Hasso Plattner Institut

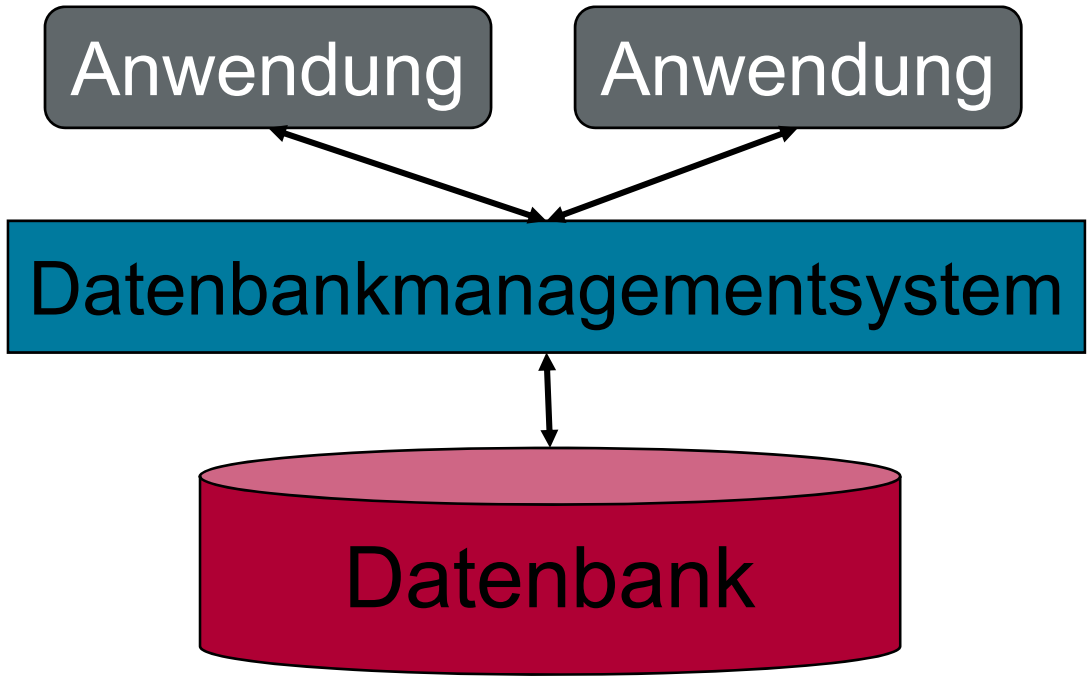
IT Systems Engineering | Universität Potsdam

Datenbanksysteme 1
Organisatorisches und Einführung

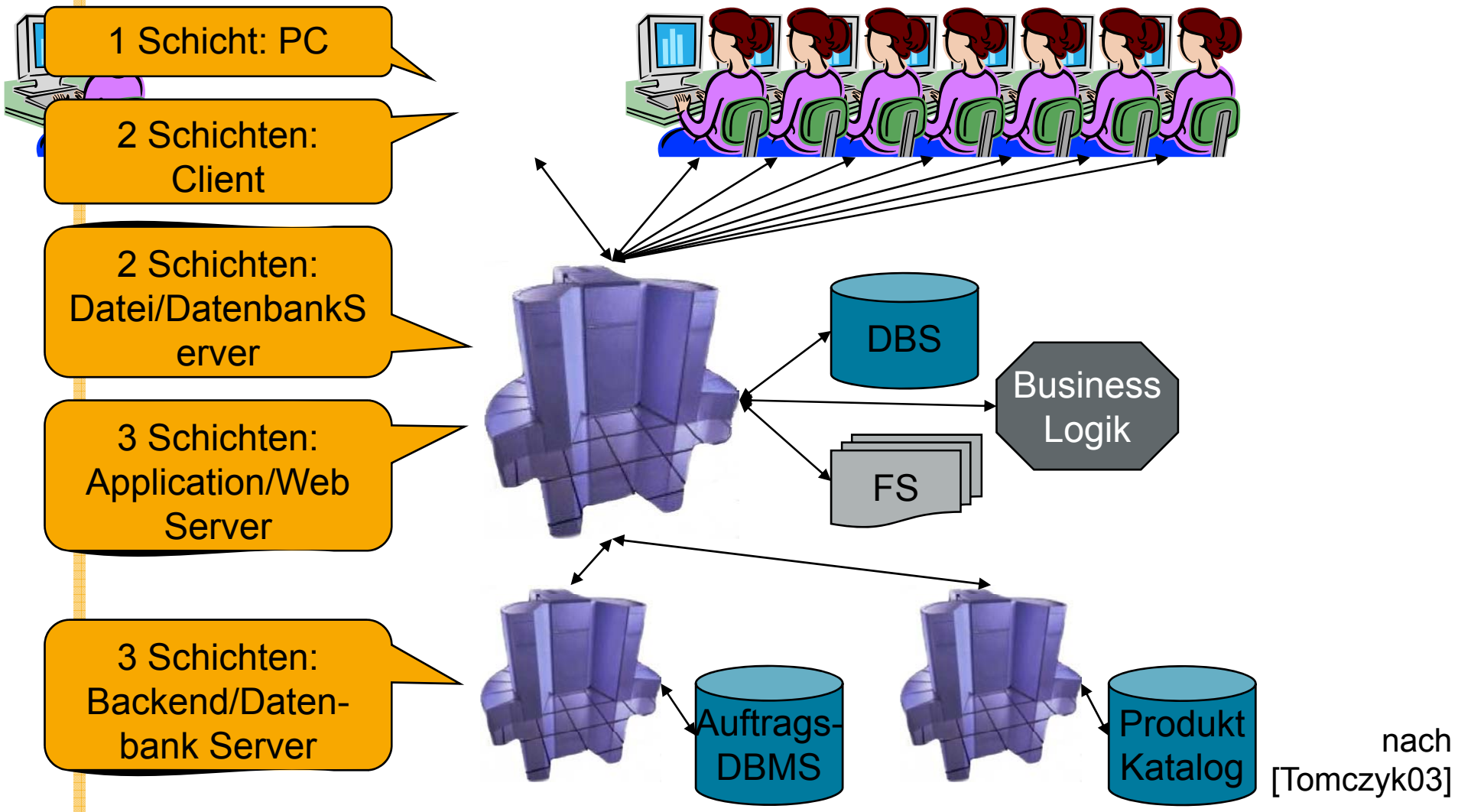
14.4.2008

Felix Naumann

Datenbanken

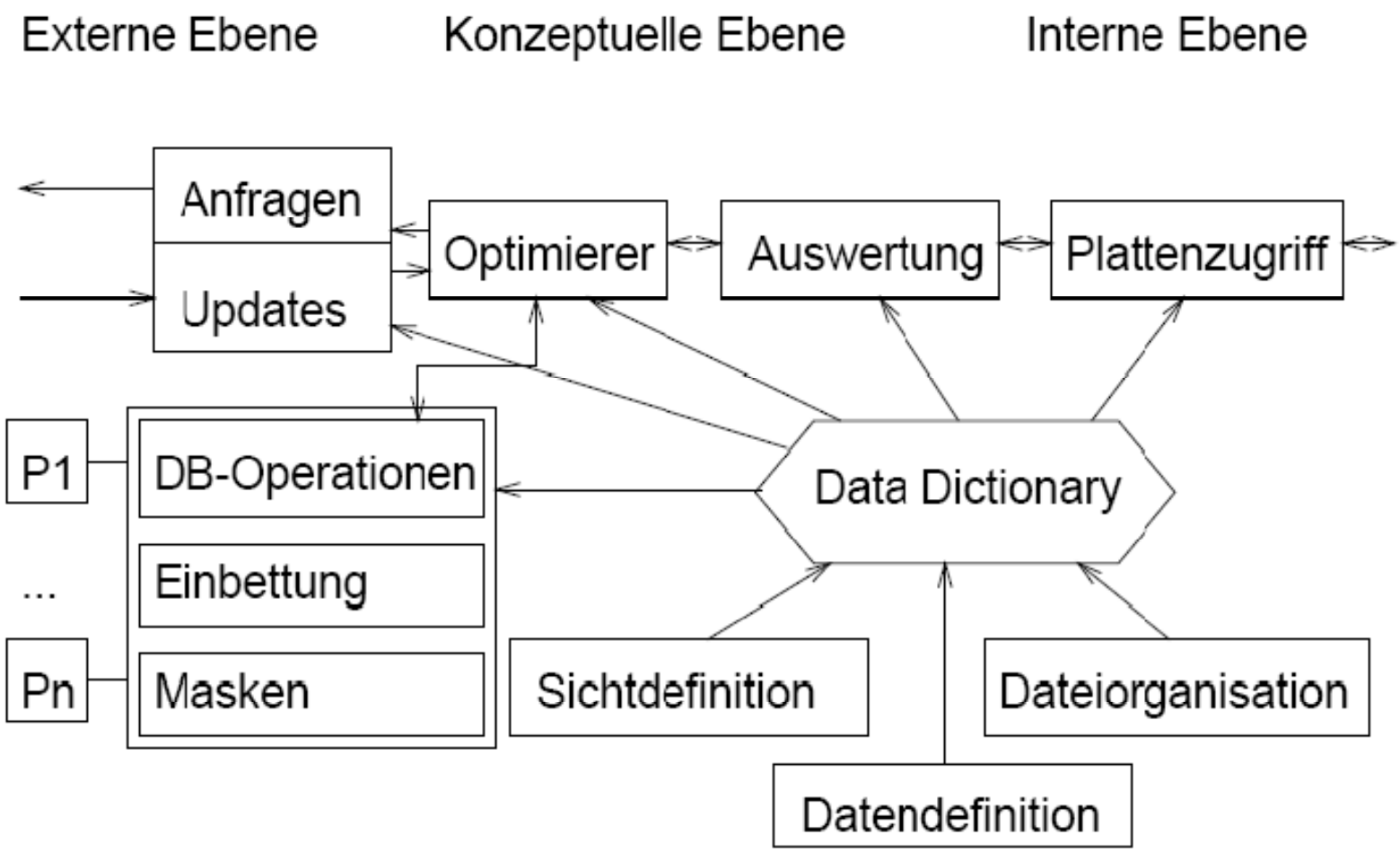


Zoom heraus: Mehr-Schichtenarchitekturen

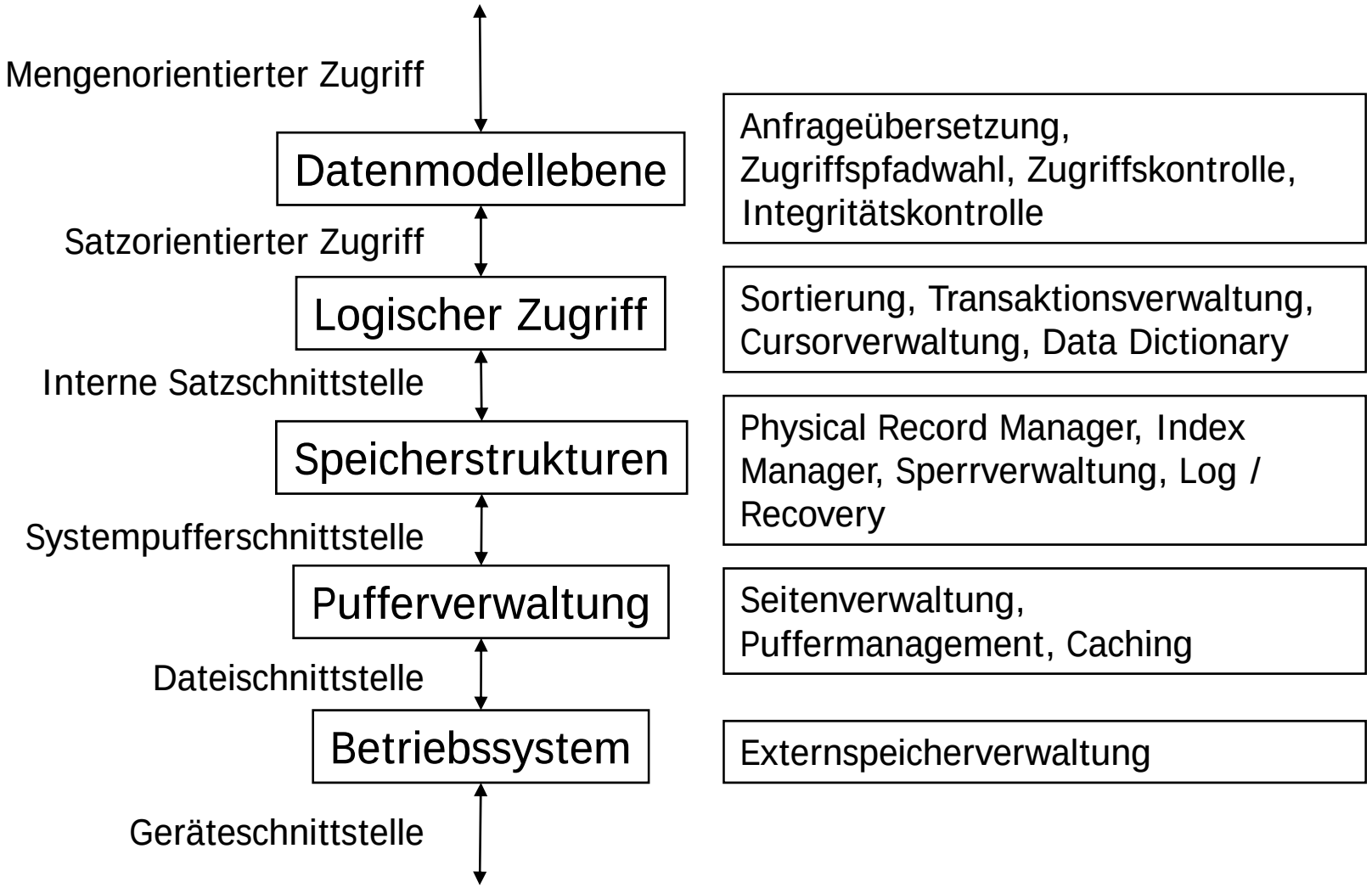


nach [Tomczyk03]

Zoom herein: ANSI/SPARC Architektur eines DBMS



Zoom in die interne Ebene: Die 5-Schichten Architektur



Überblick

- Vorstellung der Arbeitsgruppe
- Organisatorisches
- Implementierung von Datenbanken
- Ausblick auf das Semester

Arbeitsgruppe Informationssysteme

Paul **Führing**

project **ViQTOR**

Patricia **Hobro**

Prof. Felix **Naumann**

Jens **Bleiholder**

project **fusem**

Karsten **Draba**

DQ Assessment

Data Fusion

Information Integration

project **HumMer**

Information Quality

Duplicate Detection

Melanie **Weis** & Sascha **Szott**

Data Cleaning

Armin **Roth**

project **System P**

Peer Data Management Systems

Matching

Data Integration for Life Science Data Sources

project **Aladin**

Service-Oriented Systems

Ontologies

Mohammed **AbuJarour**

Frank **Kaufer**

Jana **Bauckmann**

Alexander **Albrecht**

project **XClean**

Personal Information Management

Data Profiling for Schema Management

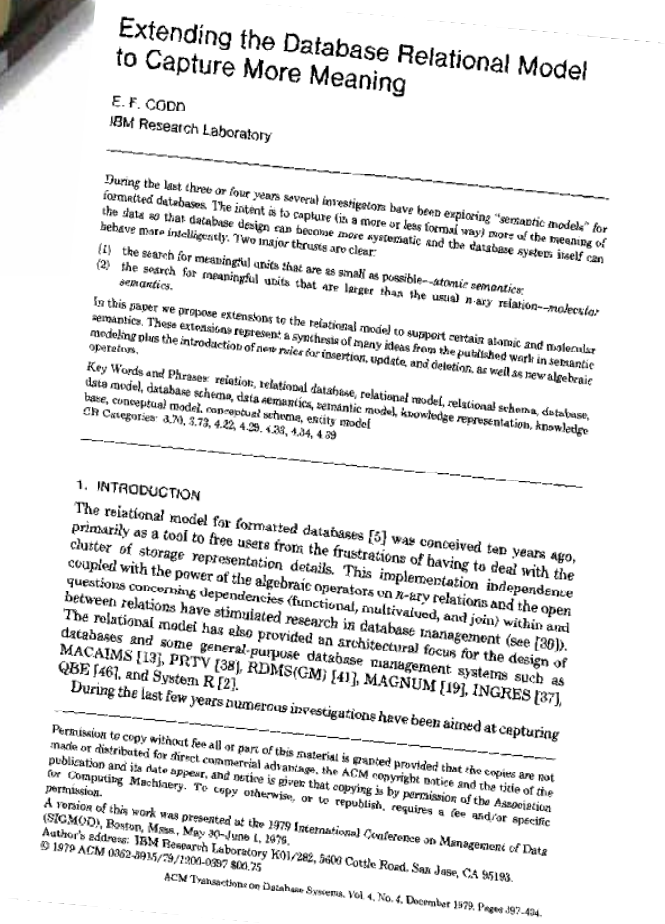
Lehrveranstaltungen in diesem Semester

Vorlesungen

- DBS II
- Informationsintegrati

Seminare

- Bachelor: Beauty is our Business
- Bachelor: www.ligageschichte.de
- Master: Duplikaterkennung
- Forschungsseminar



Extending the Database Relational Model to Capture More Meaning

E. F. CODD

IBM Research Laboratory

During the last three or four years several investigators have been exploring "semantic models" for formatted databases. The intent is to capture (in a more or less formal way) more of the meaning of the data so that database design can become more systematic and the database system itself can behave more intelligently. Two major thrusts are clear:

(1) the search for meaningful units that are as small as possible—*atomic semantics*;
(2) the search for meaningful units that are larger than the usual *n*-ary relation—*molecular semantics*.

In this paper we propose extensions to the relational model to support certain atomic and molecular semantics. These extensions represent a synthesis of many ideas from the published work in semantic modeling plus the introduction of *new rules* for insertion, update, and deletion, as well as new algebraic operators.

Key Words and Phrases: relation, relational database, relational model, relational schema, database, data model, database schema, data semantics, semantic model, knowledge representation, knowledge base, conceptual model, conceptual schema, entity model

CR Categories: 3.70, 3.73, 4.22, 4.29, 4.33, 4.34, 4.39

1. INTRODUCTION

The relational model for formatted databases [5] was conceived ten years ago, primarily as a tool to free users from the frustrations of having to deal with the clutter of storage representation details. This implementation independence coupled with the power of the algebraic operators on *n*-ary relations and the open questions concerning dependencies (functional, multivalued, and join) within and between relations have stimulated research in database management (see [30]). The relational model has also provided an architectural focus for the design of databases and some general-purpose database management systems such as MACAIMS [13], PRTV [38], RDMS(GM) [41], MAGNUM [19], INGRES [37], QBE [46], and System R [2].

During the last few years numerous investigations have been aimed at capturing

A version of this work was presented at the 1979 International Conference on Management of Data (SIGMOD), Boston, Mass., May 30–June 1, 1979.

Author's address: IBM Research Laboratory K01/282, 5600 Cottle Road, San Jose, CA 95193.

ACM Transactions on Database Systems, Vol. 4, No. 4, December 1979, Pages 397–434.

Vorstellung – Hörer

- Welches Semester?
  - 2tes, 4tes, 6tes
- HPI oder IfI?
- Erasmus o.ä.?
- DBS I bei mir?

Überblick

- Vorstellung der Arbeitsgruppe
- Organisatorisches
- Implementierung von Datenbanken
- Ausblick auf das Semester

Termine

- Vorlesung
  - Montags 9:15 - 10:45
    - in HS 3
  - Donnerstags 9:15 - 10:45
    - in HS 3
- Erste Vorlesung
  - 14.4.2008
- Letzte Vorlesung
  - 17.7.2008
- Feiertage
  - 1.5. Maifeiertag (Do)
  - 12.5. Pfingstmontag
- Sondertermine
  - 17.4. (Do): IBM Tag
  - 21.4. und 24.4.: Sascha

- Übungen
  - Ungefähr 1 SWS
  - Verteilt auf Doppelstunden im Semester
  - Bitte immer www beachten
  - Zwei Übungsgruppen
    - Montags 9:15 in HS 3
    - Dienstags 17:00 in HS 1
- Klausur
  - Eine Woche nach Semester
  - 1.5 vs. 2 vs. 4 vs. 8 Stunden

Feedback

- Folien
  - Vor der VL im WWW
    - ASAP
  - Datiert bei Updates
- Anregungen zur Verbesserung:
  - Gebrauch der Folien
  - Infos im WWW
  - …
- Fragen bitte jederzeit!
  - In der Vorlesung
  - Sprechstunde
    - Dienstags 15:00 – 16:00
    - Raum A-1.13
    - Am liebsten mit Anmeldung
  - Email: naumann@hpi.uni-potsdam.de

Literatur

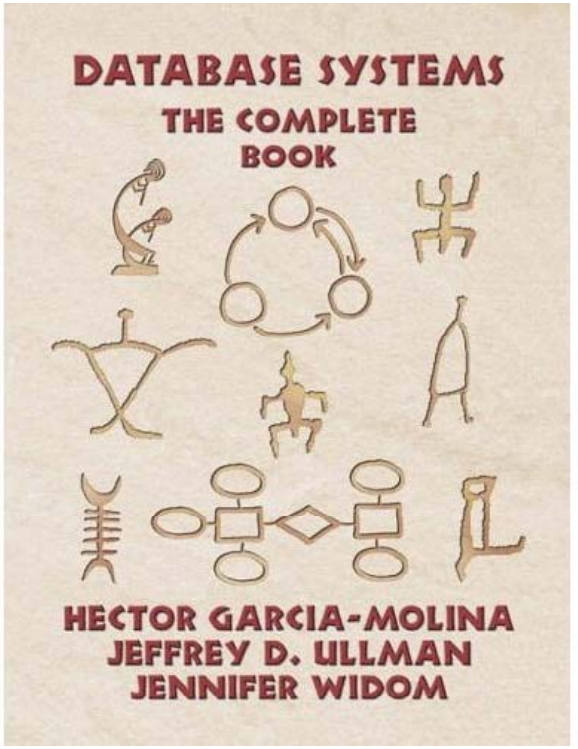
Database Systems - The Complete Book

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom: Pearson Education International, 2002.

Ebenfalls empfehlenswert

- Datenbanksysteme
  Alfons Kemper, André Eickler
  ISBN: 3486273922
- Datenbanken:
  Implementierungstechniken
  Gunter Saake, Andreas Heuer, Kai-Uwe Sattler

Und viele andere mehr…



Übungen und Leistungserfassung

- Übungen – Organisatorisches
  - Mehr von Sascha Szott bei erster Übung
  - Zweiergruppen
  - Vermutlich nur 5 Übungsblätter
  - Mind. 50% der Punkte auf jedem Blatt
    - Kein Freischuss
  - Tutor
    - Florian Reinhart
- Übungen – Inhaltliches
  - „Theoretische" Übungen
    - Auf Papier
  - Praktische Übungen
    - DB2 im Studentenpool
    - Neuer DB2 Server!

Überblick

- Vorstellung der Arbeitsgruppe
- Organisatorisches
- Implementierung von Datenbanken
- Ausblick auf das Semester

Das Megatron 2002 DBMS

- Naive Sicht
  - DBMS sollte recht einfach sein
- Megatron 2002
  - Unter UNIX verfügbar
  - Relationales Datenmodell
  - SQL Anfragen

Megatron - Implementierungsdetails

- Zuerst: NDA unterzeichnen!

- Relationen werden in ASCII Dateien gespeichert.
  - Relation Studenten(Name, Matrikel, Studiengang)
  - Wird zu Datei /usr/db/Studenten
- Eine Zeile pro Tupel
- Felder werden mit „#“ separiert

```
Smith # 123 # CS
Jones # 522 # EE
...
```

Megatron - Implementierungsdetails

- Schema wird in gesonderter ASCII Datei gespeichert
  - /usr/db/schema
- Eine Zeile pro Relation
- Zeile beginnt mit Relationenname
- Gefolgt von Attributname/Datentyp Paaren
  - Getrennt durch „#“

```
R1 # A # INT # B # STR …
R2 # C # STR # A # INT …
...
```

Eine Session mit Megatron

- Maschine: **dbhost**
- Megatron prompt: **&**

- **dbhost> MEGATRON3000**
  Welcome to MEGATRON 3000!
- **& ...**
  ...
- **& quit**
- **dbhost>**

Eine Session mit Megatron

- SQL Anfragen werden mit # beendet

- **`& SELECT * FROM Studenten #`**

Name	Matrikel	Fach
Smith	123	Informatik
Jones	522	Mathematik

- Speicherung von Anfrageergebnissen in einer Datei
- **`& SELECT * FROM Studenten WHERE Matrikel > 500 | HoheMtr #`**
- Erzeugt neue Datei /usr/db/HoheMtr mit einer Zeile

Anfrageausführung in Megatron

- **`& SELECT * FROM R WHERE <Bedingung>`**
  1. Schema-Datei lesen um Attribute und Datentypen von R zu bestimmen
  2. Prüfe, ob Bedingung semantisch korrekt ist.
  3. Zeige jedes Attribut als Zeilenkopf an
  4. Male eine Linie
  5. Lese die Datei namens R.
     Für jede Zeile
     1. Prüfe Bedingung
     2. Zeige Zeile als Tupel an, falls Bedingung erfüllt ist.

Anfrageausführung in Megatron

- **`& SELECT * FROM R WHERE <Bedingung> | T #`**
  1. Schema-Datei lesen um Attribute und Datentypen von R zu bestimmen
  2. Prüfe, ob Bedingung semantisch korrekt ist.
  3. Lesen der Datei namens R. Für jede Zeile
     1. Prüfe Bedingung
     2. Schreibe Zeile in Datei /usr/db/T, falls Bedingung erfüllt ist.
  4. Füge in Datei /usr/db/schema eine Zeile hinzu für Relation T

Anfrageausführung in Megatron

- **`& SELECT Büro FROM Studenten, Fachbereiche`**
  `WHERE Studenten.Name = „Smith“`
  `AND Studenten.Fach = Fachbereich.Name #`
  1. Schema-Datei lesen um Attribute und Datentypen von Studenten und Fachbereiche zu bestimmen.
  2. Prüfe, ob Bedingungen semantisch korrekt sind.
  3. Lesen von Datei /usr/db/Studenten. Für jede Zeile:
     1. Lesen von Datei /usr/db/Fachbereiche. Für jede Zeile:
        1. Erzeuge Join-Tupel
        2. Prüfe Bedingungen
        3. Zeige Tupel an, falls Bedingung erfüllt

Probleme mit Megatron 2002

Probleme mit Megatron 2002

- Tupel-Layout auf der Festplatte
  - Falls sich ein Wert ändert muss gesamte Datei neu geschrieben werden.
    - Alle Zeichen bewegen sich entsprechend in der Datei.
  - ASCII Speicherung ist teuer
  - Löschung ist ebenfalls teuer
- Teure Suche
  - Es kann nicht schnell, z.B. basierend auf einem Schlüssel, ein bestimmtes Tupel gefunden werden
  - Es wird immer gesamte Relation gelesen, selbst bei strengen Nebenbedingungen

Probleme mit Megatron 2002

- Anfrageausfühung ist brute force
  - Joinoperationen können wesentlich schneller implementiert werden.
  - Selektionen sollten früher ausgeführt werden
- Kein Puffermanagement
  - Hauptspeicher wird nicht genutzt
  - Sämtliche Daten werden stets von Festplatte gelesen
- Keine Nebenläufigkeitskontrolle
  - Mehrere Nutzer könnten gleichzeitig Datei ändern
- Keine Zuverlässigkeit
  - Daten können bei Systemabsturz verloren gehen
  - Operationen könnten nur teilweise ausgeführt werden

Probleme mit Megatron 2002

- Keine Sicherheit
  - Dateisystem ist unsicher
  - Granularität ist grob
- Wenig Metadatenfähigkeiten
  - Statistiken, Nebenbedingungen
- Kein API
  - Wie können anderen Anwendungen auf die Daten zugreifen?
- Keine Interaktion mit anderen DBMS
- Keine GUI

Überblick

- Vorstellung der Arbeitsgruppe
- Organisatorisches
- Implementierung von Datenbanken
- Ausblick auf das Semester

Kurzüberblick

- Physische Speicherstrukturen (2)
- Physische Repräsentation von Daten (1)
- Indexstrukturen (5)
- Anfrageausführung (2)
- Optimierung (4)
- Benchmarking (1)
- Recovery (2)
- Transaktionsmanagement (3)

Effizienter Umgang mit großen Datenmengen

Teil 1: Umgang mit großen Datenmengen

Teil 2: Effiziente Manipulation der Daten

Physische Speicherstrukturen

- Speicherhierarchie
  - Cache, RAM, Sekundärspeicher, Tertiärspeicher
- Festplatten
  - Mechanik, Controller, Charakteristiken, Read/Write
- Algorithmen für Sekundärspeicher
  - Bsp: Sortierung mit „Two-Phase, Multiway Merge-Sort“
- Zugriffsbeschleunigung
  - Clustering, Multiple Disks, Spiegelung, Prefetching
- Diskfehler
  - Checksums, Stable Storage
- Disk Recovery
  - RAID: Spiegelung, Parity Blocks, …

Physische Repräsentation von Daten

- Datenelemente und Felder
  - Relationale Datenbankelemente, Datentypen, Datensätze
- Adressierung
  - Client/Server, Physische vs. Logische Adresse
- Datentypen variabler Länge
  - Variable Felder, Variable Datensätze, Nullwerte, BLOBs
- Schreiboperationen
  - Insert, Update, Delete

Indexstrukturen

- Indizes auf sequentiellen Dateien
  - Sortiert, dicht-besetzter und dünn-besetzter Index, Mehrstufiger Index, Updates
- Sekundärindizes
  - Nicht sortiert, Aufbau, Anwendungen, Indirektion, Invertierte Indizes (IR)
- B-Bäume
  - Aufbau, Anwendungen, Bereichsanfragen, Insert, Delete
- Hashverfahren (für Sekundärspeicher)
  - Insert, Delete, erweiterbares Hashen, Lineares hashen
- Eventuell: Multidimensionale Indizes
  - GIS, DWH
- Eventuell: Bitmap Indizes

Anfrageausführung

- Physische Operatoren
  - Scan, Sortierung, I/O Kosten, Iteratoren
- One-Pass Operationen
  - Unär: Selektion, Projektion, Duplikateliminierung, Gruppierung
  - Binär: Mengenoperationen (∪∩—×), Join
  - Nested-Loop Join
- Two-Pass Operationen (mittels Sortierung)
  - Duplikateliminierung, Gruppierung, Mengenoperationen (∪∩—×), Merge-Sort Join
- Two-Pass Operationen (mittels Hashing)
  - …, Hash-Join, …
- Index-basierte Algorithmen
  - Clustering Indizes, Selektion, Joins
- Puffermanagement
  - Architektur, Strategien (LRU, FIFO, …)
- Parallele Algorithmen

Optimierung

- Teils schon in DBS I:
  - Parsing, Algebraische Transformationsregeln
- Kostenmodell
  - Projektion, Selektion, Verteilungen, Join
- Kostenbasierte Optimierung
  - Histogramme, Statistiken, Heuristiken, Enumerierung aller Pläne
- Joinreihenfolge
  - Links vs. rechts, Dynamische Programmierung, greedy Algorithmus
- Physische Anfragepläne
  - Selektionsmethode, Join-Methode, Pipelining

Benchmarking

- Leistungsbewertung
  - Komponenten eines Benchmarks, Ziele
- TPC Benchmarks
  - TPC A, B
  - TPC C: OLTP
  - TPC D/H: Decision Support DWH
  - TPC W: e-Commerce

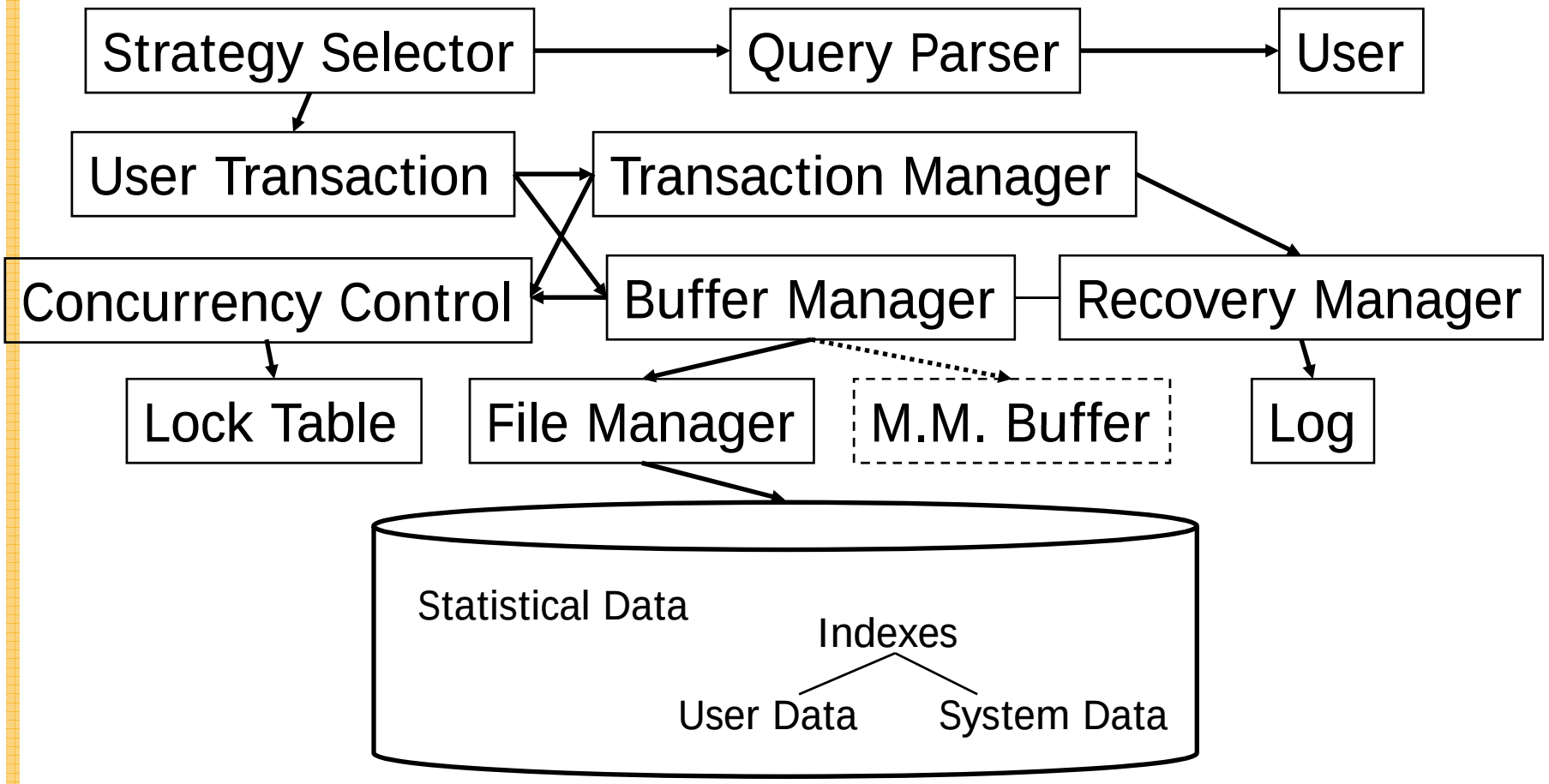
Recovery

- Fehlerarten
  - Medienfehler, Systemfehler, Transaktionen en detail
- Undo Logging
  - Log-Datensätze, Log-Regeln, Wiederherstellung mit Undo, Checkpoints
- Redo Logging
  - Log-Regeln, Wiederherstellung mit Redo, Checkpoints
- Undo/Redo Logging
  - Best of both worlds, Regeln, Wiederherstellung, Checkpoints
- Archivierung
  - Archive, Nebenläufige Archivierung, Wiederherstellung mit Archiven und Logs

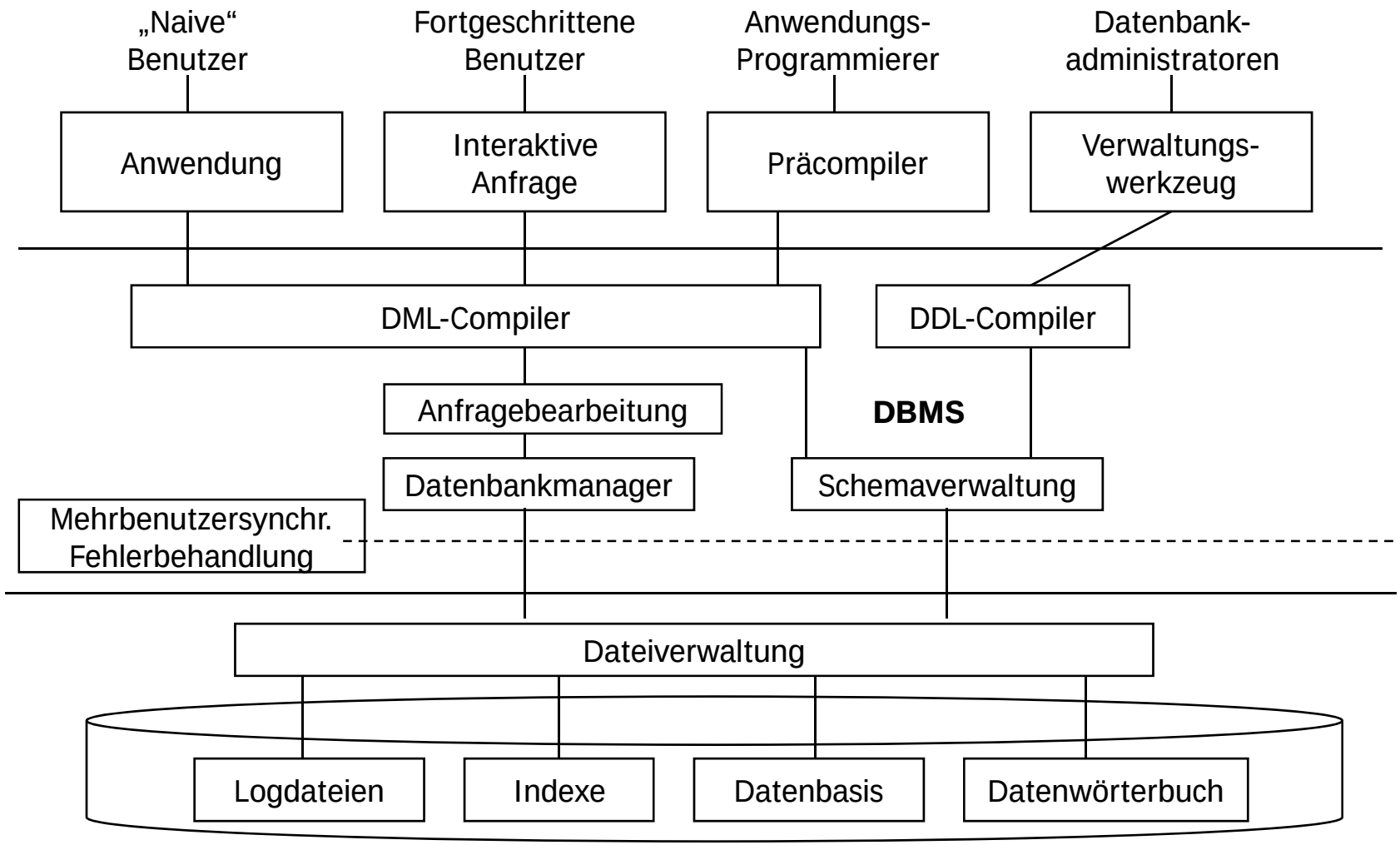
Transaktionsmanagement

- Fortsetzung aus DBS I:
  - Transaktionen, Schedules, 2PL, Sperren
- Mehrere Sperr-Modi
  - Shared und exclusive Sperren, Aufwertung von Sperren, Update-Sperren, Inkrement-Sperren
- Scheduler
  - Architektur, Sperrtabelle
- Hierarchisches Sperren
  - Sperrgranulate (Tabellen, Blöcke, Tupel), warnende Sperren
- Baumprotokolle
  - Motivation (z.B. B-Bäume), Regeln, Beweis
- Nichtsperrende Verfahren
  - Zeitmarken (Timestamps), Validierung, Optimistische Verfahren

Systemarchitektur nach Garcia-Molina



Systemarchitektur nach Leser



Feedback

Fragen bitte jederzeit!

- In der VL
- Sprechstunde: Dienstags 15-16 Uhr
- Email: naumann@hpi.uni-potsdam.de
- Telefon: (0331) 5509 280

Anregungen zur Verbesserung:

- Z.B. zu
  - Gebrauch der Folien
  - Infos im WWW