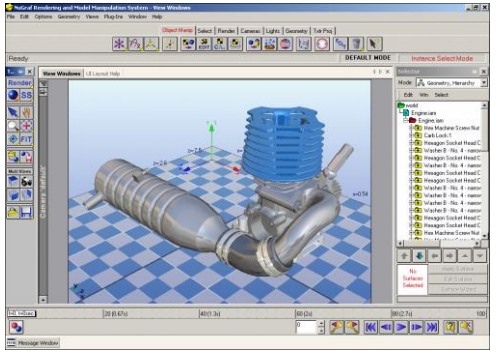




Warum … Datenbanken für Ingenieure?



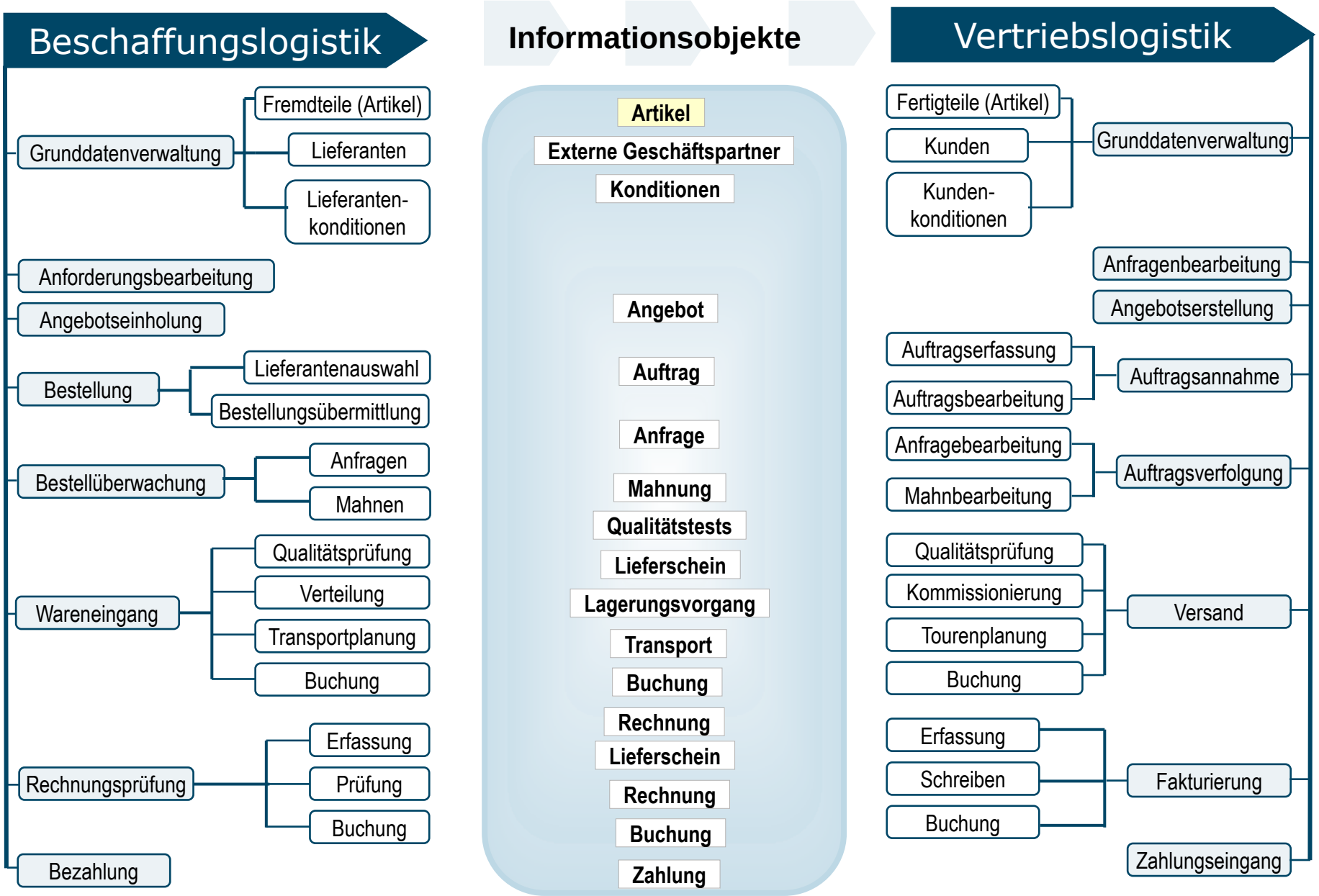


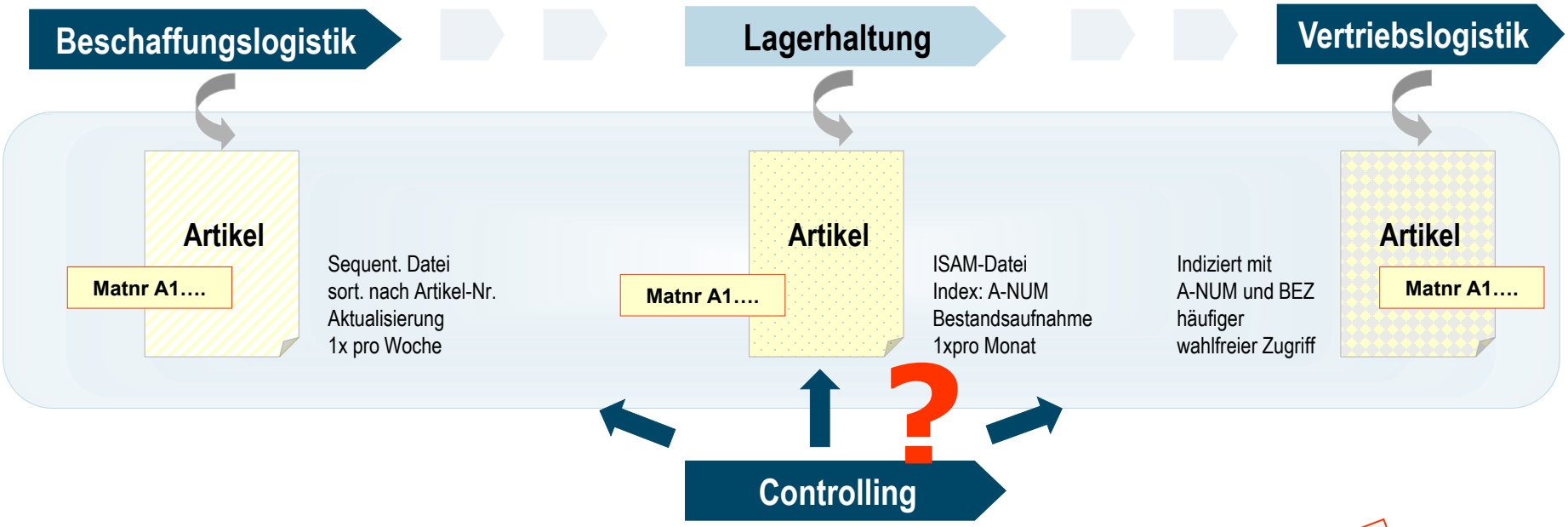


- Die Anforderungen von zahlreichen Ingenieuranwendungen sind typisch für datenbankbasierte Systeme:
  - **Große Datenmengen** für Produktmodelle
  - **Zahlreiche Mitarbeiter** (Teams von Ingenieuren), die gemeinsam diese Daten bearbeiten
  - **Hohe Anforderungen an Konsistenz, Sicherheit und Schutz** der Produktmodelldaten

- Deshalb sind zahlreiche Ingenieuranwendungen wie zum Beispiel EDM*- oder CAD-Systeme oft unter Nutzung von Datenbanksystemen eingesetzt. (*Engineering Data Management)

- Auch im kaufmännischen Arbeitsumfeld finden sich zahlreiche DB-basierte Systeme wie SAP ERP oder Workflow Management – Systeme.

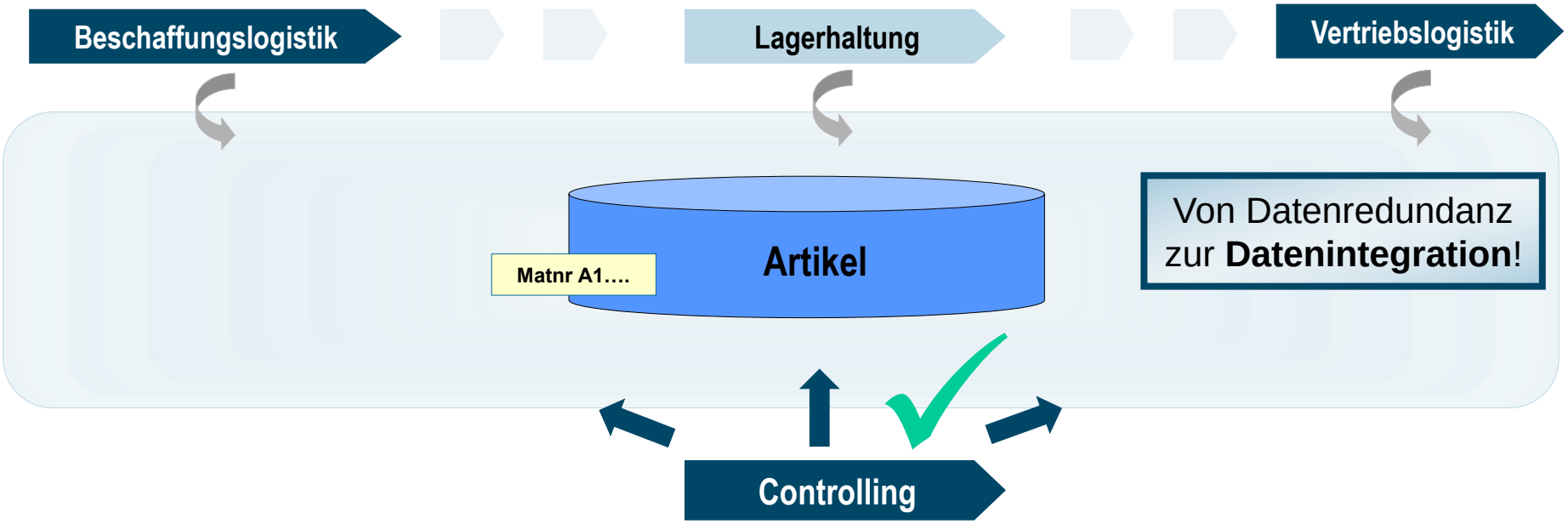

Beschaffungslogistik
Grunddatenverwaltung
Fremdteile (Artikel)
Lieferanten
Lieferanten-konditionen
Anforderungsbearbeitung
Angebotseinholung
Bestellung
Lieferantenauswahl
Bestellungsübermittlung
Bestellüberwachung
Anfragen
Mahnen
Wareneingang
Qualitätsprüfung
Verteilung
Transportplanung
Buchung
Rechnungsprüfung
Erfassung
Prüfung
Buchung
Bezahlung
Informationsobjekte
Artikel
Externe Geschäftspartner
Konditionen
Angebot
Auftrag
Anfrage
Mahnung
Qualitätstests
Lieferschein
Lagerungsvorgang
Transport
Buchung
Rechnung
Lieferschein
Rechnung
Buchung
Zahlung
Vertriebslogistik
Fertigteile (Artikel)
Kunden
Kunden-konditionen
Grunddatenverwaltung
Anfragenbearbeitung
Angebotserstellung
Auftragserfassung
Auftragsbearbeitung
Auftragsannahme
Anfragebearbeitung
Mahnbearbeitung
Auftragsverfolgung
Qualitätsprüfung
Kommissionierung
Tourenplanung
Buchung
Versand
Erfassung
Schreiben
Buchung
Fakturierung
Zahlungseingang



Probleme

- Redundanzen
- Inkonsistenzen
- Integritätsverletzungen
- Kollisionen beim Mehrbenutzerbetrieb
- Datenverlust
- Sicherheitsprobleme / Zugriffsverletzungen
- Hohe Entwicklungskosten

Isolierte Entwicklung von Anwendungen mit konventioneller Dateiverwaltung



Lösung: Zentrale Datenbank

Eine Datenbank ist das **vitale Zentrum** eines jeden Unternehmens!

Wertschöpfungskette…
Prozesse…

Warenwirtschaftssystem-anbieter

SAP · Microsoft Dynamics NAV · PeopleSoft · sage software · …

Datenbank-anbieter

ORACLE · IBM DB2 · Microsoft SQL Server 2008 · …

Gliederung

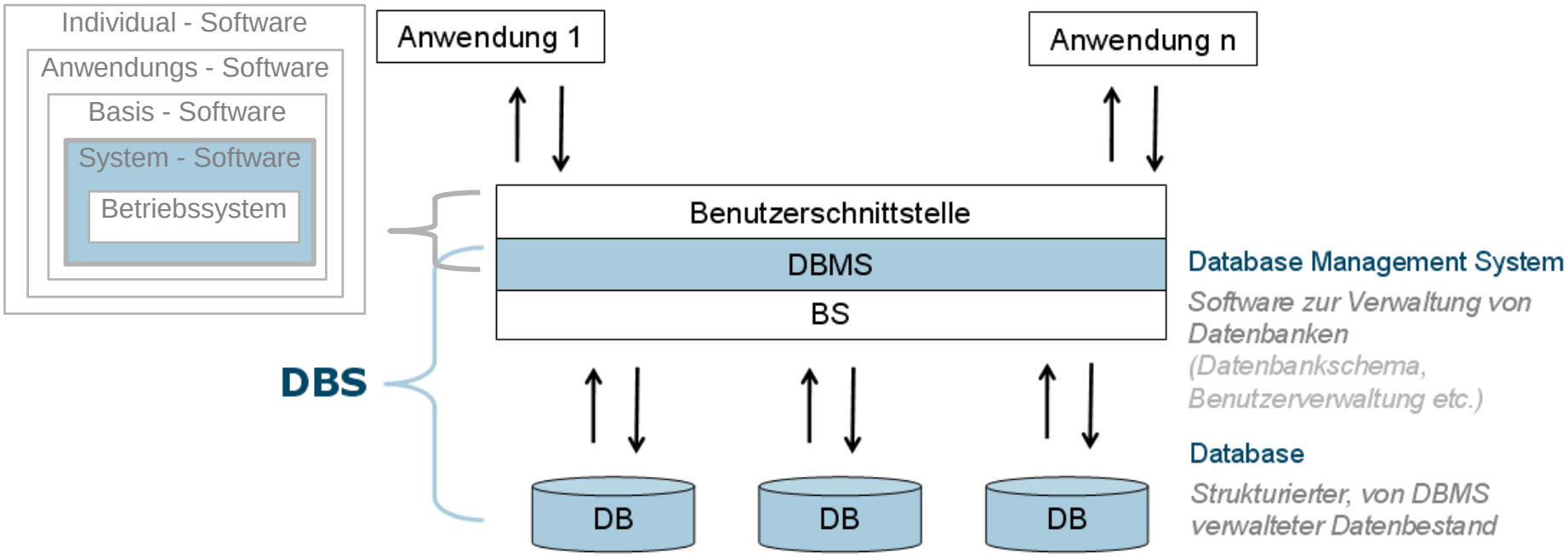
1. Einführung

- Gründe für die Einführung eines Datenbanksystems
- **Definition Datenbanksysteme**
- Architekturen von Datenbanksystemen
- Entwicklungsgeschichte der Datenbanksysteme
- Arten von Datenbanksystemen
- Verbreitete Datenbanksysteme

Idee: Datenintegration durch Datenbanksysteme

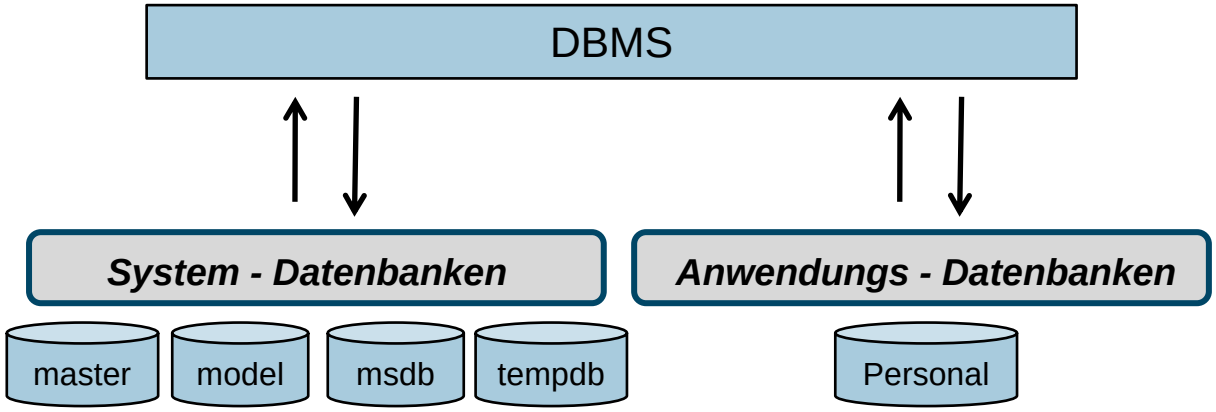
Definition Datenbanksystem:
Computergestütztes System, bestehend aus Datenbasis zur Beschreibung eines Ausschnitts der realen Welt und Programmen zum geregelten Zugriff auf die Datenbasis.

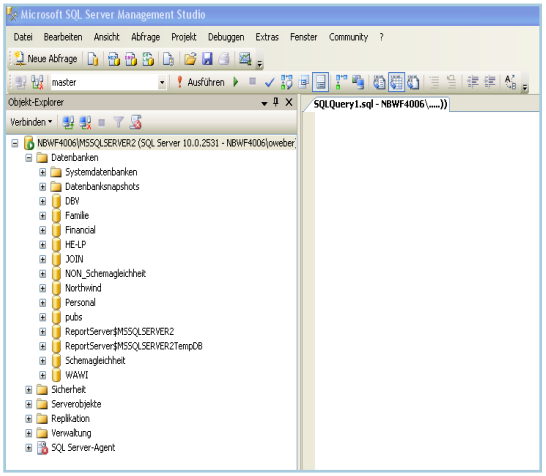
Wichtig: **Trennung von Daten und Programmen!**



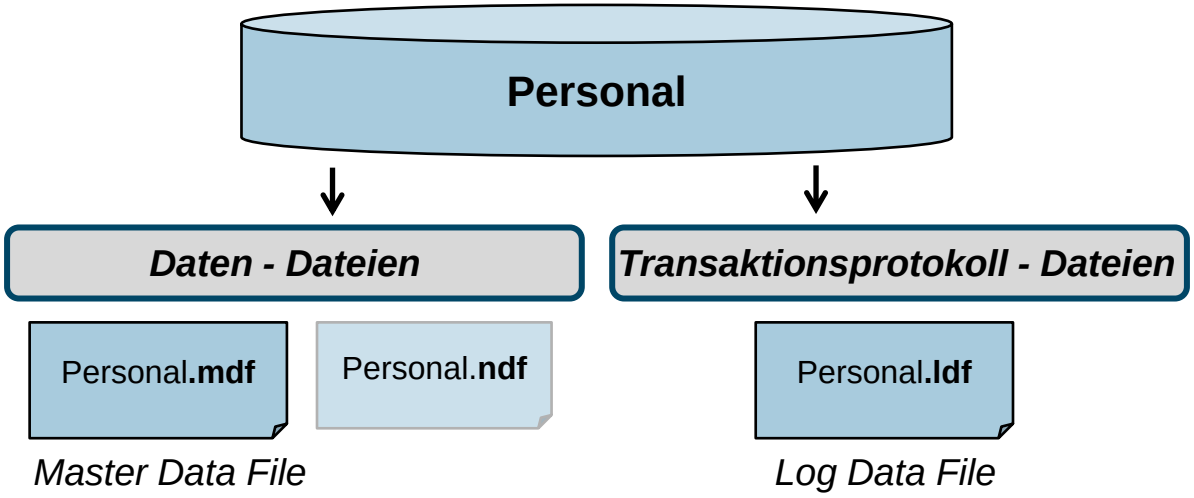
Datenbanksystem = Datenbank Management System + n * Datenbank

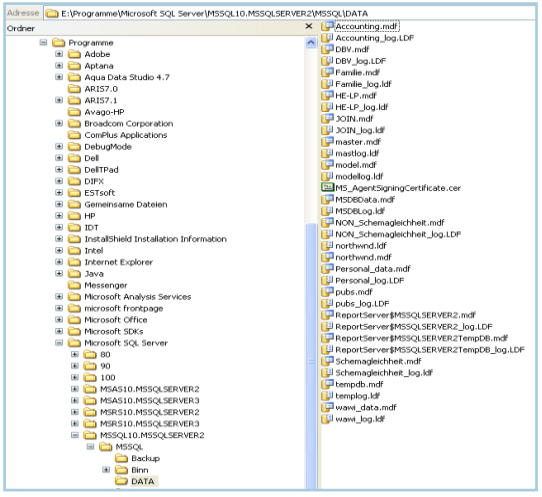
Bestandteile eines Datenbanksystems – DBMS und DB's





Bestandteile einer Datenbank - Datenbankdateien





Neun Anforderungen an ein DBMS

DBMS als „Black Box“

(Daten-) Integration	Einheitliche Verwaltung aller von Anwendungen benötigten Daten, kontrollierte, nicht redundante Datenhaltung.
Operationen	Zur Datenspeicherung, Datenänderung, Datenauswertung.
Katalog	Auch **data dictionary** genannt, enthält Verwaltungsinformationen in Form von Datenbeschreibungen (z.B. Tabellen, Sichten).
Benutzersichten	An die jeweiligen Bedürfnisse der Anwender (Anwendung) angepasste Sicht auf Daten der Datenbank.
Konsistenz-überwachung	Auch **Integritätssicherung** genannt, zur Gewährleistung der Korrektheit von Dateninhalten und der Überwachung von Änderungen, damit die Konsistenz nicht gefährdet wird.
Datenschutz	Auch **Zugriffskontrolle** genannt, zum Ausschluss unberechtigter Zugriffe.
Transaktionen	Zusammenfassung von zusammenhängenden / zusammengehörenden Operationen auf der Datenbank. Diese müssen entweder vollständig oder gar nicht ausgeführt werden.
Synchronisation	Zugriffe von verschiedenen Benutzern auf gleiche Daten müssen so koordiniert werden, dass Fehler vermieden werden können.
Datensicherung	Ermöglicht eine schnelle und weitgehend fehlerfreie Wiederherstellung der Datenbank nach Systemfehlern.

Gliederung

1. Einführung

- Gründe für die Einführung eines Datenbanksystems
- Definition Datenbanksysteme
- **Architekturen von Datenbanksystemen**
- Entwicklungsgeschichte der Datenbanksysteme
- Arten von Datenbanksystemen
- Verbreitete Datenbanksysteme

Architekturen von Datenbanksystemen

Datenbankarchitekturen kann man aus verschiedenen Blickwinkeln betrachten:

Schemaarchitektur

Die Schemaarchitektur beschreibt den Zusammenhang zwischen dem konzeptionellen, internen und externen Schema. Außerdem ordnet sie die Datenbank-Anwendungsprogramme in diese Schemata ein.

Systemarchitektur

Die Systemarchitektur beschreibt den Aufbau eines Datenbanksystems aus Komponenten, Bausteinen oder Werkzeugen. In Standardisierungsvorschlägen werden die Schnittstellen zwischen diesen Komponenten genormt, nicht jedoch die Komponenten selbst.

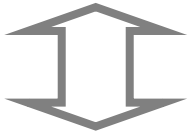
Anwendungsarchitektur

Die Anwendungsarchitektur beschreibt die Einbindung des Datenbanksystems in eine konkrete Applikation, etwa ein Web-Shop, eine ERP-Anwendung oder ein entscheidungsunterstützendes Data-Warehouse-System. Insbesondere wird hierbei die Aufteilung der Funktionalität des Gesamtsystems auf die einzelnen Komponenten und deren Verbindung festgelegt.

Daten(bank)modell vs. Daten(bank)schema

Daten(bank)-Modell

Ein **Daten(bank)modell** ist eine **formale Modellierungssprache** mit eingeschränkten Ausdrucksmitteln und genau definierter Syntax und Semantik zur Beschreibung der Struktur der Daten / Informationseinheiten.



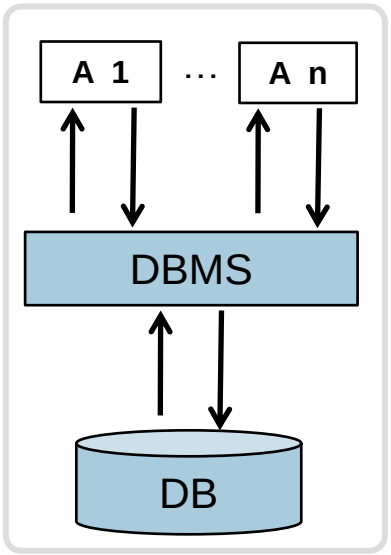
Daten(bank)-Schema

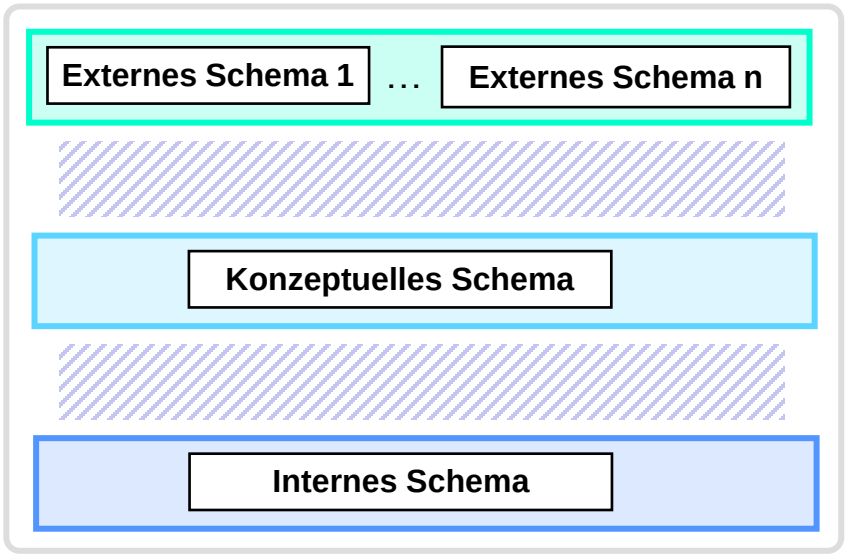
Ein **Date(nbank)schema** ist eine **formale Festlegung der Struktur** der Daten für eine konkrete Datenbank / Anwendung, das zur Datendarstellung ein Daten(bank)modell verwendet.

- In der Praxis oft: Daten(bank)modell = Daten(bank)schema -

Drei-Ebenen-Schemaarchitektur

Die Schemaarchitektur teilt ein Datenbankschema in drei aufeinander aufbauenden Ebenen auf:





Externe Ebene
Präsentation

Konzeptuelle Ebene
Information

Interne Ebene
Datenspeicherung

Die Umsetzung der einzelnen Ebenen erfolgt durch entsprechende Schemata. Damit wird für jede Ebene eines DB-Systems ein entsprechendes Schema (Modell) gebildet, d. h. jeweils ein Abbild der Informationswelt. Folgende Abstraktionsstufen von Modellen werden unterschieden:

Abstraktionsstufe	Modell	Algorithmen
Entwurfsmodelle (semantisches Modelle) - abstrakt-	**Entity-Relationship-Modell (ER-Modell)**	**Struktogramme**
Realisierungsmodelle (systemabhängige Modelle) - konkret -	**Relationenmodell Netzwerkmodell Hierarchisches Modell**	**Struktogramme → SQL Java, C#, C++, C, Pascal**

Drei-Ebenen-Schemaarchitektur

Datenbankschema besteht demnach aus

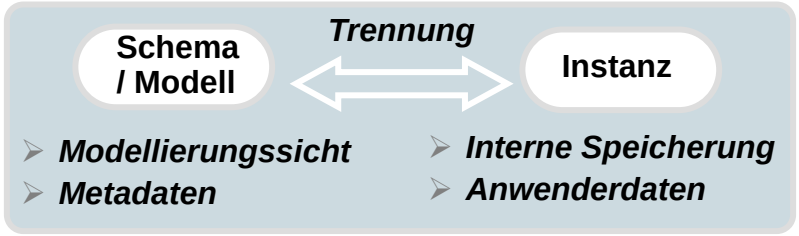
	Modellierungsinhalte	Modellierungsmethoden	Systemabhängigkeit
Externes Schema	▪ **Ergebnis der Sichtdefinitionen auf den Datenbestand**	▪ **ER-Modell oder Relationales Modell**	▪ **keine**

Konzeptuelles Schema	▪ **Gesamtsicht auf den Datenbestand bzw. vollständige Beschreibung der Struktur einer Datenbank**
	▪ **Ergebnis**
	  ▪ **der konzeptionellen Datenmodellierung**
	  ▪ **des logischen Datenbankdesigns**
	  ▪ **der Datendefinition**
		▪ **ER-Modell**
		▪ **Relationales Modell**
		▪ **SQL-Implementierung**
			▪ **keine**

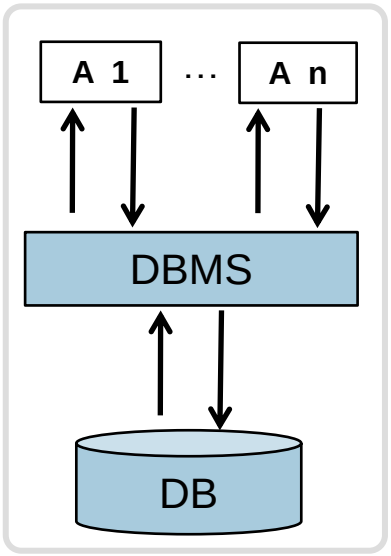
Internes Schema	▪ **Physische Abbildung der Daten des konzeptuellen Modells auf Speichermedien** ▪ **Festlegung der Dateiorganisation und der Zugriffspfade für das konzeptuelle Schema fest**		▪ **BS systemabhängig**
Anwendungsprogramme	▪ **Ergebnis der Datenbankanwendungsprogrammierung** ▪ **Arbeiten idealerweise auf den externen Schemata**		

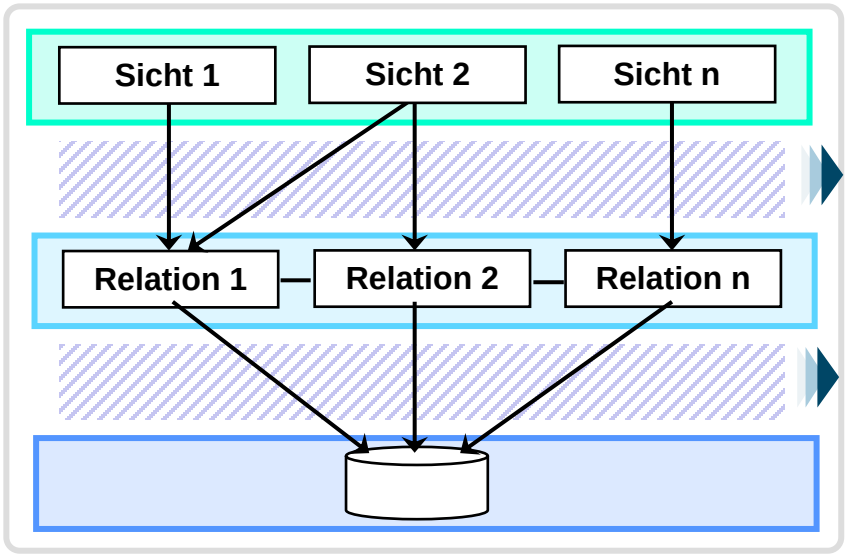
Drei-Ebenen-Schemaarchitektur

- Die Objekte der Modelle einschließlich ihrer Beziehungen werden im Data Dictionary hinterlegt



- Durch das Drei-Ebenen-Konzept Gewährleistung der notwendigen Unabhängigkeit der Daten





Externe Ebene
Präsentation

Logische Datenunabhängigkeit
Anwendungsunabhängigkeit

Konzeptuelle Ebene
Information

Physische Datenunabhängigkeit
Implementierungsunabhängigkeit

Interne Ebene
Datenspeicherung

Ziel dieser Architektur: Datenunabhängigkeit & Co.

Jede dieser Ebenen ist **unabhängig** voneinander zu gestalten.
Das bedeutet, dass jede der drei Ebenen **ausgetauscht** werden kann, ohne die anderen zu beeinflussen!

Stabilität der Benutzerschnittstelle gegen Änderungen:

Logische Datenunabhängigkeit

Anwendungsunabhängigkeit

Die Datenbank wird von den Änderungen und Erweiterungen der Anwendungsschnittstellen abgekoppelt

Änderungen am konzeptuellen und gewissen externen Schemata haben keine Auswirkungen auf andere externe Schemata und Anwendungsprogramme

- ✓ **Systemunabhängigkeit / Portierbarkeit**

Physische Datenunabhängigkeit

Implementierungsunabhängigkeit

Änderungen der Dateiorganisation und Zugriffspfade haben keinen Einfluss auf das konzeptuelle Schema

- ✓ **Tuning – Vereinfachung**
- ✓ **Hohe Performance**
- ✓ **Gute Speicherplatznutzung**
- ✓ **Entwicklung standardisierter Schnittstellen**

Ohne dieser Architektur:

Anwendungsprogrammierer / Benutzer können Anwendungen nicht programmieren / benutzen, ohne
→ interne Darstellung der Daten
→ Speichermedien oder Rechner
zu kennen (Datenunabhängigkeit nicht gewährleistet!)

Deklarative Sprache vs. Prozedurale Sprache

Datenbanken haben sich aber auch aus einem anderen Grund durchgesetzt. Während die Programmierung von Dateizugriffen auf praktisch jedem Betriebssystem anders aussieht, wurde für die Manipulation von relationalen Datenbanken eine Sprache entwickelt, die auf allen Plattformen (fast) gleich verwendet werden kann: SQL (Structured Query Language). Sie ermöglicht sowohl die Manipulation der Datenbankstruktur als auch der Daten selbst. Unterschiede zwischen den SQL-Dialekten sind lediglich datenbankspezifisch.

natürliche Sprache

„Selektiere die Namen der Studenten, die in Esslingen wohnen“

deklarative bzw. deskriptive Sprache

```
SELECT   Name
FROM     Studenten
WHERE    Ort = 'Esslingen'
```

Keine Kenntnisse über physische Datenstruktur erforderlich !

prozedurale Sprache

```
get first STUDENTEN
  search argument (Ort = 'Esslingen')
while status = 0 do
begin
  print (Name)
  get next STUDENTEN
    search argument (Ort = 'Esslingen')
end
```

Kenntnisse über physische Datenstruktur erforderlich !

Der Benutzer hat eine „deskriptive“ Sprache zur Verfügung, mit welcher er beschreibend auf die gespeicherten Daten zugreifen kann; demgegenüber benötigt das System eine **Prozedur** (Transformationsschritte), anhand derer der Benutzerauftrag bearbeitet werden kann.

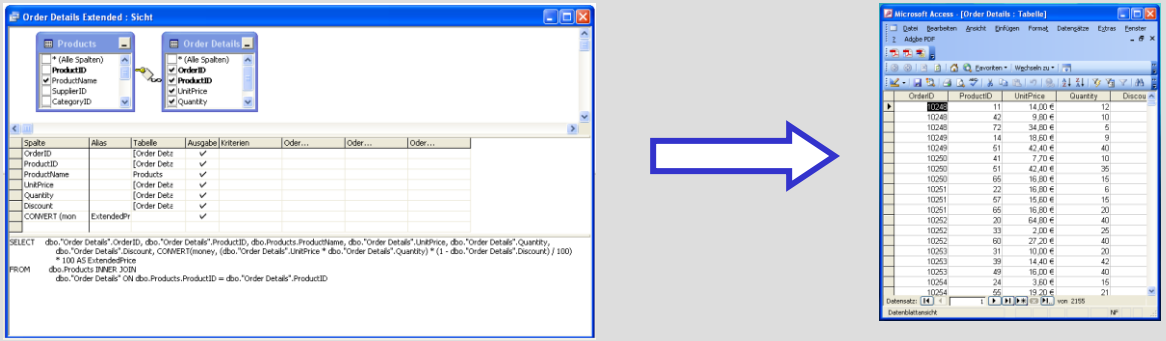
Zusammenarbeit der Komponenten eines DBMS im Detail

Befehlsentgegennahme	Input/Output-Prozessor (z.B. Web Browser) nimmt Kommandos entgegen
Syntaxprüfung	Parser führt syntaktische Analyse durch, evtl. Aufruf des Precompilers
Autorisierungskontrolle	Feststellung, ob der Benutzer mit den angeforderten Daten überhaupt arbeiten darf
Integritätsprüfung	Prüfung auf semantische Korrektheit (Konsistenz) der DB, Aufruf des Update-Prozessors im Falle einer Veränderung an der Struktur
Übersetzung der Anfrage in eine Zwischenform	Query-Prozessor übersetzt die Anfrage des externen Schemas in das konzeptionelle Schema
Anfrageoptimierung	Optimierer verändert ungeschickte oder unnötig kompliziert definierte Anfragen in bessere bzw. effizienter ausführbare Anfragen (bzw. Anfragen-Zwischenform)
Zugriffsplanerstellung	Festlegung der auf die benötigte Daten verfügbaren Zugriffsstrukturen (z.B. Indexe), Auswahl eines möglichst effizienten Zugriffspfades und Erstellung eines Zugriffs- bzw. Ausführungsprogramms, d. h. einer Code-Generierung für den Benutzerauftrag
Transaktions-verwaltung	Transaktions-Manager lässt nach außen die DB für jeden Benutzer als ein exklusiv verfügbares Betriebsmittel erscheinen, intern arbeitet er nach dem "Alles oder Nichts" - Prinzip, d. h. jede Transaktion wird immer vollständig oder gar nicht ausgeführt (Protokollierung im Log der DB)
Transaktions-wiederherstellung	Recovery-Manager im Falle des Systemabsturzes versetzt die DB in den Zustand, in dem sie sich vor dem Start der Transaktion befand (Infos aus dem Log der DB)
Speicherverwaltung im Hauptspeicher	Puffer-Manager verwaltet des im Hauptspeichers des betreffenden Rechners für jedes laufende Datenbank-Programm (jede Transaktion) bereitgestellten Puffer, Geräte- und Sekundärspeicher - Manager verwaltet der dem DBMS zur Verfügung stehenden Hardware-Betriebsmittel

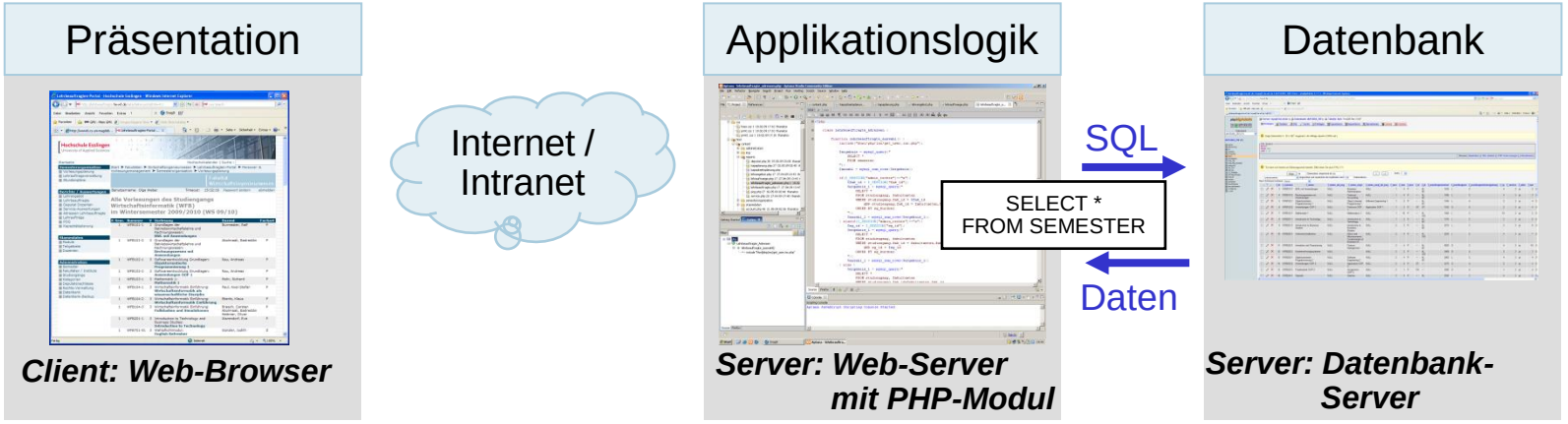
Beispiele zur Implementierung eines Datenbanksystems

Interaktives SQL (z. B.: mit MS Access)

Präsentation + Applikationslogik + Datenbank

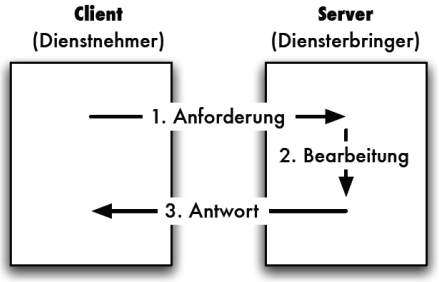


Embedded SQL (z. B.: in den PHP-Skripten einer Web-Anwendung (E-Shops etc.))



Anwendungsarchitekturen

Die Funktionalitäten einer Datenbank-applikation (Software) lassen sich auf Basis des Client-Server-Modells (Server = Datenbanksystem) in drei logische Schichten (tiers) aufteilen:



Präsentationsschicht

Front-End
Präsentation und Benutzerinteraktion

Logikschicht

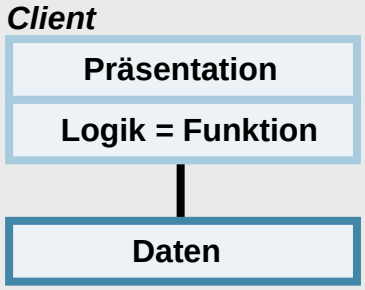
Business-Logic
Anwendungslogik

Datenschicht (DBS)

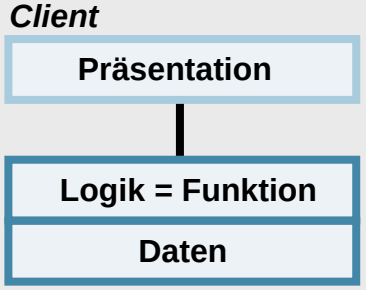
Back-End
Datenbank und Datenmanagement-funktionen (Speichern, Anfragen, …)

Zwei – Schichten – Architektur (2-Tier)

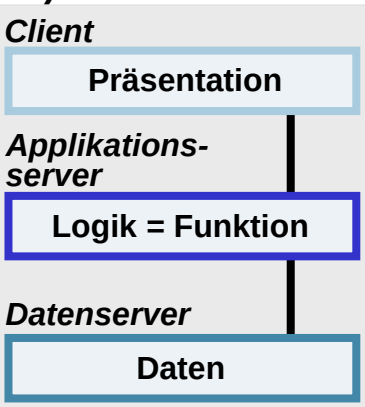
Fat - Client Architektur

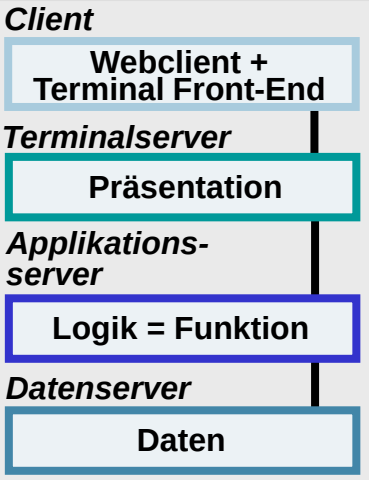


Thin - Client Architektur



Drei – Schichten – Architektur (3-Tier)





Entwicklungsgeschichte der Datenbanksysteme

1. Generation (50er Jahre): Dateisysteme auf Magnetband

2. Generation (60er Jahre): Dateisysteme auf Magnetplatte

3. Generation (70er Jahre): Prärelationale Systeme

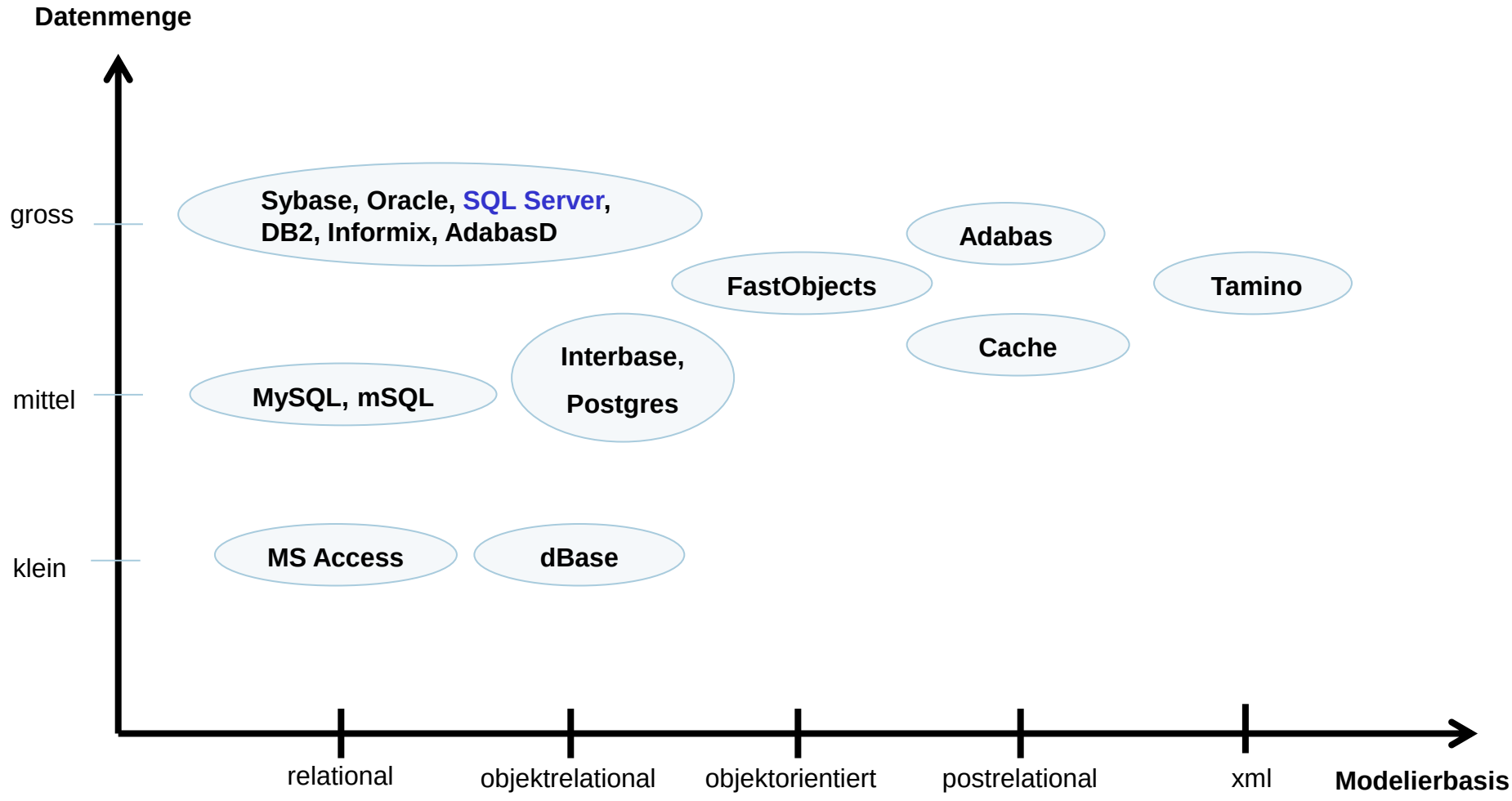
   z.B. Netzwerk-, Hierarchische Datenmodelle

4. Generation (80er Jahre): Relationale Datenbanksysteme

5. Generation (90er Jahre): Postrelationale Datenbanksysteme

   z.B. Objektrelationale und Objektorientierte Datenbanksysteme

Heute (mehr oder weniger) verbreitete Datenbanksysteme



Fragen?

Danke für Ihre Aufmerksamkeit !

F r a g e n