



Kapitel 1 – Einleitung

Vorlesung Datenbanken

Dr. Kai Höfig



- ◆ 1.1 Motivation
- ◆ 1.2 Relationen
- ◆ 1.3 Schema Architektur
- ◆ 1.4 Anwendungs- und System-Architekturen
- ◆ 1.5 Datenmodelle



Was ist also eine Datenbank?

- ♦ **Datenbank** = Menge von logisch gruppierten, strukturierten Informationseinheiten



Martin	Müller	Hauptstraße 1	Rosenheim	08031-99999999

Was sind die allgemeinen Vorteile einer Datenbank?

Was ist die einfachste Art Daten zu speichern?
Flat File



Vorteile der Datenbank gegenüber der „Speicherung auf Papier“

- ◆ Suchen/Filtern von Informationen
 - Überall oder nur in bestimmten Attributen
 - Beispiele
 - Alle Datensätze mit Vorname „Klaus“
 - Alle Personen, die im Oktober Geburtstag haben
- ◆ Sortieren nach verschiedenen Attributen
 - Beispiele
 - Sortiere nach Vornamen aufsteigen/ascending (A-Z)
 - Sortiere nach Namen absteigend/descending (Z-A)
- ◆ ...und viele weitere Operationen!
- ◆ Sicherheit vor Verlust der Daten
- ◆ Langfristige Aufbewahrung



Einige Begriffe am Beispiel einer einfachen Tabelle

Tabellenüberschriften einer Spalte (column) nennt man **Attribute**. Diese haben **Datentypen** z.B. Zahl, String oder Datum

Vorname	Name	Straße	PLZ	Ort	Geburtsdatum	Telefon
Hans	Huber	Hofstr. 4	83024	Rosenheim	10.10.1990	012-3456
Peter	Petersen	Parkweg 3	83024	Rosenheim	9.9.1965	9999-9999
Susi	Sorglos	Sandstr 2	80801	München	1.1.1991	0177-7777
Andrea	Ammer	Am Bach 1	88888	Ambach	21.12.1989	176462
Klaus	Klammer	Klarastr. 9	83646	Bad Tölz	4.4.1984	08041-4444
Mark	Markl	Marktstr. 2	83646	Bad Tölz	3.3.1983	08041-3333
Gabi	Genau	Giselastr. 5	81888	München	8.8.1988	089-8888

Den Inhalt einer Zelle (cell,field) nennt man **Attributwert**.

Eine Zeile (row) ist ein Datensatz (record) und wird **Tupel** genannt



Nachteile von Flat File Datenbanken

Inkonsistenz: für dasselbe Objekt in der Realität liegen unterschiedliche (redundante) Daten in der Datenbank vor.

Datenredundanz: Dieselben Daten werden mehrfach gespeichert.

Vorname	Name	Straße	PLZ	Ort	Geburtsdatum	Telefon
Hans	Huber	Hofstr. 4	83024	Rosenheim	10.10.1990	012-3456
Peter	Petersen	Hofstraße 4	83024	Rosenheim	9.9.1965	9999-9999
Susi	Sorglos	Sandstr 2	80801	München	1.1.1991	0177-7777
Andrea	Ammer	Am Bach 1	88888	Ambach	21.12.1989	176462
Klaus	Klammer	Klarastr. 9	83646	Bad Tölz	4.4.1984	08041-4444
Mark	Markl	Marktstr. 2	83646	Bad Tölz	3.3.1983	08041-3333
Gabi	Genau	Giselastr. 5	81888	München	8.8.1988	089-8888

Einfügeanomalie: Nur eine neue Straße mit PLZ alleine zu speichern macht hier keinen Sinn.

Änderungsanomalie: Beim Ändern der Hofstraße können leicht Inkonsistenzen entstehen, da diese redundant vorliegt.

Löschanomalie: Wird Andrea gelöscht, verliere ich auch die PLZ zur Straße.



Relationen beheben diese Nachteile

Vorname	Name	Geburtsdatum	Telefon		Straße	PLZ	Ort
Hans	Huber	10.10.1990	012-3456	↔	Hofstraße 4	↔ 83024	↔ Rosenheim
Peter	Petersen	9.9.1965	9999-9999	↔	Sandstr 2	↔ 80801	↔ München
Susi	Sorglos	1.1.1991	0177-7777	↔	Am Bach 1		
Andrea	Ammer	21.12.1989	176462				

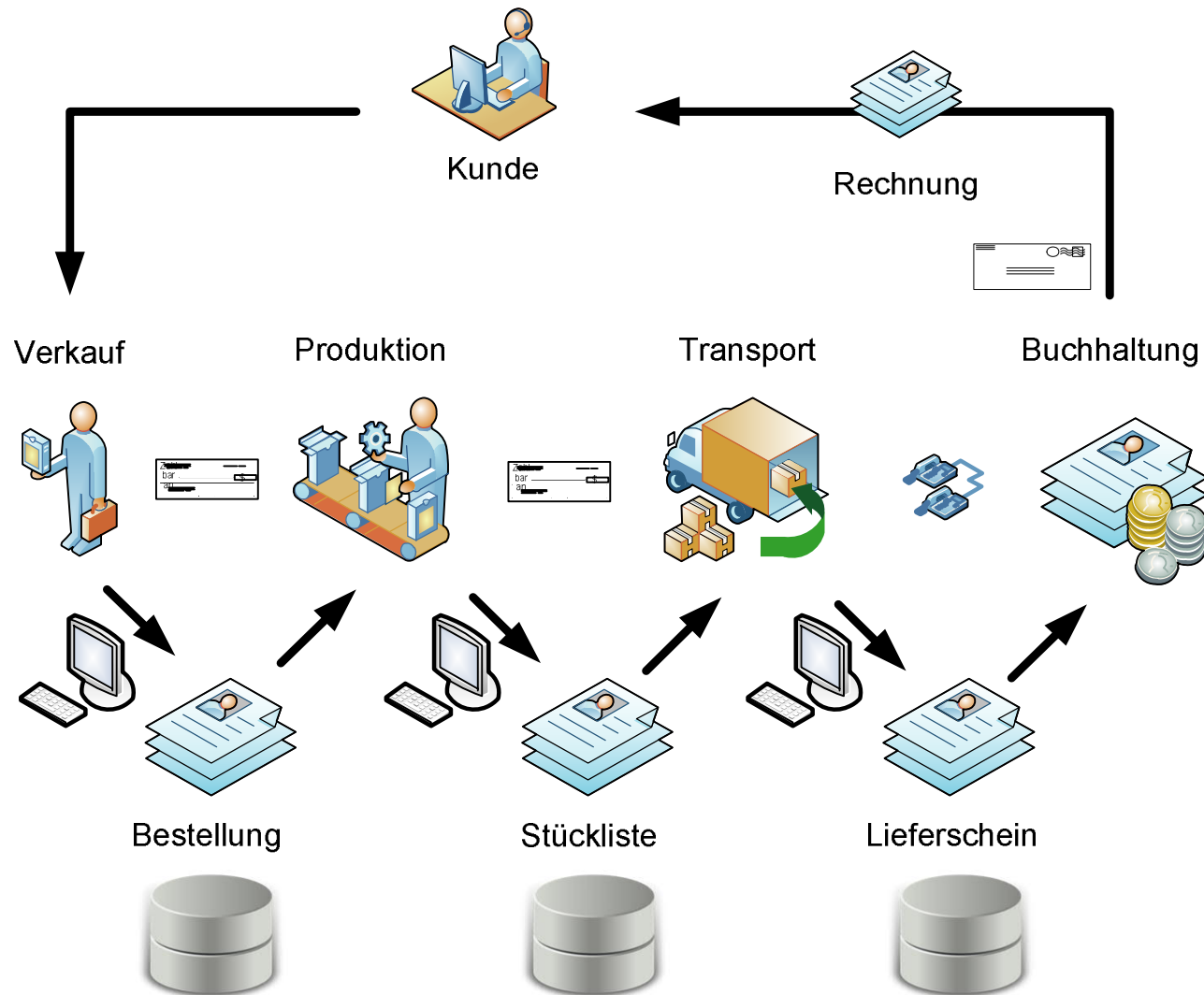
Die Bezeichnung von Hofstraße ist immer einheitlich, da sie nur einmal gespeichert ist.
Die Stadt Rosenheim muss nur einmal gespeichert werden
Es können Straßen hinzugefügt werden ohne das dort jemand wohnen muss
Personen können gelöscht werden, ohne dass Informationen über PLZ verloren gehen

Aber:

Mehrere Telefonnummern zu einer Person lassen sich trotzdem nicht speichern. Der Erfolg von Systemen ist daher an den geschickten Entwurf solcher Datenbank-Relationen und an die jeweilige Domäne gebunden. Dazu später mehr.



Eine heile Flat File Welt?



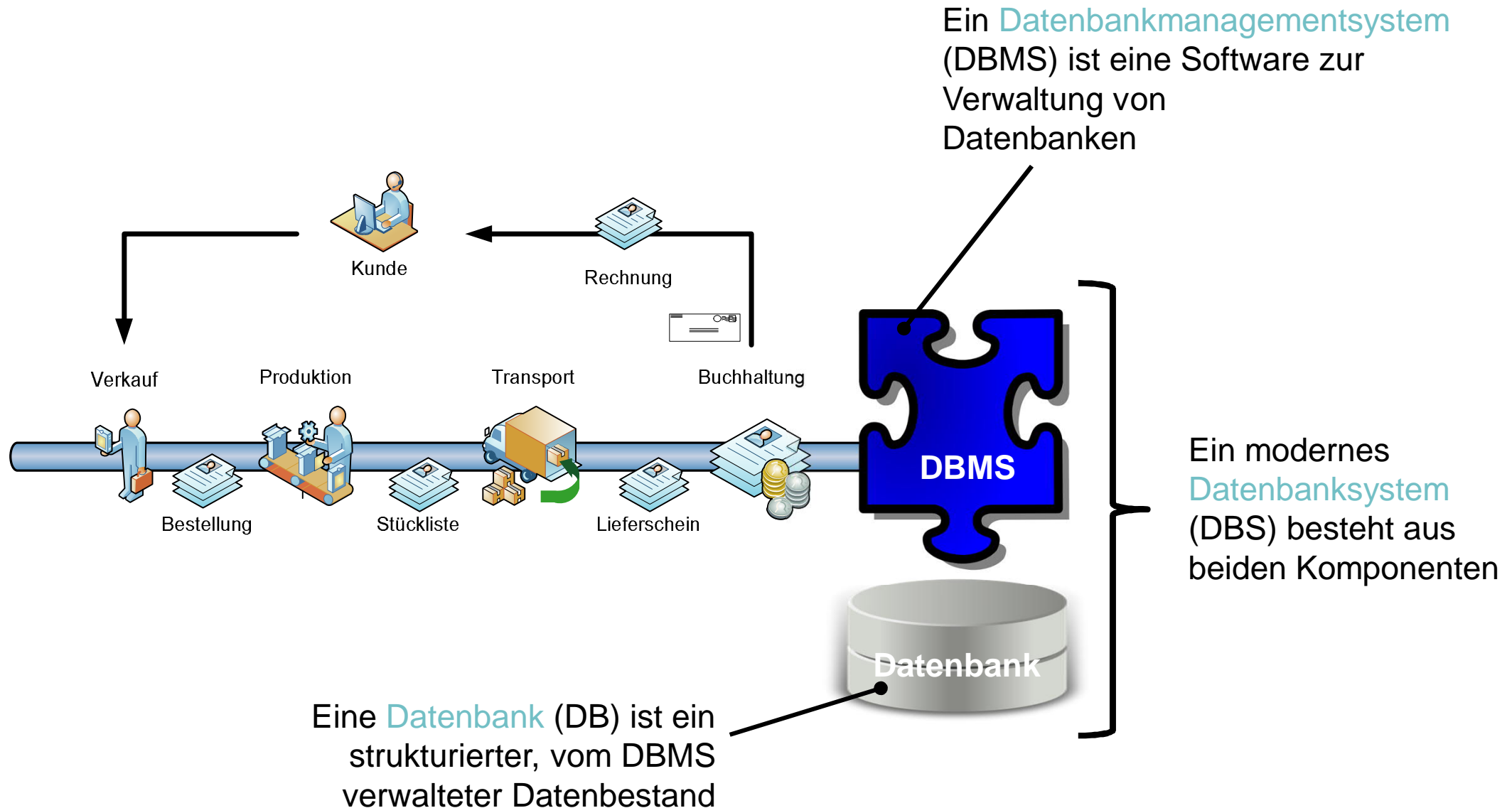
Selbst eine mit hohem Aufwand programmierte Flat-File Datenhaltung die Redundanzen wirkungsvoll verhindert, löst nicht das Problem der Verteilung von Prozessen auf unterschiedliche Systeme.

Oder anders ausgedrückt:

Das kann man schon mit Excel machen, aber dann ist es halt Sch****.



Datenintegration durch Datenbanksysteme





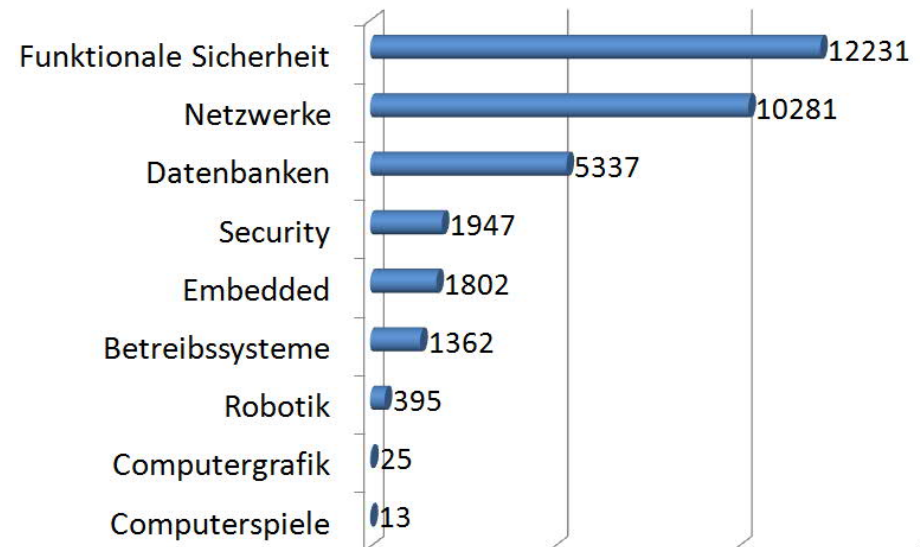
Zusammenfassung: Herausforderungen (relationaler) Flat File Datenbanken

- ◆ Inkonsistenzen durch redundante Speicherung der gleichen Identität mit unterschiedlichen Attributwerten.
- ◆ Vergessen von Änderungen und dadurch hervorgerufene Inkonsistenzen (Änderungsanomalie).
- ◆ Keine zentrale, „genormte“ Datenhaltung: Jede einzelne Anwendung muss die interne Darstellung der Daten und den Speicherort kennen
- ◆ Jede einzelne Anwendung muss für die effiziente Verarbeitung der Daten optimiert werden
- ◆ Daten häufig wertvoller und langlebiger als Anwendungen, leben aber „in“ der Anwendung
- ◆ Mehrere Benutzer oder Anwendungen können nicht parallel auf den gleichen Daten arbeiten, ohne sich zu stören
- ◆ Datenschutz und Datensicherheit sind nicht gewährleistet
- ◆ Verschwendung von Speicherplatz
- ◆ Einfügeanomalie
- ◆ Änderungsanomalie
- ◆ Löschanomalie



Diese Idee kommt an!

- ◆ Datenbanksysteme sind Herzstück heutiger IT-Infrastrukturen
- ◆ . . . allgegenwärtig
- ◆ Datenbankspezialisten sind gefragt



Quelle: Stepstone.de September 2017




Ziele und Herausforderungen von DBS

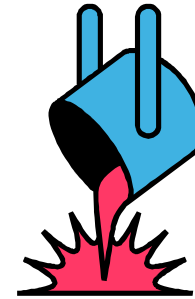
- ◆ Effiziente Verwaltung sehr **großer Datenmengen**
 - ➔ Wie kann man riesige Datenmengen (Terabytes) effizient verarbeiten?
- ◆ **Paralleler Zugriff mehrere Benutzer** auf die Daten
 - ➔ Wie können **viele Nutzer** (>10.000) gleichzeitig mit den Daten arbeiten?
- ◆ Gewährleistung von **Datenunabhängigkeit**
 - ➔ Wie organisiert (modelliert und nutzt) man Daten?
- ◆ Gewährleistung von **Datenschutz** und **Datensicherheit**
 - ➔ Wie werden Daten dauerhaft verlässlich gespeichert?
 - ➔ Wie kontrolliert man den Zugriff?



Schema Architektur Einleitung

- ♦ Auf den unterschiedlichen Ebenen eines Datenbank-basierten Systems liegen die Daten in unterschiedlichen Darstellungen und Ausschnitten vor.
- ♦ Eine solche zweckbezogene Art der Ausprägung der Datendarstellung nennt man ein **Schema**, damit sind nicht die Daten selber gemeint (**Instanz**).
- ♦ Beispiel: Der Hersteller Xoco stellt viele Arten von Schokoladentafeln aus einzelnen Zutaten (Bohnen, Butter, Zucker, etc.) her.
- ♦ Wie sehen die Daten in den einzelnen Schemata aus?

Interne Externe
Schemata

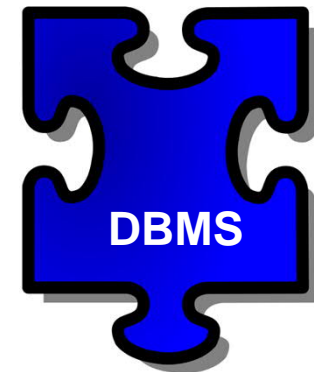


Produktions-System



Bestell-System

Konzeptionelle
Schemata



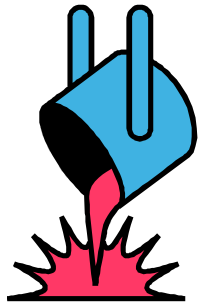
Interne
Schemata





Externe Schemata

- ◆ (Externes) **Schema** für das Produktions-System



Name	Zutaten	Temperatur	Rührdauer
Akapulko-Zartherb	30% Criollo Venezuela 20% Nacional Ecuador 48% Zucker 2% Vanille	76 Grad	90h
Criollo Venezuela 80%	80% Crillo Venezuela 20% Zucker	77 Grad	80h
Milch Zartschmelzend	35% Arriba Venezuela 65% Zucker	78 Grad	30h

- ◆ (Externes) **Schema** für das Bestell-System



Name	Kakao	Preis	Lieferzeit
Akapulko-Zartherb	50%	2,22 EUR	24h
Criollo Venezuela 80%	80%	5,30 EUR	48h
Milch Zartschmelzend	35%	1,50 EUR	12h



Konzeptionelles Schema – Darstellung in Tabellen (Relationen)

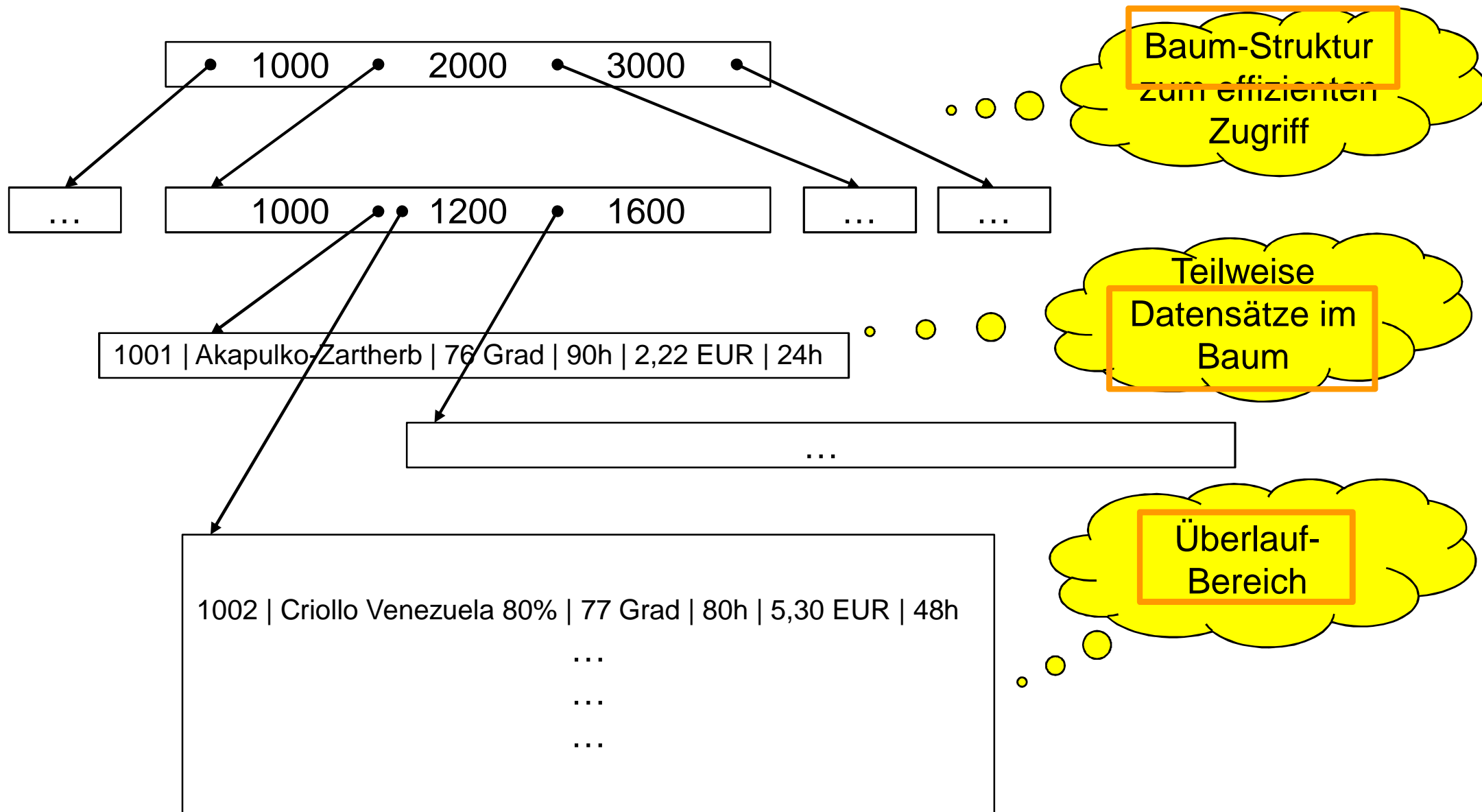
SNr	Name	Temperatur	Rührdauer	Preis	Lieferzeit
1001	Akapulko-Zartherb	76 Grad	90h	2,22 EUR	24h
1002	Criollo Venezuela 80%	77 Grad	80h	5,30 EUR	48h
1003	Milch Zartschmelzend	78 Grad	30h	1,50 EUR	12h

ZNr	Name	IstKakao
2001	Criollo Venezuela	True
2002	Nacional Ecuador	True
2003	Arriba Venezuela	True
2004	Vanille	False
2005	Zucker	False

SNr	ZNr	Anteil
1001	2001	30%
1001	2002	20%
1001	2005	48%
1001	2004	2%
1002	2001	80%
1002	2005	20%
1003	2003	35%
1003	2005	65%

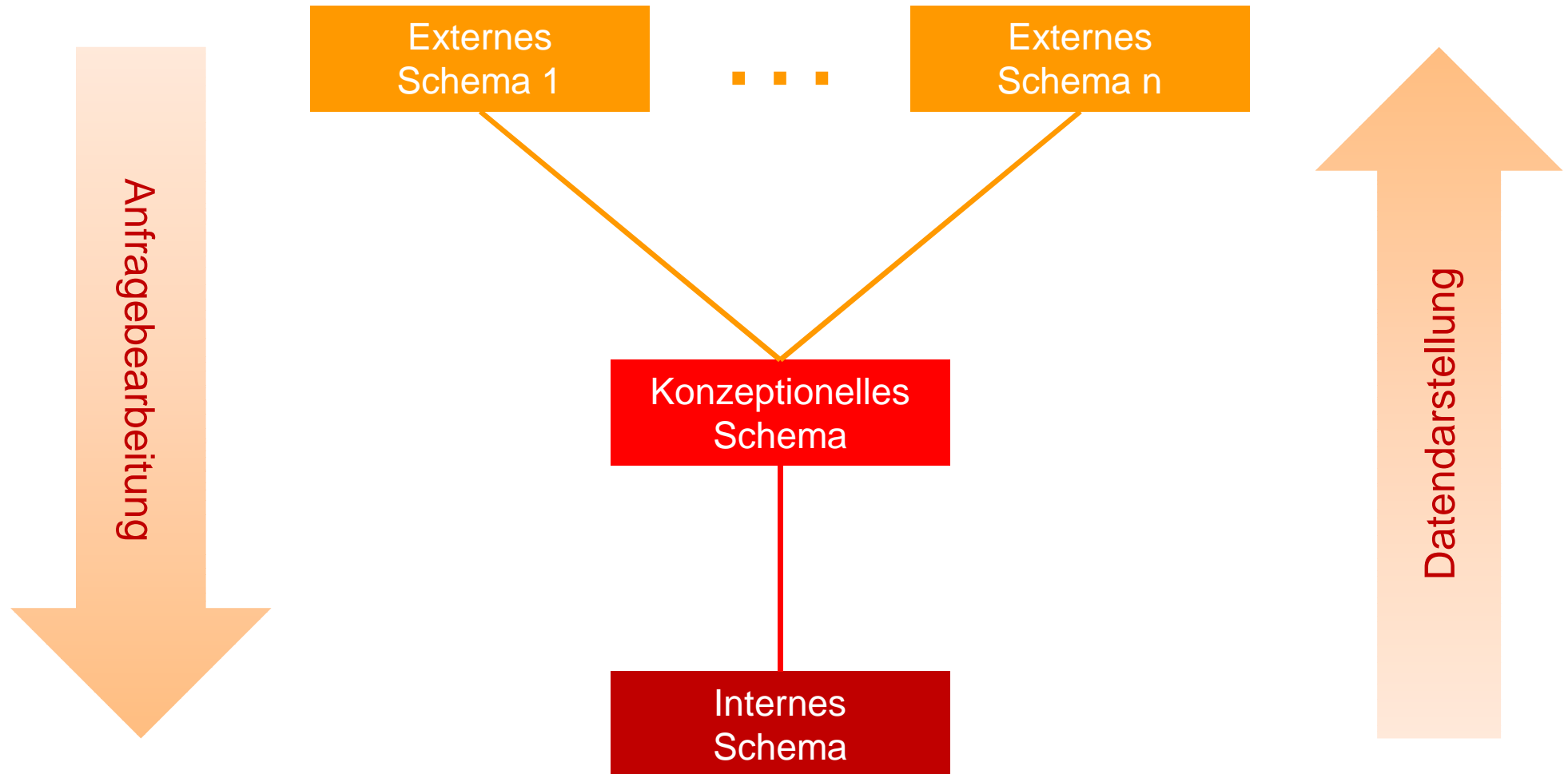


Internes Schema – interne Organisation der Daten





Schema-Architektur - allgemein





Schema-Architektur – allgemein – mit Beispiel

Anfragebearbeitung

Name	Zutaten	Temperatur	Rührdauer
Akapulko-Zartherb	30% Criollo Venezuela 20% Nacional Ecuador 48% Zucker 2% Vanille	76 Grad	90h
Criollo Venezuela 80%	80% Criollo Venezuela 20% Zucker	77 Grad	80h
Milch Zartschmelzend	35% Arriba Venezuela 65% Zucker	78 Grad	30h

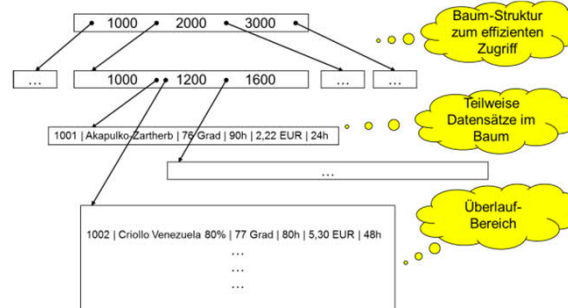
Name	Kakao	Preis	Lieferzeit
Akapulko-Zartherb	50%	2,22 EUR	24h
Criollo Venezuela 80%	80%	5,30 EUR	48h
Milch Zartschmelzend	35%	1,50 EUR	12h

SNr	Name	Temperatur	Rührdauer	Preis	Lieferzeit
1001	Akapulko-Zartherb	76 Grad	90h	2,22 EUR	24h
1002	Criollo Venezuela 80%	77 Grad	80h	5,30 EUR	48h
1003	Milch Zartschmelzend	78 Grad	30h	1,50 EUR	12h

ZNr	Name	IstKakao
2001	Criollo Venezuela	True
2002	Nacional Ecuador	True
2003	Arriba Venezuela	True
2004	Vanille	False
2005	Zucker	false

SNr	ZNr	Anteil
1001	2001	30%
1001	2002	20%
1001	2005	48%
1001	2004	2%
1002	2001	80%
1002	2005	20%
1003	2003	35%
1003	2005	65%

Datendarstellung





- ◆ Trennung Schema – Instanz
 - **Schema** (Metadaten, Datenbeschreibungen)
 - **Instanz** (Anwenderdaten, Datenbankzustand oder -ausprägung)
- ◆ **Schema-Architektur** beschreibt den Zusammenhang zwischen
 - konzeptionellen Schema (Ergebnis der Datendefinition)
 - internen Schema (Festlegung der Dateiorganisationen und Zugriffspfade)
 - externen Schema (Ergebnis der Sichtdefinition)
 - Anwendungsprogrammen (Ergebnis der Anwendungsprogrammierung)
- ◆ **Datenbankschema**: internes + konzeptuelles + externe Schemata
- ◆ DBMS haben i.allg. zwei Arten von Befehlen („Sprachen“)
 - **DDL (Data Definition Language)** → Änderungen am Schema (Struktur)
 - **DML (Data Manipulation Language)** → Änderungen an der Inhalt (Datensätze)

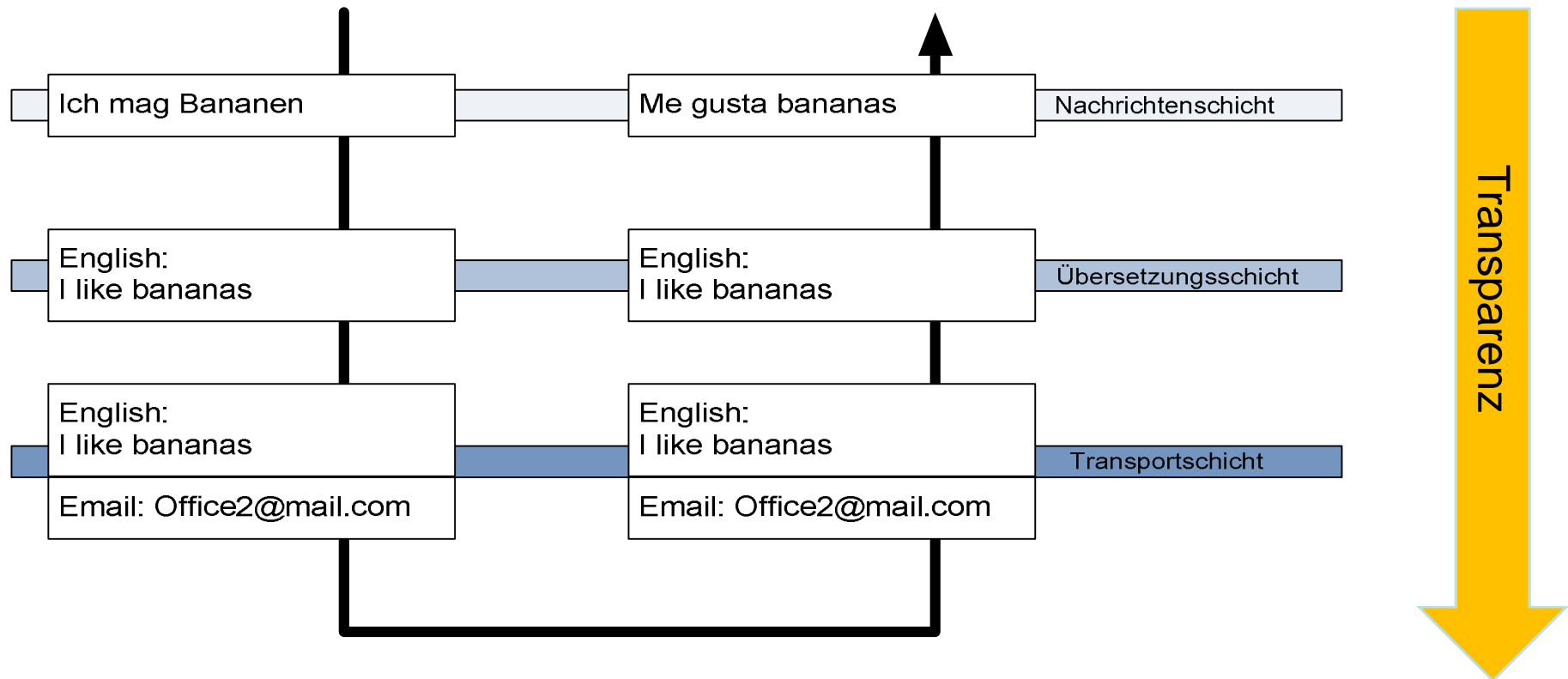


- ◆ Stabilität der Benutzerschnittstelle gegen Änderungen
 - **physisch**: Änderungen der Dateiorganisationen und Zugriffspfade haben keinen Einfluss auf das konzeptionelle Schema (Zugriffs-, Orts-, Skalierungs- und Migrationstransparenz)
 - **logisch**: Änderungen am konzeptionellen und gewissen externen Schemata haben keine Auswirkungen auf andere externe Schemata und andere Anwendungsprogramme

- ◆ Mögliche Auswirkungen von Änderungen am konzeptionellen Schema:
 - eventuell externe Schemata betroffen (Ändern von Attributen)
 - eventuell Anwendungsprogramme betroffen (Rekompilieren der Anwendungsprogramme, eventuell Änderungen nötig)
 - Nötige Änderungen werden jedoch vom DBMS erkannt und überwacht



Philosoph/Übersetzer Metapher

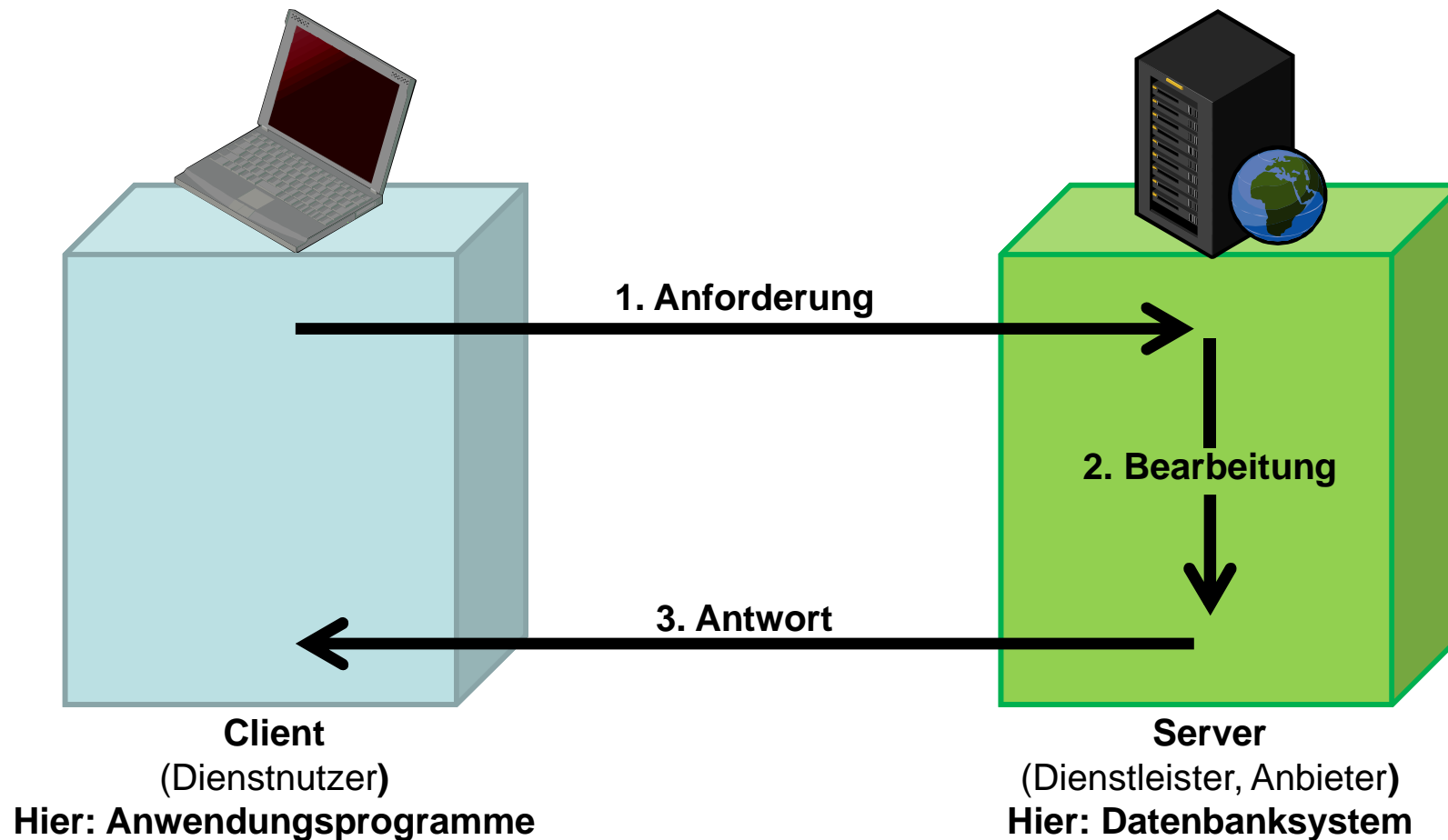


- Niedrige Schichten sind **transparent** für höhere Schichten.
- Für den Sender einer Nachricht, ist es unsichtbar, dass die Nachricht in Englisch übersetzt wird. Es könnte auch Chinesisch sein. Die verwendete Sprache in der Übersetzungsschicht ist für den Sender transparent.
- Außerdem ist die Transportschicht für den Sender transparent. Die Nachricht könnte auch per Fax übermittelt werden.



Anwendungs-Architekturen (1)

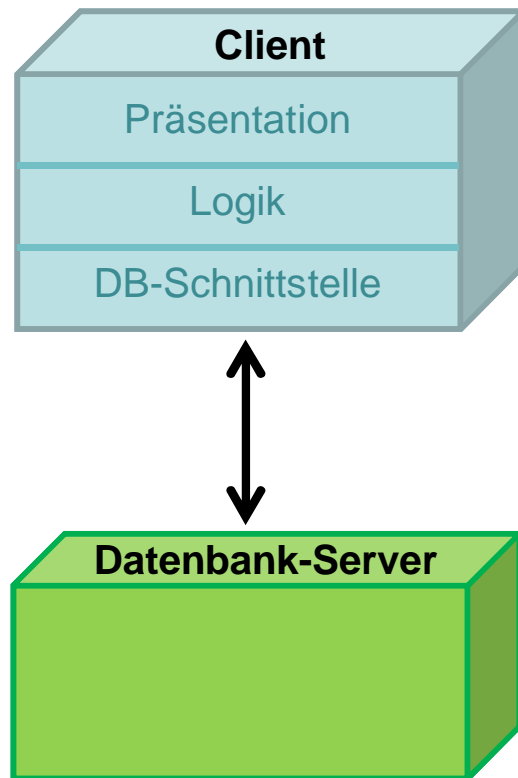
- Architektur von Datenbank Anwendungen typischerweise auf Basis des **Client-Server-Modells**: Server = Datenbanksystem



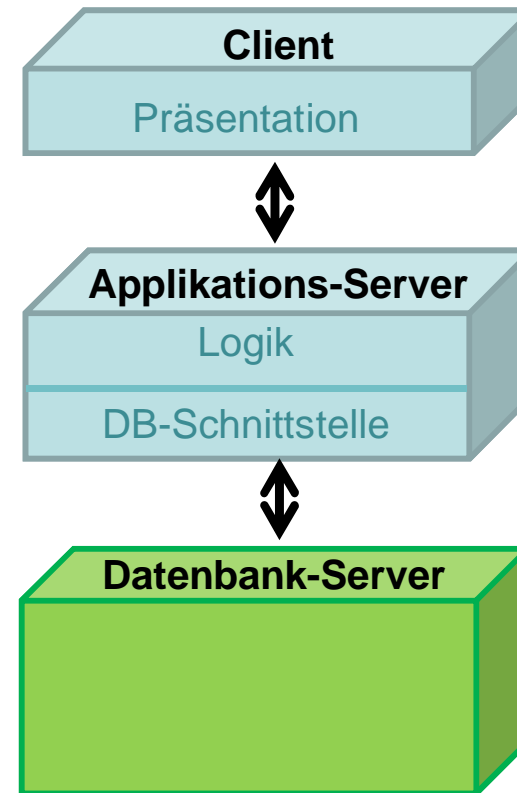


Anwendungs-Architekturen (2)

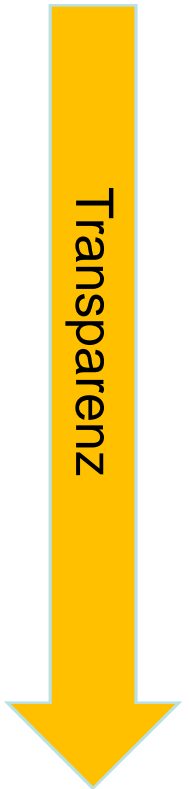
- ◆ Aufteilung der Funktionalitäten einer Anwendung
 - **Präsentation**: Präsentation und Benutzerinteraktion
 - **Logik**: Anwendungslogik („Business“-Logik)
 - **Datenbankschnittstelle**: Speichern, Anfragen, ...



2-Schichten-Architektur



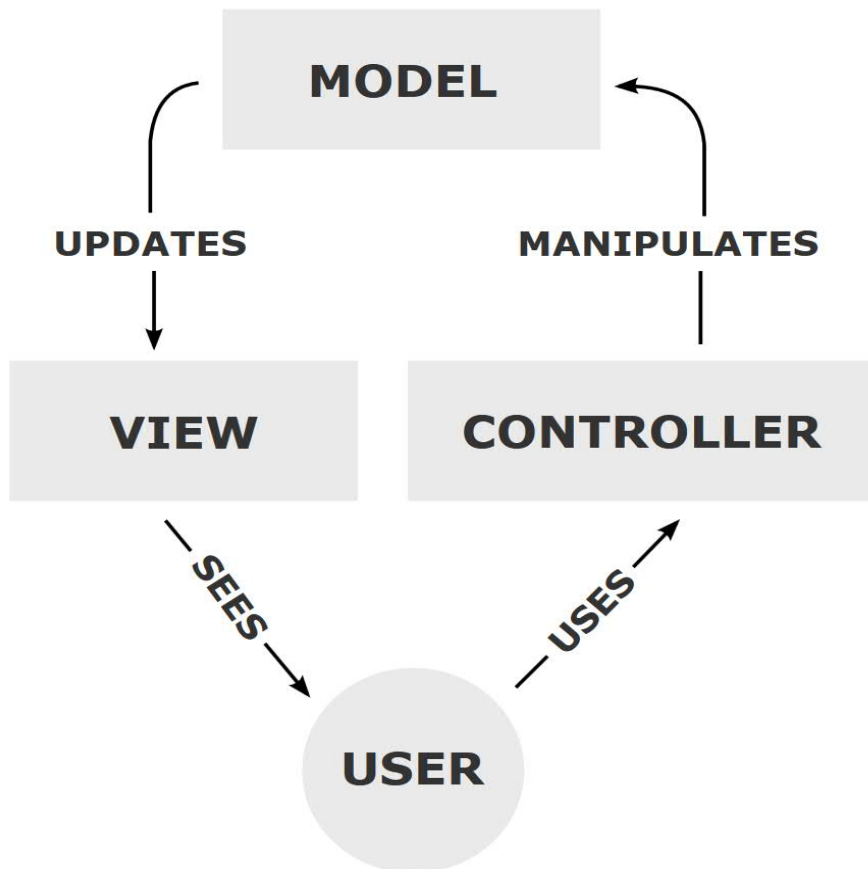
3-Schichten-Architektur





MVC – Ein Software Architekturmuster

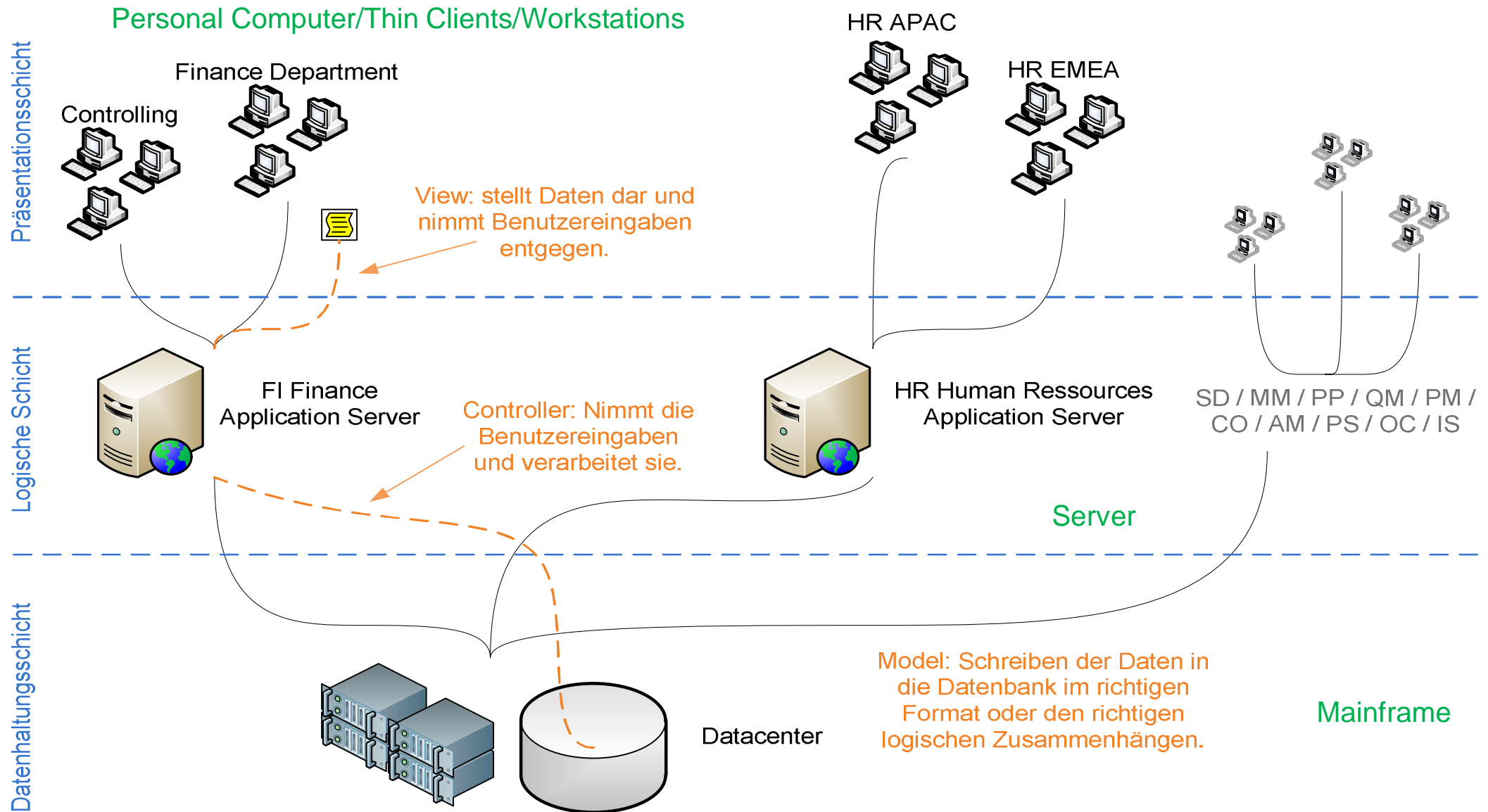
◆ Model View Controller



- Ähnlich zu der 3-Schichten Anwendungsarchitektur, macht es Sinn auch die Software einer Datenbank Anwendung logisch zu unterteilen.
- Der Benutzer sieht nur das, was ihm die Komponenten die zum View gehören zur Verfügung stellen (externes Schema), zum Beispiel dargestellt durch eine Webseite.
- Der Controller nimmt Benutzeranfragen entgegen und verarbeitet diese. Dazu verwendet er die externen und internen Schemata (Model).



Beispiel SAP R3

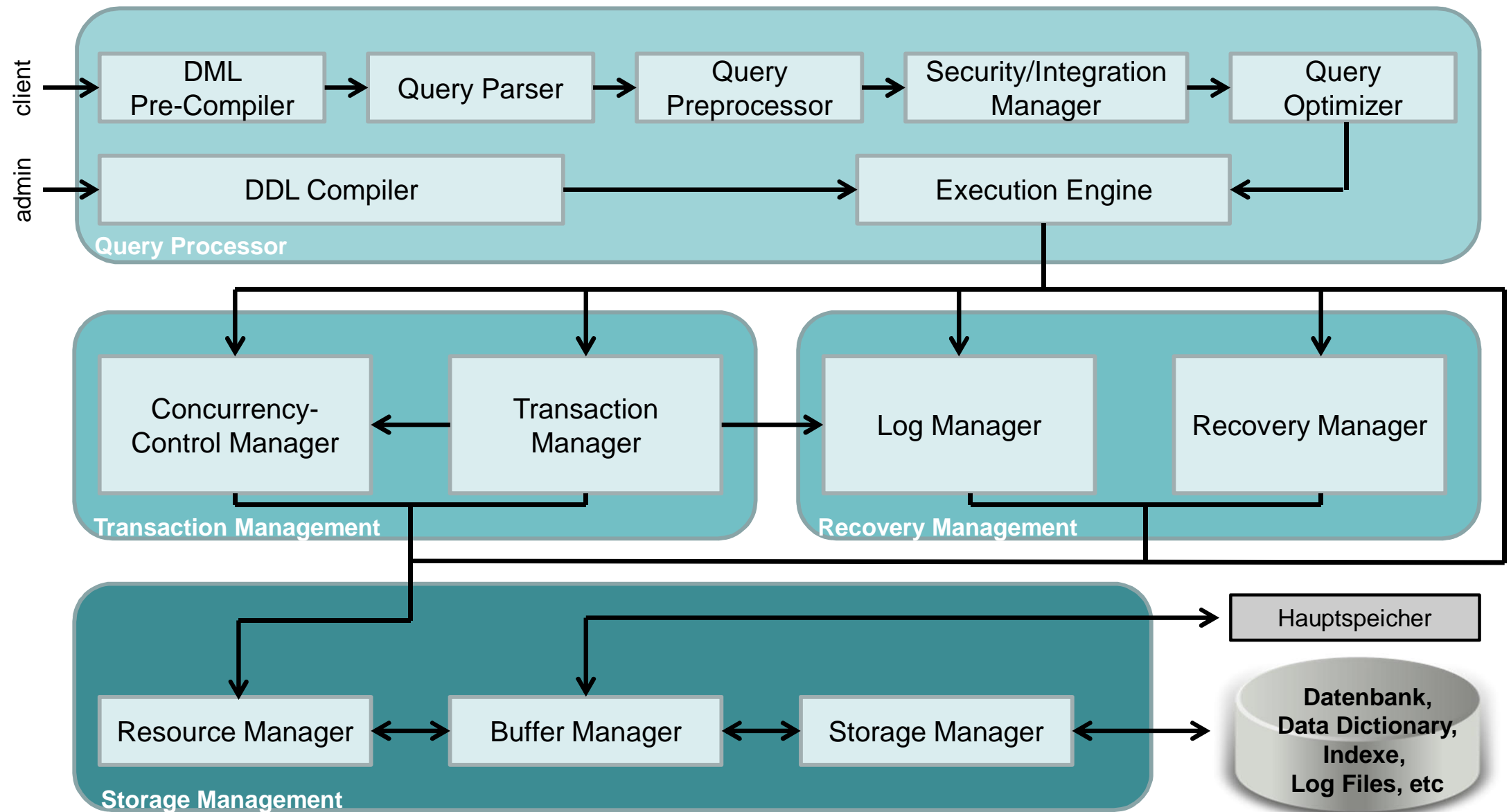




- ◆ **Beschreibung** der Komponenten eines Datenbanksystems
- ◆ **Standardisierung** der Schnittstellen zwischen Komponenten
- ◆ Architekturvorschläge
 - **ANSI-SPARC-Architektur** → Drei-Ebenen-Architektur (1978)
 - ANSI: American National Standards Institute
 - SPARC: Standards Planning and Requirement Committee
 - Im Wesentlichen eine Verfeinerung der vorgestellten 3-Ebene Architektur
 - Interne Ebene / Betriebssystem verfeinert
 - Mehr Interaktive und Programmier-Komponenten
 - Schnittstellen bezeichnet und normiert
 - **Fünf-Schichten-Architektur** (1987)
 - Beschreibt Transformations-Schritt im Detail
 - Im Rahmes des DBMS-Prototypen „System R“ von IBM vorgestellt
- ◆ Jedes DBMS hat seine eigene Architektur, häufig ähnlich zu den Architektur-Vorschlägen



Beispiel: Architektur von MySQL (vereinfacht)



Vereinfachte Darstellung basierend auf : R. Bannon et al.: MySQL Conceptual Architecture



Einige konkrete Systeme

- ◆ (Objekt-)Relationale DBMS
 - Oracle, IBM DB2, Microsoft SQL, MS Access
 - MySQL (www.mysql.org), PostgreSQL (www.postgresql.org),
 - Ingres (www.ingres.com), FireBird (www.firebirdsql.org)
 - CoreData (iOS), SQLite (Android)

- ◆ Objektorientierte DBMS
 - Poet, Versant, ObjectStore

- ◆ XML-DBMS
 - Tamino (Software AG), eXcelon



Begriff des Datenmodells

Ein Datenmodell (auch Datenbankmodell genannt) legt fest...

1) **Statische Eigenschaften:** Struktur der Daten

a) Objekte

b) Beziehungen

Beispiele: als Graph, in Relationen, als Objekte, als Schlüssel-Wert Paare, ...

2) **Dynamische Eigenschaften:** Operationen

a) Anfrage-Operationen (*queries*) und Änderungs-Operationen

b) Beziehungen zwischen Operationen

Beispiel: „gibt mir alle Kunden, die im letzten Quartal etwas bestellt haben“

3) **Integritätsbedingungen (*constraints*):** Bedingungen an

a) Objekte

b) Operationen

Beispiele: Werte des Attributs „Alter“ muss zwischen 1 und 150 liegen



Beispiele für Datenmodelle

- ◆ **Entwurfsmodelle:** Datenmodelle für den Entwurf von DBs
 - **ER-Modell** (Entity-Relationship-Modell)
 - UML (Unified Modeling Language)
- ◆ **Realisierungsmodelle:** Datenmodelle für die Implementierung von DBs
 - **Relationenmodell** / Relationales Modell
 - Hierarchisches Modell → legacy Datenbanken
 - Netzwerkmodell → legacy Datenbanken
- ◆ **Neuere Datenmodelle** für spezielle Anwendungen
 - Objektorientiertes Modell
 - Key-Values-Stores
 - Graph-Datenbanken



- ◆ Was ist eine Datenbank und welche Motivation sie einzusetzen gibt es?
- ◆ Was sind die Vorteile der Datenspeicherung in Relationen?
- ◆ Was versteht man unter der Schema-Architektur?
- ◆ Welche Architekturmuster im Zusammenhang mit Datenbanken gibt es und wozu sind sie gut?
- ◆ Was versteht man unter einem Datenmodell?