HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Datenbanksysteme 1

Einführung

Prof. Stefan Keller

Weshalb Datenbanken?

- Die gute und die schlechte Nachricht:
  - Die Festplattenkapazität verdoppelt sich bei leicht sinkenden Preisen etwa alle zwei bis fünf Jahre. Sie zeigt (wie auch Rechenleistung) einen exponentiellen Verlauf (Mooresches Gesetz).
  - Die Informationsmenge verdoppelt sich ca. alle 5 Jahre.
- D.h.
  - die „Grenze“, was als „grosse“ Datenbank gilt verschiebt sich,
  - die Herausforderungen der Verarbeitung grosser Datenmengen bleiben.
- Nennen Sie Beispiele einer Datenbasis!

Weshalb Datenbanken?

- Sie stecken hinter vielen Computeranwendungen
- bieten Mechanismus zum Speichern, Verwalten und Anfragen von Daten, ohne dass sich der Anwendungsprogrammierer um die Details kümmern muss
- versuchen Ordnung und Struktur in die Informationsflut zu bringen
- dienen als Grundlage für Informationssysteme
- Daher gehören Datenbanksysteme zu den Grundlagen eines Informatikstudium (vgl. Joint ACM and IEEE/CS Computing Curricula (Update 2008)

Ziele

- Nach dieser Vorlesungswoche werden Sie u.a.
  - Grundlegende Begriffe, wie Datenbanksystem, Datenbank, Datenbasis, Data Dictionary, Datenunabhängigkeit und Datenmodell definieren können,
  - Anwendungsfelder und Grundfunktionalitäten benennen können
  - wissen, welche Architekturkomponenten Datenbanksysteme im Allgemeinen besitzen,
  - wissen, was das ANSI 3-Ebenen-Modell bedeutet und wie man es anwendet

Inhalt

- Traditionelle Datenverarbeitung
- DBMS-Eigenschaften und -Anforderungen
- Datenbankmodelle (Paradigmen)
- 3-Ebenen-Modell
- Datenbank in Entwurf- und Betriebsphase
- Benutzer eines DB-Systems
- DB-System-Architekturen

Information

- Information
  - ist für ein Unternehmen von strategischer Bedeutung; ein Produktionsfaktor (wie Personal, Maschinen etc.); Grundlage für Steuerung, Kontrolle und Planung von Betriebsabläufen
  - Def.: „Informationen sind vom Empfänger interpretierte Daten“
- Daten
  - Sind maschinell verarbeitbar; weder wahr/falsch noch wichtig/unwichtig.
  - D. sind an Datenträger gebunden, die zur materiellen Verkörperung oder dauerhaften Aufnahme von Daten geeignete physikalische Mittel sind.
  - Lebensdauer der D.: mehrere Jahre > Lebensdauer Betriebssystem > Lebensdauer HW
  - Aufwand für Erfassung und Pflege kann enorm sein und die Aufwendungen für die SW-Entwicklung weit übersteigen
  - Daten müssen sicher und konsistent verwaltet werden

Daten ff.

- Es gibt einfache und komplexe (zusammengesetzte) "Daten„.
- Formatierten und unformatierte Daten. Beispiel:
  - Formatiert:
    - Feldname Pers-Nr Name Vorname Beruf Fam.-Stand Abt.
    - Daten (Feldinhalte) 4711 Maier Hans Bäcker verh. 10
  - Unformatiert:
    - „Der Mitarbeiter Hans Maier hat die Personalnummer 4711, er ist Bäcker, arbeitet in der Abteilung 10 und ist verheiratet.“
- Es gibt Systeme für formatierte und Systeme für unformatierte Daten: Datenbanksysteme sowie Volltextsuch- und Information Retrieval Systeme (z.T. überlappend)

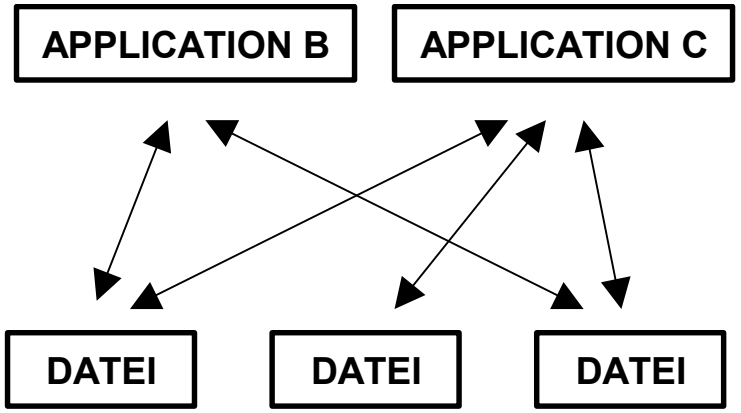
Datenbasis und Datenbanksystem

- Eine Datenbasis (Synonyme: Datenbestand, Datensatz) ist eine strukturierte Sammlung von Daten.

- Ein Datenbanksystem (Syn. Datenbank, DBS) besteht aus einem Datenbankmanagementsystem (DBMS) und einer (oder mehreren) Datenbasis/Datenbasen (DB).
  - (DB/Datenbank wird umgangsspr. leider oft auch als Datenbasis verstanden)
  - DBS: Datenbanksystem
  - DBMS: Datenbankmanagementsystem: anwendungsunabhängige Dienste
  - DB/Datenbasis: Anwendungsspezifische Informationen

- "Merkformel": DBS = DBMS + (n *) DB.

Datenhaltung in Dateien



Vorteile ?

Nachteile ?

- Daten sind in verschiedenen Dateien abgelegt
- Fehlende Kapselung, enge Kopplung Daten-Applikation
  - Applikationen greifen direkt auf die Daten in den Dateien zu
  - Änderungen in der Dateistruktur führt zu Anpassungen in den Applikationen.
- Kompliziertere Abfragen erfordern Zugriff auf mehrere Dateien
- Technologien:
  - Format: Textdateien, Binäre Dateien, Serialisierung, XML
  - Zugriff: sequentiell, random access

Warum nicht…?

- Schon eine eigene Datenbank erstellt?
- Warum genügt eine Applikation nicht?
  - Warum nicht eine Java Collection?
- Warum nicht Text im Texteditor?
  - Einfachste Form der "Datenhaltung“. Suche mittels "pattern matching"
  - Inhaltliche Frage wie "Wer ist Chef der Abt. X?" lässt sich nicht beantworten.
  - Änderungen durch Zeichenersetzungen im Texteditor möglich.
    - In einer "echten" Datenbank dagegen, werden Daten "bewusst" in strukturierter Form gehalten. Man kann Zeilen, Spalten und Felder voneinander unterscheiden und sie individuell manipulieren. Spalten- und Tabellenbezeichner sind von Feldeinträgen unterschieden und haben eine eigenständige Bedeutung.

=> Datenbanken verwalten auch „nur“ Dateien im Dateisystem.

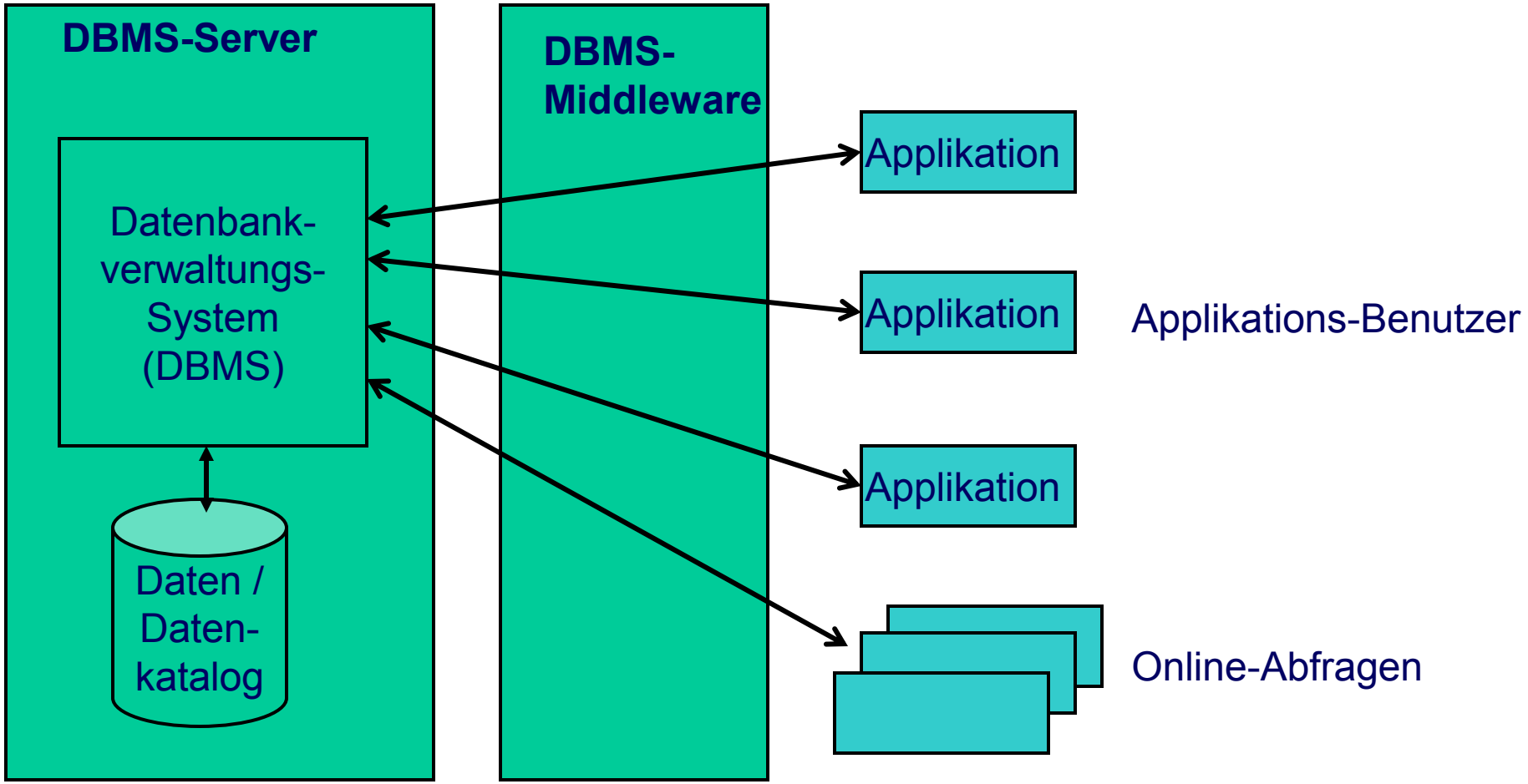
Motivation für den Einsatz eines DBMS

- Enthält eine "Suchmaschine" wie Google ein Datenbanksystem?
- Ist MS Access eine Datenbank?
- Informatiker müssen genauso mit Datenbanken umgehen können, wie sie programmieren können (müssen):
  - SQL und DB-Programmierung ist (fast) so wichtig wie Java oder C#, C++ - oder sonst eine ‚coole' Sprache wie Python…!

Motivation für den Einsatz eines DBMS

- Typische Probleme bei Informationsverarbeitung mit Dateien
  - Redundanz zwischen den Daten in den verschiedenen Dateien => Inkonsistenzen
  - Mehrbenutzerbetrieb: Dateien auf Fileserver, keine vernünftige Synchronisation bei gleichzeitigem Zugriff
  - Verlust von Daten: Auch bei regelmässigem File-Backup können die Änderungen zwischen zwei Backups verloren gehen
  - Integritätsverletzung: Fehler in den Applikationen können zu korrupten Daten führen.
  - Sicherheitsprobleme: Zugriffschutz nur auf Stufe Datei möglich
  - hohe Entwicklungskosten für Anwendungsprogramme: Änderungen an Dateiformat bedingt Anpassungen der Applikationen
- Hauptanwendung
  - Speicherung von lokalen Daten einer Applikation (Serialisierung)
  - Benutzer- und Applikationseinstellungen

Systemübersicht DBMS



Eigenschaften einer DBMS-Lösung

- Zentrale Datenbasis (Datenbank), verwaltet durch das DBMS
- Anwendungen greifen nicht direkt auf die Daten zu, sondern via DBMS, d.h. Änderungen der Datenorganisation haben keine Programmänderungen zur Folge (Kapselung, Datenunabhängigkeit der Applikationen)
- Die Daten sind strukturiert und die Struktur ist im Datenkatalog beschrieben. Der Datenkatalog (oder Data Dictionary) ist Bestandteil der Datenbank und enthält alle Daten über die gespeicherten Daten (Meta-Daten)
- Datenorientiert, d.h. Datenflüsse und Verarbeitungsreihenfolge sind unabhängig
- Client-Server Struktur / Mehrbenutzerbetrieb: Gleichzeitiger Zugriff mehrerer Clients (Applikationen, Benutzer) auf die Daten des DBMS
- Kontrollierte Redundanz in den Daten, Datenintegrität wird zentral vom DBMS gesichert
- Datenpflege, Datenschutz und Datensicherheit kann zentral vom DBMS sichergestellt werden

Anforderungen an eine DBMS-Lösung

- Redundanzfreiheit
  - Jedes Datenelement soll nur einmal gespeichert sein.
  - Zeitliche Anforderung in der Verarbeitung stehen dieser Forderung oftmals entgegen und bedingen eine kontrollierte Redundanz
- Datenintegrität
  - Die Daten in einer Datenbank stellen eine wertvolle Ressource dar. Sie werden aber rasch unbrauchbar, wenn ihre Integrität nicht gewährleistet wird.
    Integrität umfasst:
    - Datenkonsistenz = logische Widerspruchsfreiheit der Daten
    - Datensicherheit = Sicherung der Daten vor physischem Verlust.
    - Datenschutz = Schutz der Daten vor unberechtigtem Zugriff

DBMS-Funktionen

- Beim DBMS sind alle Daten zentral gespeichert. Welche Funktionen sind nötig, damit dieser zentralisierte Ansatz funktioniert?
  - Transaktionen
    - konsistenzerhaltende Operationen ("alles oder nichts")
  - Synchronisation von parallelen Zugriffen (Mehrbenutzerbetrieb)
  - Sicherheit: Authentifizierung und Autorisierung
  - Backup und Recovery
  - 'generische' Datenstrukturen
    - Definition von Datenstrukturen (deskriptiv)
    - Datenkatalog (Metadata, Data Dictionary)
  - Abfragesprache, Programmierschnittstelle

Datenbankmodelle – Überblick

- Datenbankmodell
  - legt die Datenstrukturierungs-Konzepte des DBMS fest (Beziehungen)
  - legt die Datenstrukturen im DBMS fest (Datentypen, Speicherung)
  - auch Paradigma (in UML 2: Meta-Modell) genannt
- DBMS unterstützen der folgende Modelle (Paradigmen):
  - Hierarchisches Datenbankmodell
  - Netzwerkmodell
  - Relationenmodell
  - Postrelationale Modell
    - Objektrelationales Modell
    - Objektorientiertes Modell
  - NoSQL

1. Hierarchisches Datenbankmodell

- Strukturen
  - Die zu speichernden Informationen werden in einer einzigen Hierarchie organisiert
  - Beziehungen innerhalb der Hierarchie werden zusammen mit den Daten gespeichert
  - Ermöglicht innerhalb der Hierarchiestufen eine einfache, sequentielle Verarbeitung
- Nachteile:
  - Die ‚Welt' lässt nicht immer als Hierarchie modellieren
  - Keine Trennung zwischen Anwendung (externes Schema) und Daten (internes Schema)
  - Aufwendige Anpassungsarbeiten bei Änderungen von Datenstrukturen und/oder Algorithmen
  - heute nur noch in ‚legacy systems' anzutreffen (IBM-IMS)
- Aber: Totgesagte leben länger: Revival mit XML-DB!

2. Netzwerk-Datenbankmodell

- Struktur
  - Die zu speichernden Informationen werden in vernetzten Hierarchien organisiert
  - Beziehungen innerhalb der Hierarchie werden zusammen mit den Daten gespeichert, wobei Mehrfachbeziehungen zwischen Datenelementen möglich sind, was eine wirklichkeitsgetreuere Modellierung ermöglicht
  - Gut geeignet für die effiziente Verwaltung von strukturierten und vernetzten Daten
- Nachteile:
  - Effiziente Anwendungen nur möglich in Kenntnis der komplexen Netzwerk-Datenstruktur, dadurch starke Abhängigkeiten zwischen Anwendung (externem Schema) und Daten (internem Schema)
  - Praktische keine kommerzielle Bedeutung; wurden durch OODBMS verdrängt

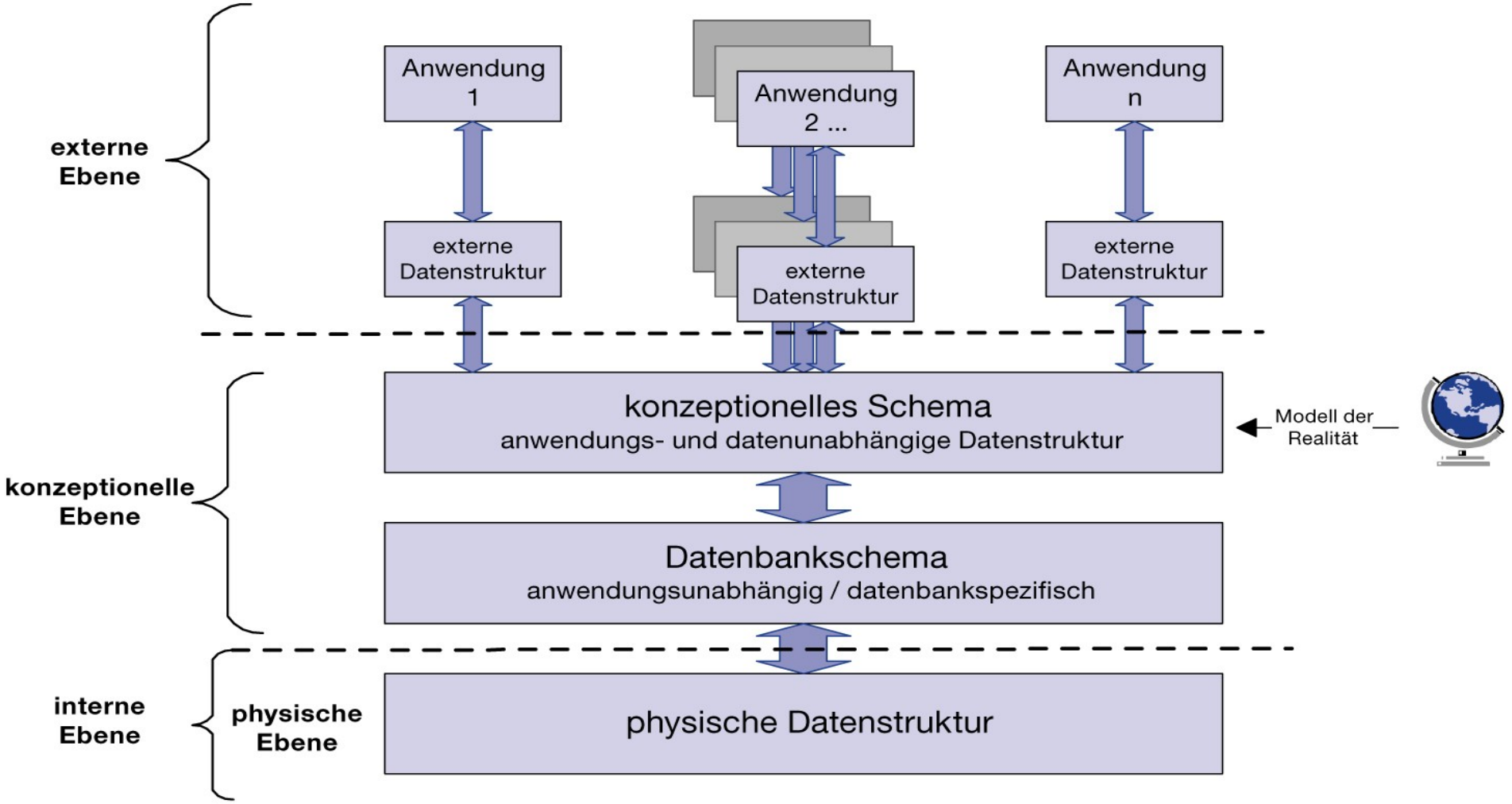
3. Relationales Datenbankmodell

- Struktur
  - alle Informationen werden als unstrukturierte Daten (1. Normalform) in Form von Tabellen (=Relationen) gespeichert (basiert auf theoretischen Arbeiten von E.F.Codd, 70er Jahre)
  - sehr flexibel, praktisch alle möglichen Abfragen auf den Daten möglich
  - ermöglicht eine klare und genaue Trennung zwischen Daten und Anwendungen
- Nachteile:
  - Impedance Mismatch zwischen Typensystem der Programmiersprachen und den Tabellenstrukturen führt zu aufwändigen Konversionen beim Zugriff auf die Daten über ein Programm
  - Komplexe Abfragen können zu Performance-Problemen führen
- Marktanteil
  - alle gängigen Datenbanksysteme basieren auf dem relationalen Modell

4. Postrelationale Datenbankmodelle

- Objektrelationale Modelle
  - Erweiterung des relationalen Datenmodelles: benutzerdef. Typen, Tabellen-Vererbung
  - Verschachtelung von Tabellen erlaubt (d.h. die 1. Normalform muss nicht erfüllt sein).
  - neben den Daten (=Objekt und Attribute) können auch Methoden gespeichert werden.
  - Evolutionärer Ansatz: Realisierung als Erweiterung eines relationalen Datenbanksystems um OO-Konzepte (Bsp. Oracle 8i ff., IBM DB2)
- Objektorientierte Modelle
  - Modelle basieren auf einer OO-Sprache wie C++ oder Java.
  - DB-Funktionalitäten für Transaktionen, Persistenz, parallelen Zugriff etc.
  - Revolutionärer Ansatz: Neuimplementierung
  - Marktanteil: gering (im Prozentbereich), nur für Spezialanwendungen geeignet

Das ANSI-3-Ebenen-Modell (1 von 3)



Quelle: Faeskorn et al. Datenbanksysteme

3-Ebenen-Modell (1 von 3)

- Logische Ebene:
  - logische Struktur der Daten
  - Definition durch logisches Schema („Trägermodell“)
- Interne Ebene:
  - Speicherungsstrukturen
  - Definition durch internes Schema
- Externe Ebene:
  - Sicht einer Benutzerklasse auf eine Teilmenge der Datenbank
  - Definition durch externes Schema
- Mapping
  - Zwischen den Ebenen ist eine mehr oder weniger komplexe Abbildung notwendig

3-Ebenen-Modell (1 von 3)

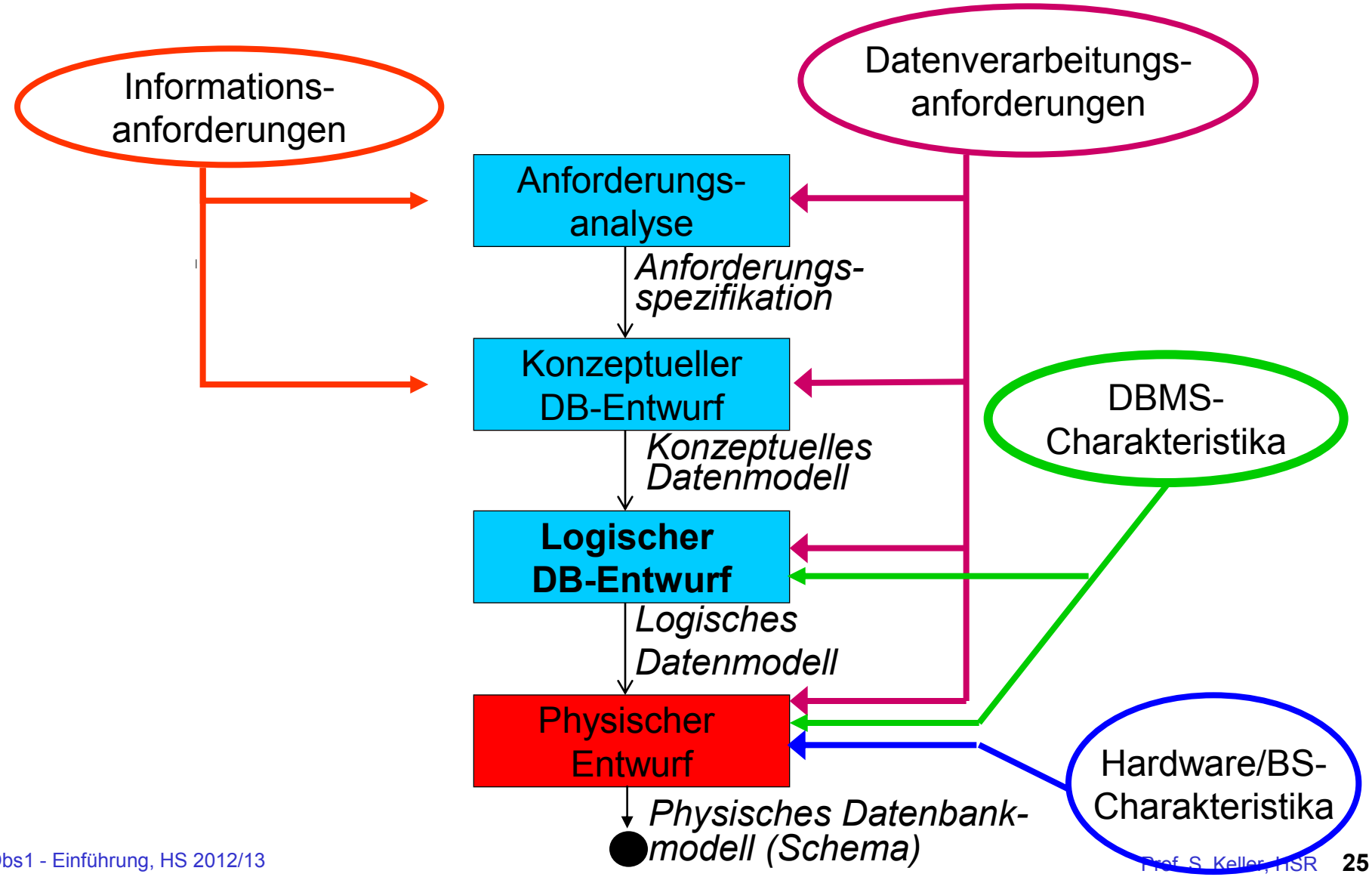
- Wieso diese Aufteilung? ➔ Datenunabhängigkeit:
  - Die Separation zwischen den einzelnen Schichten erlaubt, dass eine Schicht (in einem beschränkten Rahmen) reorganisiert werden kann, ohne dass die darüber liegenden Schichten angepasst werden müssen.
  - Verschiedene Benutzer haben verschiedene Sichten derselben Datenmenge. Die Art, wie ein Benutzer die Daten sieht, kann sich im Laufe der Zeit ändern.
  - Der DBA sollte die logische Struktur der Datenbank reorganisieren können, ohne dass die Benutzer davon tangiert werden.
  - Der DBA sollte die Speicherungsstruktur ändern können, ohne dass die logische Struktur davon beeinflusst wird.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

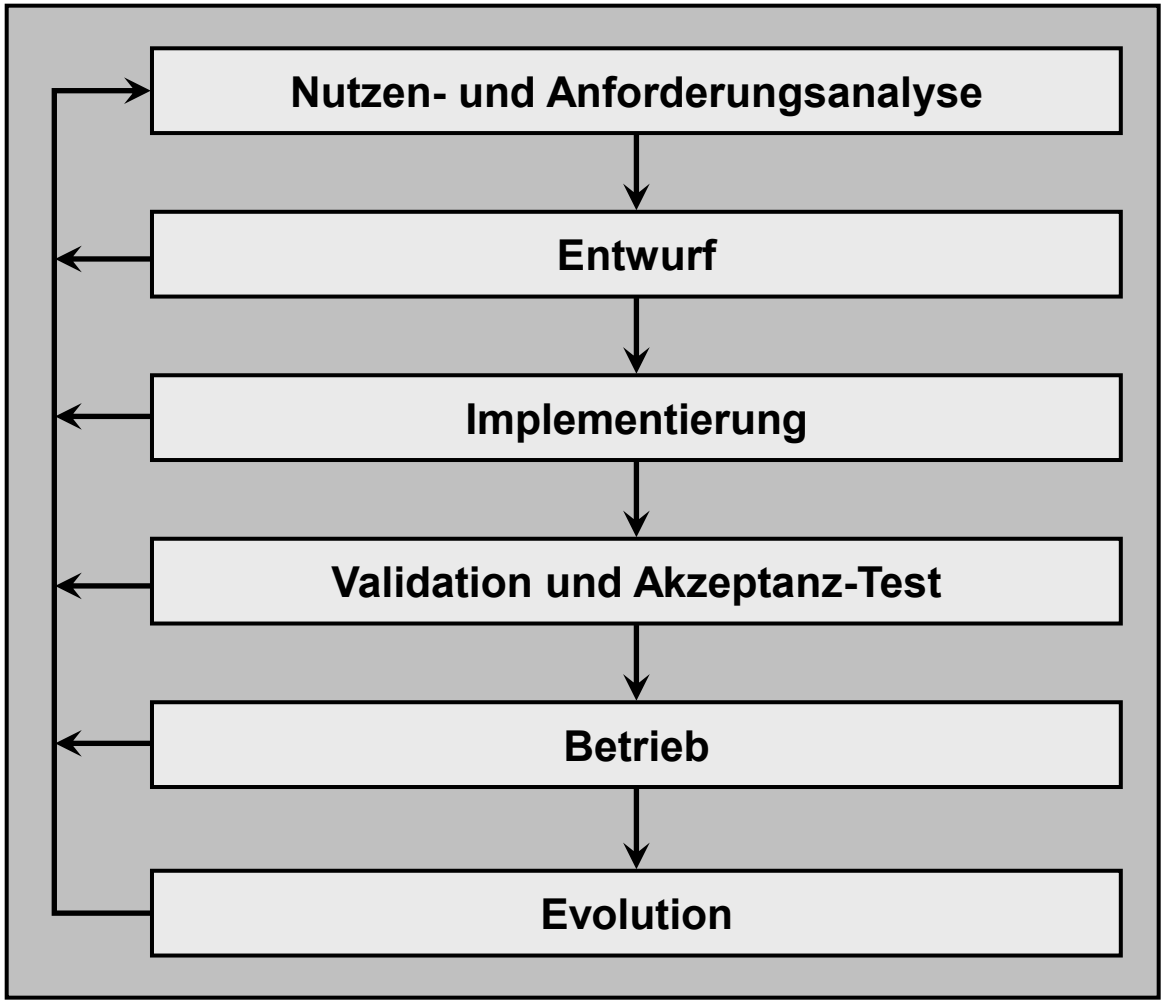
INFORMATIK

Der DB-Entwurfsprozess

DB-Entwurfsprozess

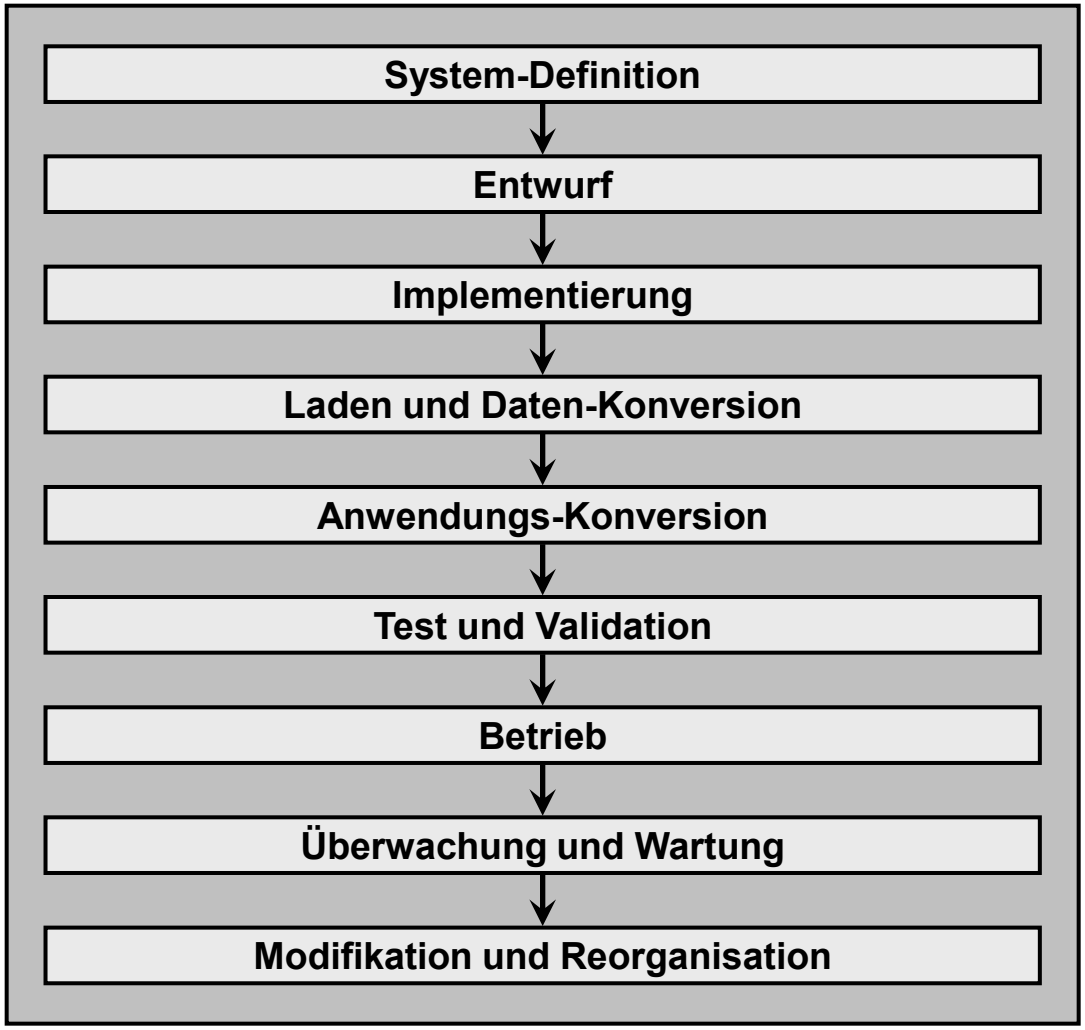


Informationssystem-Lebenszyklus



Quelle: G. Vossen

Datenbankanwendungs-Lebenszyklus



Entwurfsphase: Datenbank-Entwurf

- Definition der Datenbankstrukturen
  - Datenstrukturen (Schema) werden mit einer Data Definition Language (DDL) beschrieben.
  - Der DDL-Compiler übersetzt die DDL-Beschreibungen und generiert daraus die Struktur der Datenbank. Die Strukturinformationen werden als Systemdaten vom DBMS verwaltet.
  - Die Systemdaten umfassen den Datenkatalog (Data Dictionary mit Datenbeschreibungen, Zugriffsberechtigungen, Referenzdaten etc.)

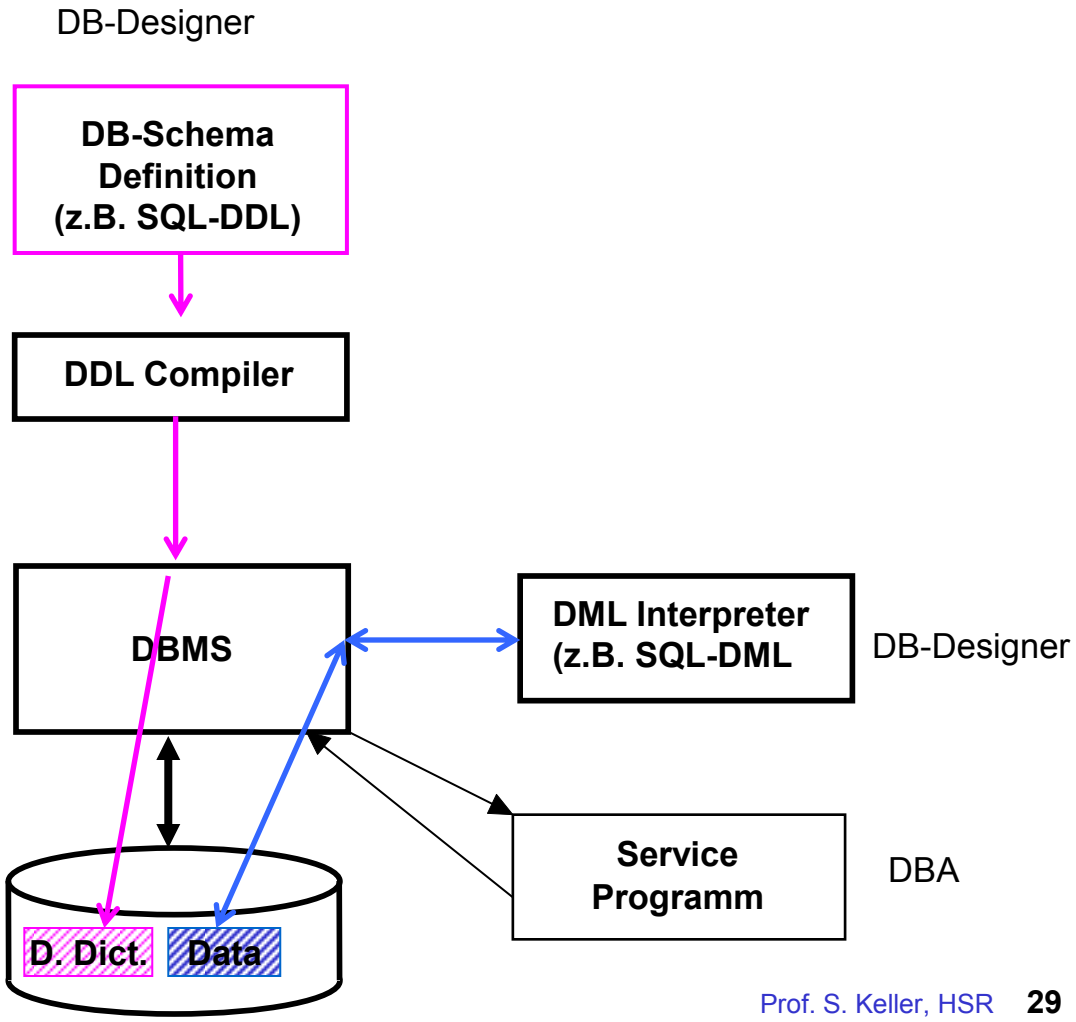
Entwurfsphase: Datenbank-Entwurf

Aufgaben

- Entwurf des DB-Schemas (z.B. mit UML oder ER)
- Definition des DB-Schemas mit DDL (z.B. SQL-DDL)
- Generierung der Datenbank
- Eingabe von Testdaten (z.B. mit SQL-DML)
- Definition und Test von Abfragen und Views (z.B. mit SQL)

Wer

- DB-Designer, DBA



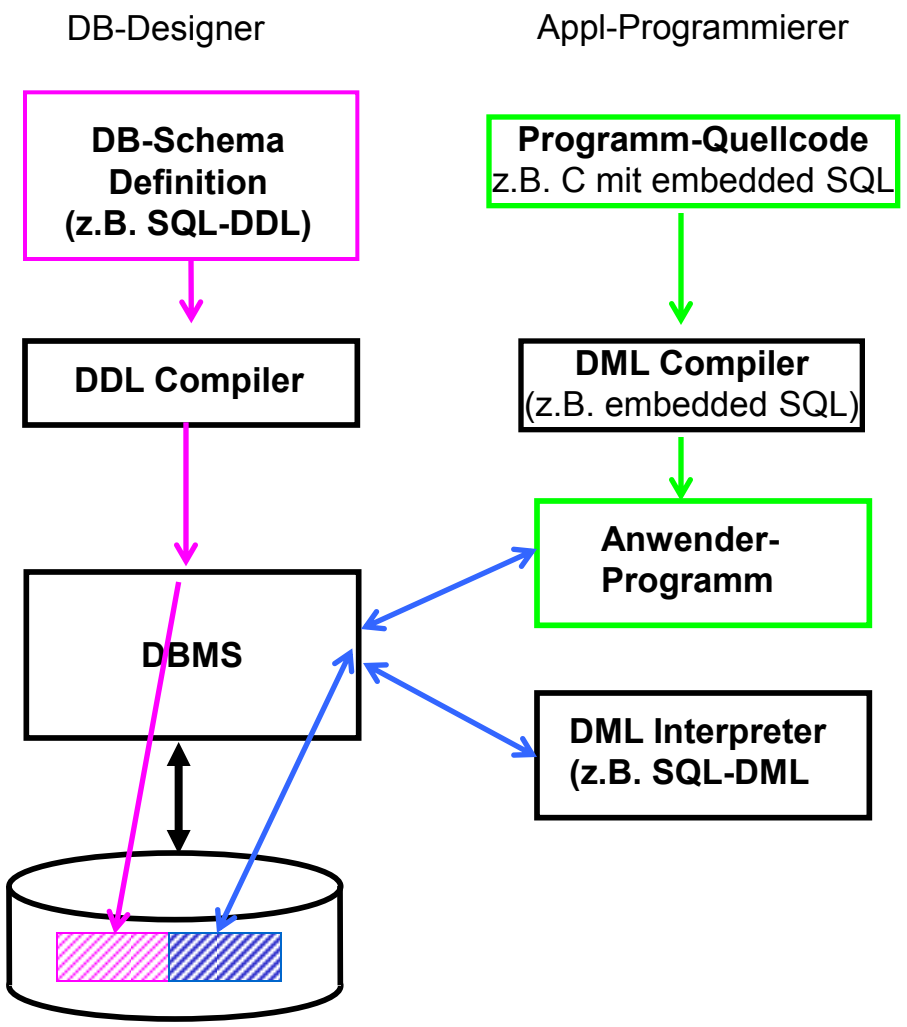
Beispiel DDL (SQL)

```
CREATE TABLE Dept_tab (
    Deptno  NUMBER(3) CONSTRAINT Dept_pkey PRIMARY KEY,
    Dname   VARCHAR2(15),
    Loc     VARCHAR2(15),
    CONSTRAINT Dname_ukey UNIQUE (Dname, Loc),
    CONSTRAINT Loc_check1
                CHECK (loc IN ('NEW YORK', 'BOSTON', 'CHICAGO'))
);
CREATE TABLE Emp_tab (
    Empno    NUMBER(5) CONSTRAINT Emp_pkey PRIMARY KEY,
    Ename    VARCHAR2(15) NOT NULL,
    Job      VARCHAR2(10),
    Mgr      NUMBER(5) CONSTRAINT Mgr_fkey REFERENCES Emp_tab,
    Hiredate DATE,
    Sal      NUMBER(7,2),
    Comm     NUMBER(5,2),
    Deptno   NUMBER(3) NOT NULL
     CONSTRAINT dept_fkey REFERENCES Dept_tab ON DELETE CASCADE
);
```

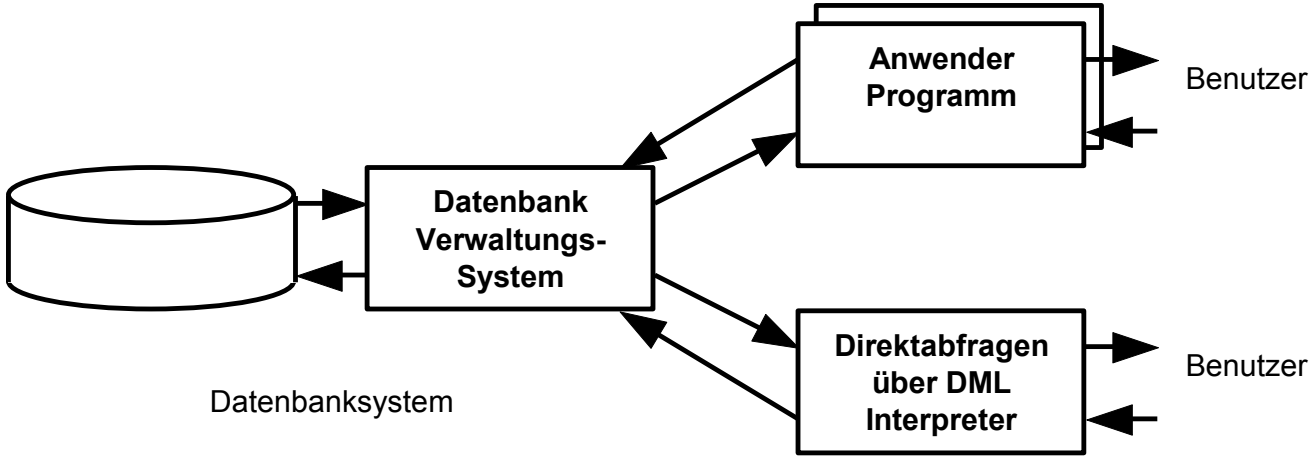
Entwurfsphase: Entwurf der DB-Applikationen

- Entwicklung der DB-Appliaktionen
  - Call-level interface (API)
    - Die Anwenderprogramme greifen über eine Programmierschnittstelle auf die Datenbank zu.
    - Bsp.: JDBC, ODBC
    - Zur Laufzeit wird die Bibliothek mit den DB-Zugriffsroutinen und der DB-Middleware dazu gelinkt.
  - Embedded SQL
    - Die Anwenderprogramme werden z.B. in C/C++ geschrieben und um spezielle Konstrukte für den DBMS-Zugriffe angereichert.
    - Der DML-Precompiler analysiert die DBMS-Zugriffe und übersetzt sie in Aufrufsequenzen von DBMS-Zugriffsprozeduren. Anschliessend können die Programme normal kompiliert und mit der DB-Bibliothek gelinkt werden.

Entwurfsphase: Entwurf der DB-Applikationen



Datenbank in Betriebsphase



Betriebsphase

Die Anwenderprogramme greifen über die Prozedurschnittstelle (API=Application Programming Interface) auf die Datenbank zu.
Der DML-Interpreter übersetzt interaktiv in DML (Data Manipulation Language) formulierte Abfragen und ruft ebenfalls Prozeduren des API auf.

Beispiel DML (SQL)

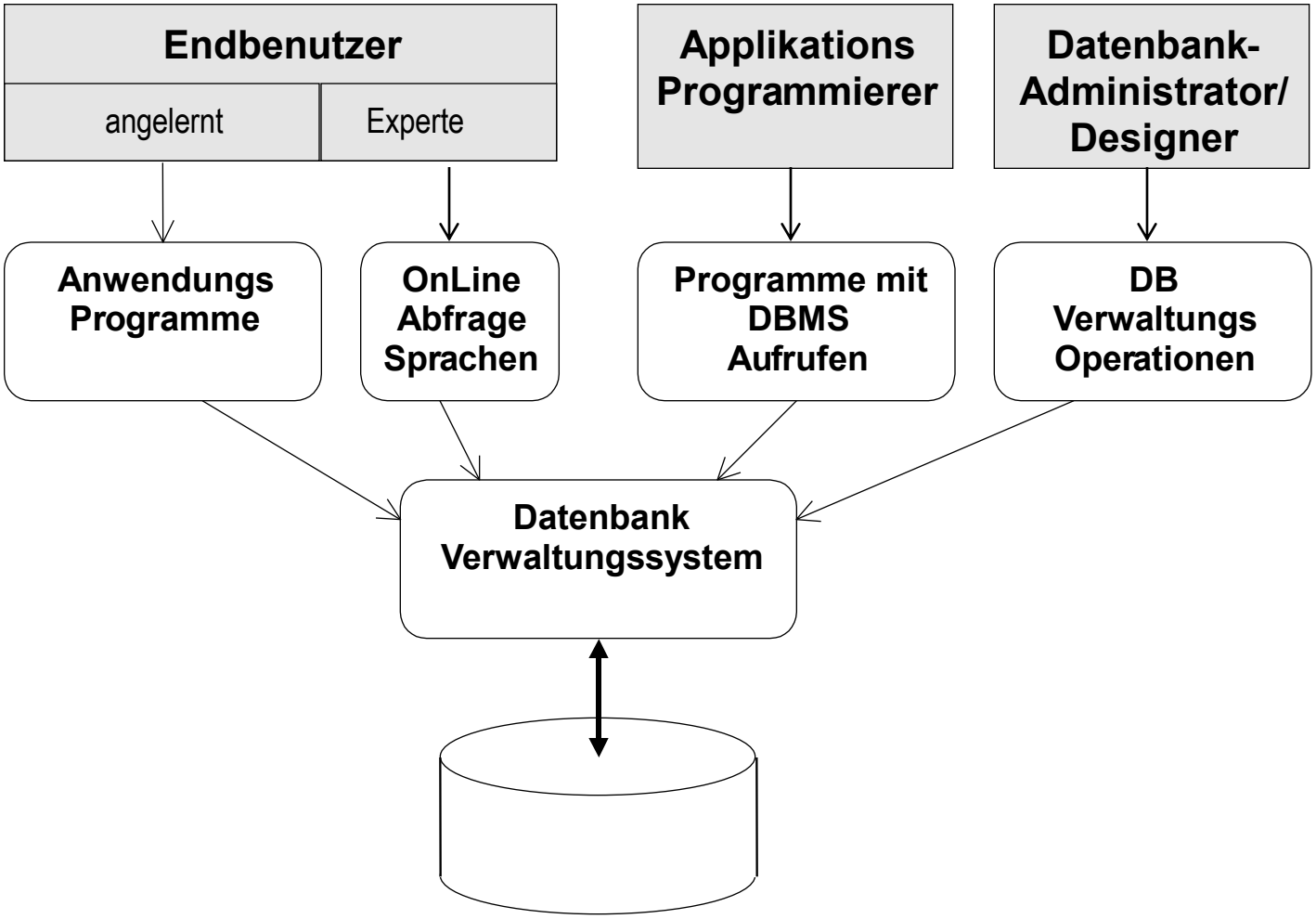
```
UPDATE Emp_tab
SET sal = sal * 1.1
WHERE deptno IN
(SELECT deptno FROM Dept_tab WHERE loc = 'DALLAS');
```

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Benutzer und Architekturen von DB

Benutzer eines DBMS



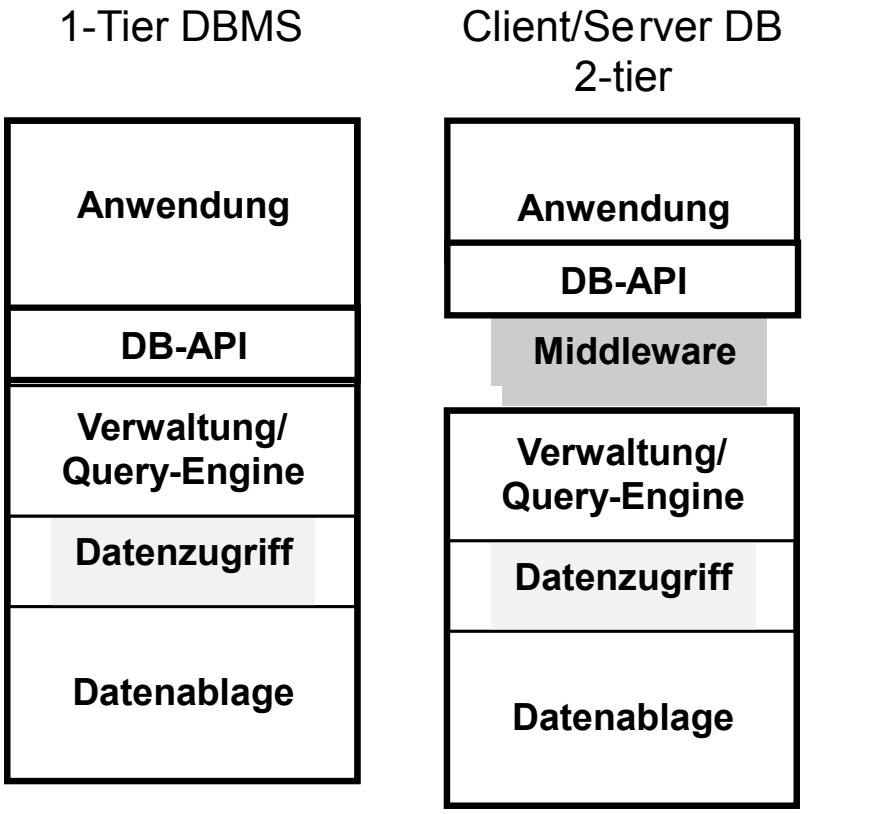
Benutzer eines DBMS

- Fortgeschrittener Benutzer (DB-Experte)
  - hat Datenbankwissen, kann über spezielle Abfragesprachen Datenbankabfragen oder Mutationen ausführen.
  - Abfrage = Query , Abfragesprache Query-Language, Bsp. SQL Structured Query Language
- Angelernter Benutzer (Applikations-Benutzer)
  - (z.B. Bankangestellter): benutzt vorgefertigte Applikationen und kann damit bestimmte Abfragen und Mutationen ausführen.
- Applikations-Programmierer
  - erstellen DB-Applikationen. Dazu werden einerseits Programmiersprachen wie C/C++, Cobol, Java oder spezielle DB-Entwicklungssysteme (sog. 4 GL Sprachen) verwendet.
- Datenbankadministrator:
  - Der DBA (oder das DBA-Team) ist zuständig für den Entwurf, das Generieren und das Betreiben der Datenbank. Der DBA hat "superuser"-Privilegien und kann die Zugriffsrechte der verschiedenen Benutzer festlegen.

DB-System-Architekturen

Client-Server System

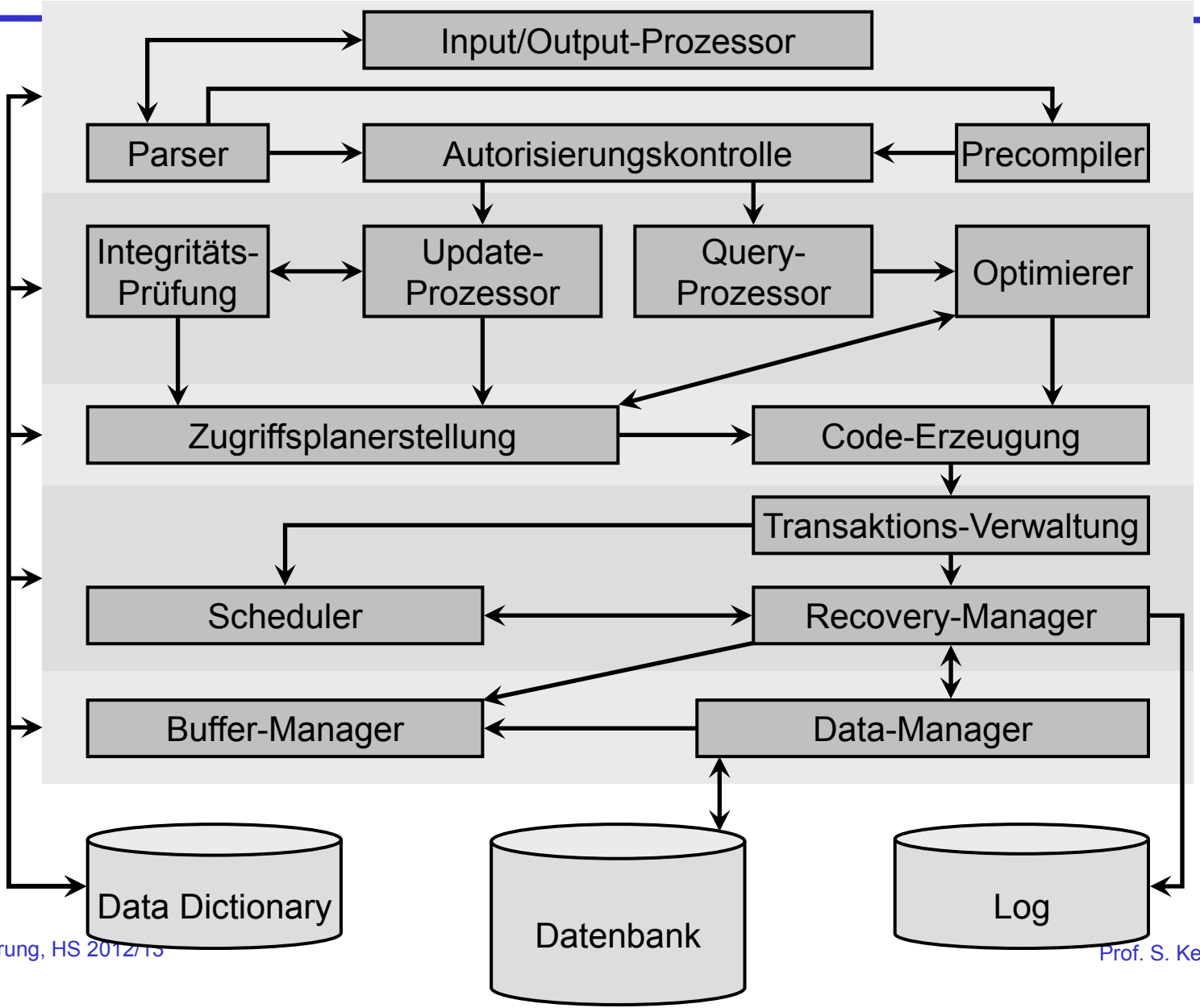
Client
Server
DB-API
Middleware



DB-Systeme wie
dBase, Access,
SQLite, etc.

Oracle, Db2
MS SQL Server
PostgreSQL, MySQL, etc.

DBMS: Innere Architektur, Begriffe und Subsysteme



Quelle: G. Vossen