Wolfgang Lehner

Datenbanken Systemarchitektur I
Wintersemester 2013/2014

1 Einführung

Motivation - Was? Warum?

New Realities

- TB disks < $100
- Everything is data
- Rise of data-driven culture
  - CERN's LHC generates 15 PB a year
  - Sloan Digital Sky Survey (200 GB/night)

The quest for knowledge used to begin with grand theories. Now it begins with massive amounts of data.

Welcome to the Petabyte Age.

The Web is a huge source of information: search engines (Google, Yahoo!) collect and store billions of documents

- 20 PB processed every day at Google (2008)
- eBay has 6.5 PB of user data + 50 TB/day (2009)

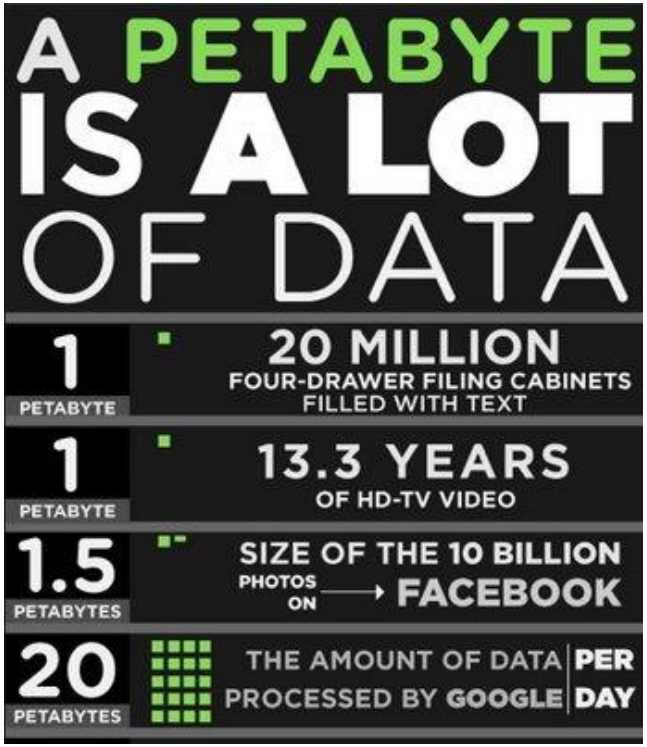
- Structured data, text, images, video
  - 35 hr of video uploaded to YouTube every min

- World of Warcraft utilizes 1.3 PB of storage space
- Valve Steam delivers 20 PB of content monthly

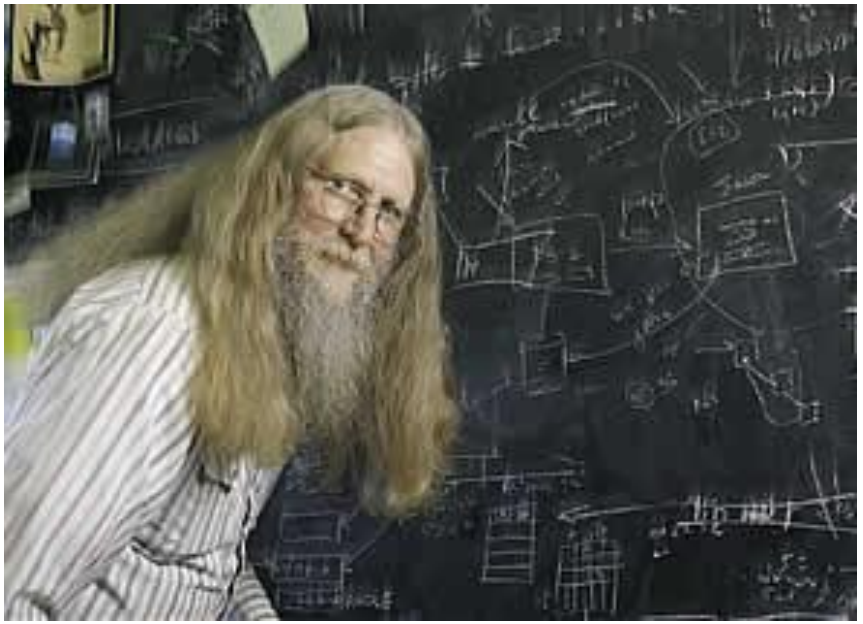
- Data is partitioned, computation is distributed
  - Reading 20PB would take 12 years at 50MB/s

Other data produces

- Call Centers (forms, Voice, Text, Video, Images)
  - 4.6 * 10^9 mobile phones



Relational databases are the foundation of western civilization



Bruce Lindsay,
IBM Fellow @ IBM Almaden Research Center

Konzepte, Methoden, Werkzeuge und Systeme für die

- dauerhafte — Lebensdauer Daten > Dauer Erzeugungsprozess
- zuverlässige — Integrität, Konsistenz, Verlustsicherheit
- Unabhängige — wechselseitige Änderungsimmunität AWP <-> DBS

Verwaltung und

- komfortable — “höhere” abstrakte Schnittstelle
- flexible — Ad-hoc-Zugriffsmöglichkeit

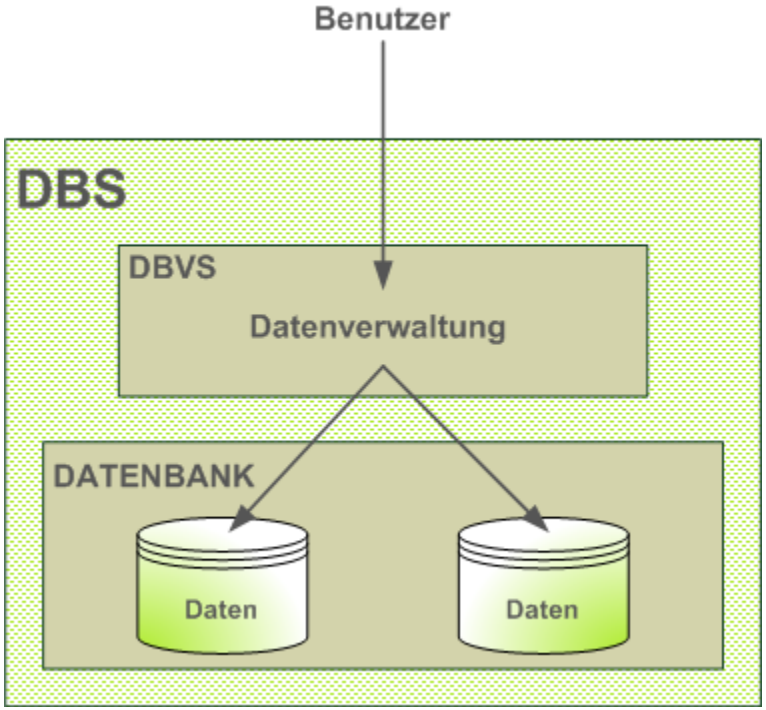
Benutzung von

- großen — Datenvolumen >> Hauptspeicher
- integrierten — kontrollierte Redundanz von/für mehrere Anwendungen
- mehrfach benutzbaren — paralleler Zugriff

Datenbasen

Begriffseinführung: "Datenbanksystem"

- ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen, die von mehreren Anwendungsprogrammen benutzt werden



Komponenten eines Datenbankverwaltungssystems

- Datenbank, in der die Daten abgelegt werden
- Datenbanksoftware, die die Daten entsprechend den vorgegebenen Beschreibungen abspeichern, auffinden oder weitere Operationen mit den Daten durchführen (entnommen aus Informatik-Duden)

Leistungen in den folgenden Bereichen

- Datenmodell und Datendefinition
- Datenzugriff und -manipulation
- Steuerung und Überwachung

Kommerzielle Systeme

DB2 Universal database

IBM Informix

ORACLE

Microsoft SQL Server 2008

SYBASE

tamino

Frei verfügbare Systeme

Firebird THE POWER WITHIN

MySQL

Drizzle

cloudscape

PostgreSQL

XMLDB

Herausforderungen

Motivation

- Behauptung
  um systemtechnische Vorgänge verstehen zu können, ist deren Notwendigkeit aus Sicht der Anwendung aufzuzeigen
- Beschränkung auf drei Bereiche
  - Relationenmodell
  - Anfragesprache SQL
  - Transaktionen

Ziel eines Datenbanksystems

- Repräsentation von Informationen der Anwendungswelt (“Miniwelt”)
- Abbildung der realen Welt auf das Datenbanksystem (strukturell)
- ‘Nachziehen’ von Veränderungen in der realen Welt (operationell)

Was macht Datenbank- und TA-Verarbeitung so schwer?

- Antwort: die “-ities” ....
  - Reliability — system should rarely fail
  - Availability — system must be up all the time
  - Response time — within 1-2 seconds
  - Throughput — thousands of transactions/second
  - Scalability — start small, ramp up to Internet-scale
  - Security — for confidentiality and high finance
  - Configurability — for above requirements + low cost
  - Atomicity — no partial results
  - Durability — a transaction is a legal contract
  - Distribution — of users and data

Was macht Datenbank- und TA-Verarbeitung so wichtig?

- Kern des eCommerce
- Vielzahl mittlerer und großer Betriebe sind darauf angewiesen
- Großer Markt (Lowell Report)

Typen von Datenbanksprachen

- Prozedurale Datenbanksprachen
  - Tupel- oder satzorientiert
  - Programmierer denkt in Satzfolgen
  - Navigation über Zugriffspfade durch die vorhandenen Daten
    findNext()
    findFirst()

- Deskriptive Datenbanksprachen
  - Mengenorientiert (typisch für Relationenmodell)
  - Programmierer denkt in Mengen von Sätzen mit bestimmten Eigenschaften
  - Zugriff erfolgt durch inhaltliche Kriterien
    (... alle Sätze mit der Eigenschaft ...)

ZENTRAL

- In proz. DB Sprachen wird spezifiziert, WIE d
- In deskr. DB-Sprachen wird spezifiziert, WAS das DBS zu suchen hat

DDL (Data Definition Language)

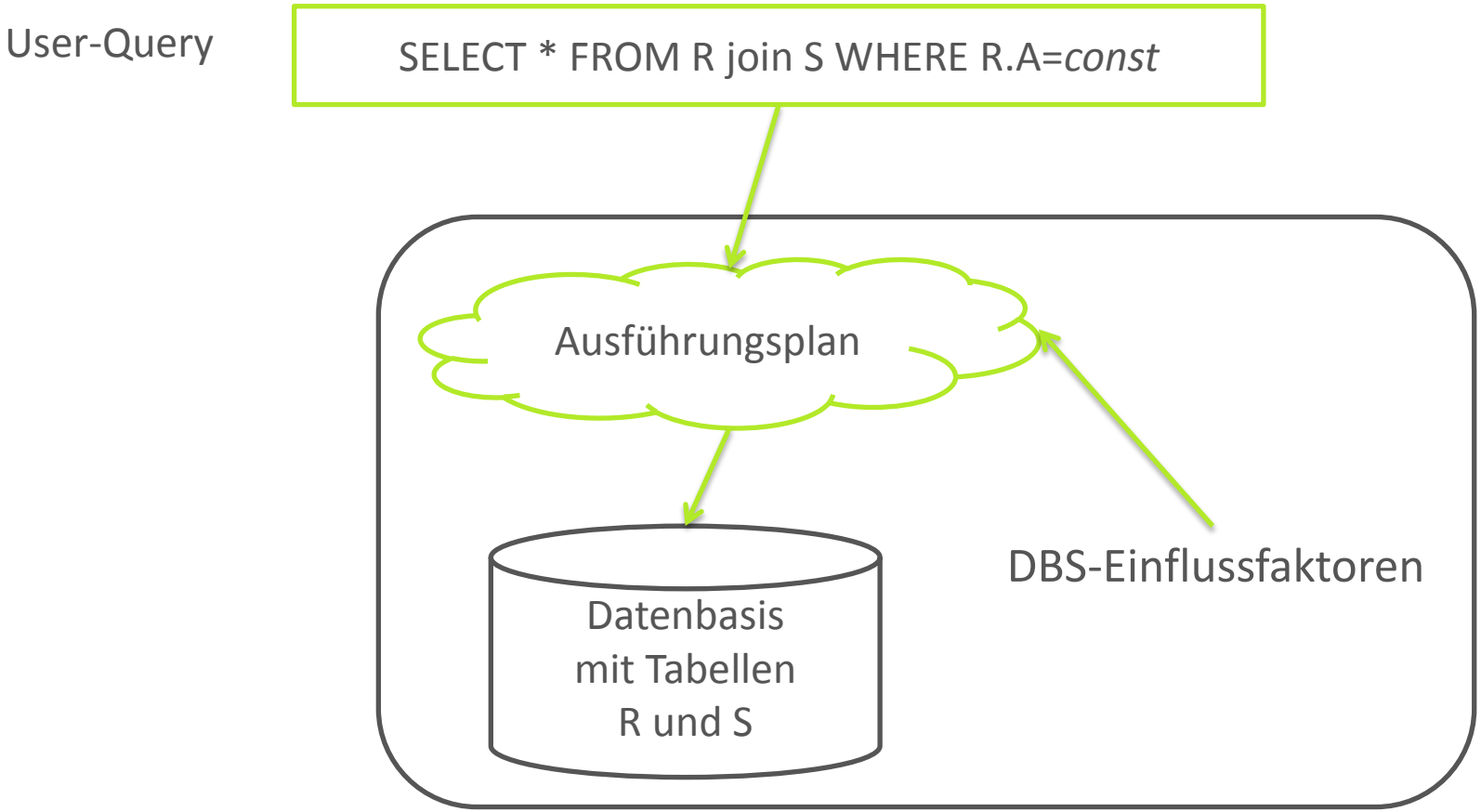
```
CREATE TABLE Buch (
   ISBN CHAR(10),
   Titel VARCHAR(200),
   Verlagsname VARCHAR(30),
   PRIMARY KEY (ISBN),
   FOREIGN KEY (Verlagsname) REFERENCES Verlage (Verlagsname))
```

DML (Data Manipulation Language)

- Anfragen und Änderungsoperationen

```
SELECT Buch.InventarNr, Titel, Name
   FROM Buch, Ausleih
   WHERE Name = 'Meyer' AND Buch.InventarNr = Ausleih.InventarNr
UPDATE Angestellte
   SET Gehalt = Gehalt + 1000
   WHERE Gehalt < 5000
INSERT INTO Buch VALUES(4867,'XQuery', '3-876','Lehner')
INSERT INTO Kunde
   (SELECT LName, LAdr, 0 FROM Lieferant)
```

Herausforderung (Ausführungsplan)



Äquivalenz von Algebra-Termen

- Beispiel
  - $\sigma_{A=const}$ (REL_1 join REL_2) und A aus REL_1
  - $\sigma_{A=const}$ (REL_1) join REL_2
- allgemeine Strategie: Selektionen möglichst früh, da sie Tupelzahl verkleinern
- Beispiel: REL_1 100 Tupel, REL_2 50 Tupel (Tupel sequenziell abgelegt)
  - 5000 (join) + 5000 (σ) = 10000 Operationen
  - 100 (σ) + 10 · 50 (join) = 600 Operationen
    falls 10 Tupel in REL_1 die Bedingung A=const erfüllen

Join-Verarbeitung

- (Sort-)Merge-Join: Verbund durch Mischen von R1 und R2
  - effizient, wenn eine oder beide Relation(en) sortiert nach den Verbund-Attributen vorliegen
- Nested-Loop-Joins: Verbund durch doppelte Schleife

Logische Optimierung

- nutzt nur algebraische Eigenschaften der Operatoren -> algebraische Optimierung
- keine Kenntnis über physischen Datenbestand
- Beispiel
  - Entfernung redundanter Operationen
  - Verschieben von Operationen derart, dass Selektionen möglichst früh ausgeführt werden

Physische Optimierung

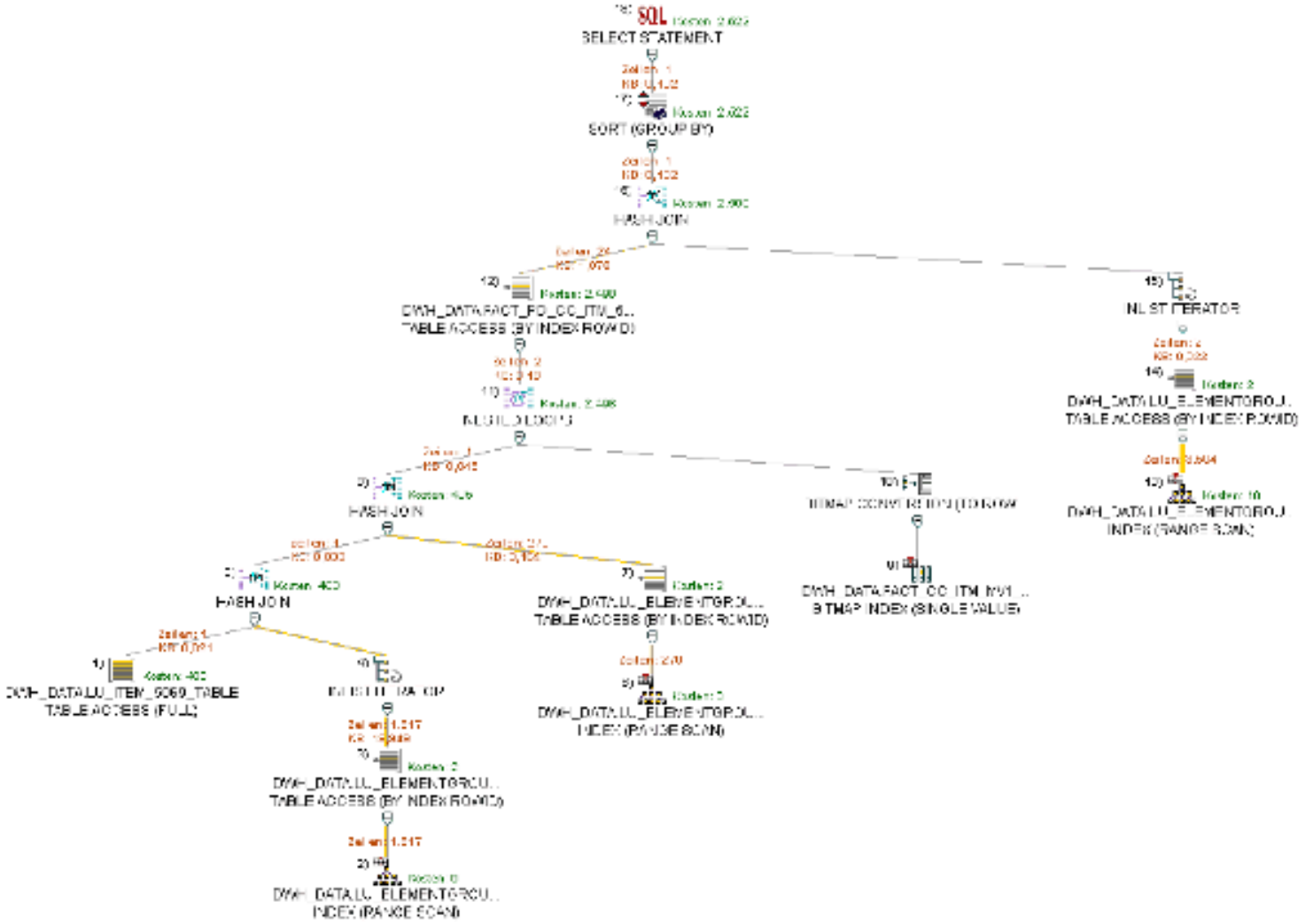
- Nutzung von Informationen über die vorhandenen Speicherungsstrukturen
- Auswahl der Implementierungsstrategie einzelner Operationen (Merge Join vs. Nested-Loops-Join)
- Beispiel
  - Verbundreihenfolge anhand der Größe und Unterstützung der Relationen durch Zugriffspfade
  - Reihenfolge von Selektionen nach der Selektivität von Attributen und dem Vorhandensein von Zugriffspfaden

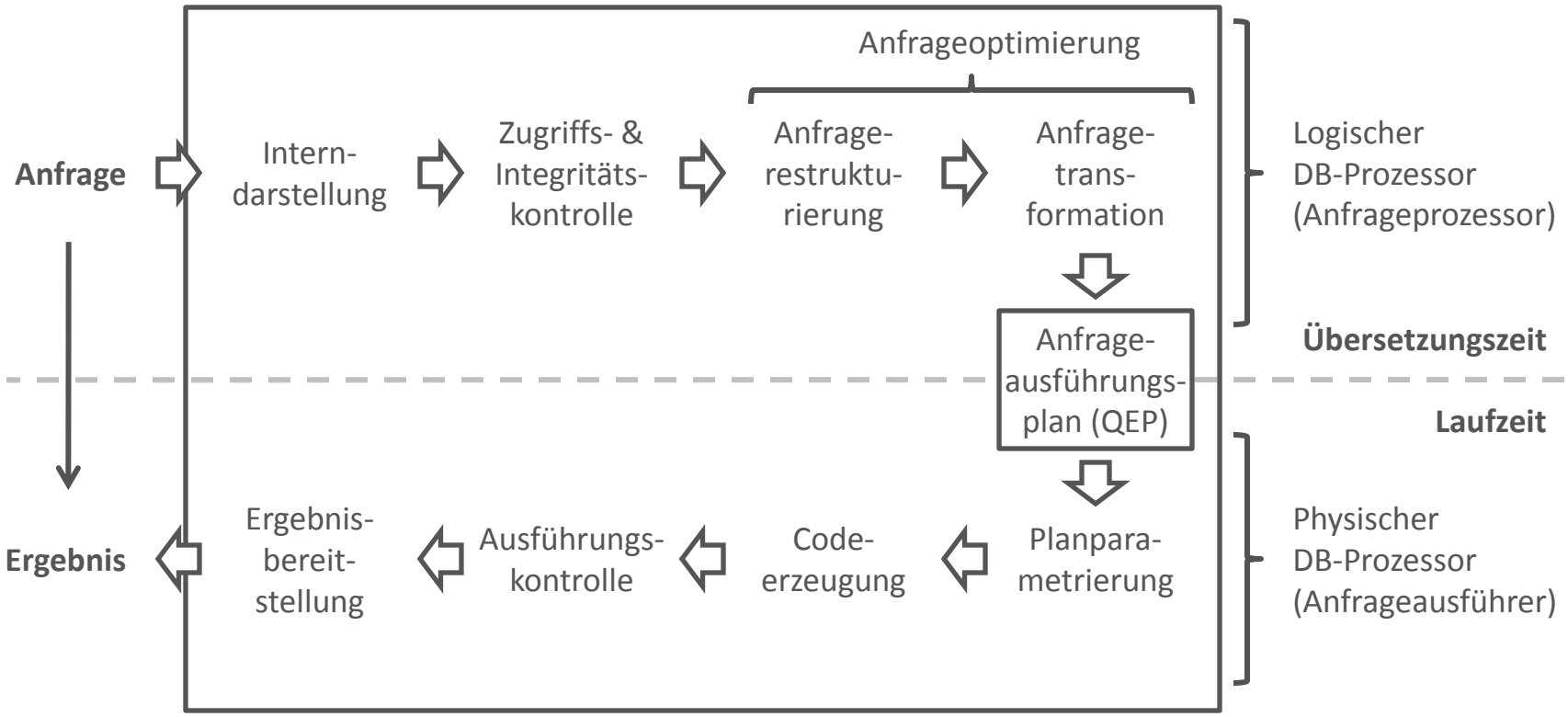
```
SELECT /*+ ORDERED INDEX(t1) USE_HASH(t1) USE_HASH(t2) NO_INDEX(t2,
lu_item_pk) USE_HASH(t3) USE_HASH(t4) INDEX(t5) NO_INDEX(t5,
lu_outlet_5069_pk) USE_HASH(t5) USE_HASH(t6) */
   'G' || t3.elementgroup_id pg_featurevalue_07_id,
   'G' || t4.elementgroup_id pg_featurevalue_17_id,
   'B' || t5.ch_featurevalue_17_id ch_featurevalue_17_id,
   'B' || t5.country_id country_id,
   'B' || t2.productgroup_id productgroup_id,
   'G' || t6.elementgroup_id period_id,
   SUM(t1.pd_sales_units*t1.pd_projection_factor*t1.pd_price_units_eur)
salesvaluereteur,
SUM(t1.pd_sales_units*t1.pd_projection_factor*t1.pd_price_units_eur)/1000 thsalesvaluereteur,
   SUM(t1.pd_sales_units*t1.pd_projection_factor) salesunitsret,
   SUM(t1.pd_sales_units*t1.pd_projection_factor)/1000 thsalesunitsret,
SUM((t1.pd_sales_units*t1.pd_projection_factor*t1.pd_unit_factor)/1000)/1000
thssalespiecelitrekilosret,
   NVL(SUM
(t1.pd_sales_units*t1.pd_projection_factor*t1.pd_price_units_eur)/LTRIM(SUM (t1.pd_sales_units*t1.pd_projection_factor),0),0) priceeurret
FROM
   lu_item_5069 t2,
   lu_elementgroup_rel t3,
   lu_elementgroup_rel t4,
   fact_pd_out_itm_5069 t1,
   lu_elementgroup_rel t6,
  lu_outlet_5069 t5
WHERE
/* Attribute Joins */
    ((t1.item_id = t2.item_id
/* Customizing Begin */
 AND   t1.productgroup_id = t2.productgroup_id)
/* Customizing End */
 AND  (t2.pg_featurevalue_07_id = t3.value_id)
 AND  (t2.pg_featurevalue_17_id = t4.value_id)
 AND  (t1.outlet_id = t5.outlet_id
```

> Beispiel: Datenbankanfrage (2)

```
/* Customizing Begin */ AND   t1.period_id = t5.period_id
  AND   t1.project_id = t5.project_id)
/* Customizing End */
  AND  (t1.period_id = t6.value_id)
     )
/* Attribute Filters */
  AND ((t2.productgroup_id = 15638)
  AND  (t1.productgroup_id = 15638) /* Push Down Filters */
  AND  (t5.country_id = 15)
  AND  (t1.country_id = 15) /* Push Down Filters */
  AND  (t2.sector_id IN (1,10,5))
  AND  (t5.ch_featurevalue_17_id = 7740)
  AND  (t3.elementgroup_id IN (14386,14387,4941,4945,6789,6790))
  AND  (t4.elementgroup_id = 5015)
  AND  (t6.elementgroup_id IN
(34542,34697,34809,34810,34811,34812,34813,34815,34816,34817,34820,34881,39080))
  AND  (t1.period_id IN
(20020299999030,20020399999030,20020399999060,20020499999030,20020599999030,20020599999060,20020599999120,20020699999030,2
     0020799999030,20020799999060,20020899999030,20020999999030,20020999999060,20020999999120,20021099999030,20021199999030,
     20021199999060,20021299999030,20030199999030,20030199999060,20030199999120,20030299999030,20030399999030,2003039999906
     0,20030499999030,20030599999030,20030599999060,20030599999120,20030699999030,20030799999030,20030799999060,200308999990
     30,20030999999030,20030999999060, 20030999999120,20031099999030,20031199999030,20031199999060,20031299999030,
     20040199999030,20040199999060,20040199999120,20040299999030))
/* Resolved ElementGroup Filters */
     )
/* Fact Filters */
  AND (t1.project_type_id IN ('2','3')   AND  t1.project_type_id IN ('2','3')   AND  t1.project_type_id IN ('2','3')
  AND  t1.project_type_id IN ('2','3')   AND  t1.project_type_id IN ('2','3')   AND  t1.project_type_id IN ('2','3')
     )
GROUP BY
   t3.elementgroup_id,    t4.elementgroup_id,    t5.ch_featurevalue_17_id,    t5.country_id,    t2.productgroup_id,    t6.elementgroup_id
```

> Beispiel eines Anfragegraphens



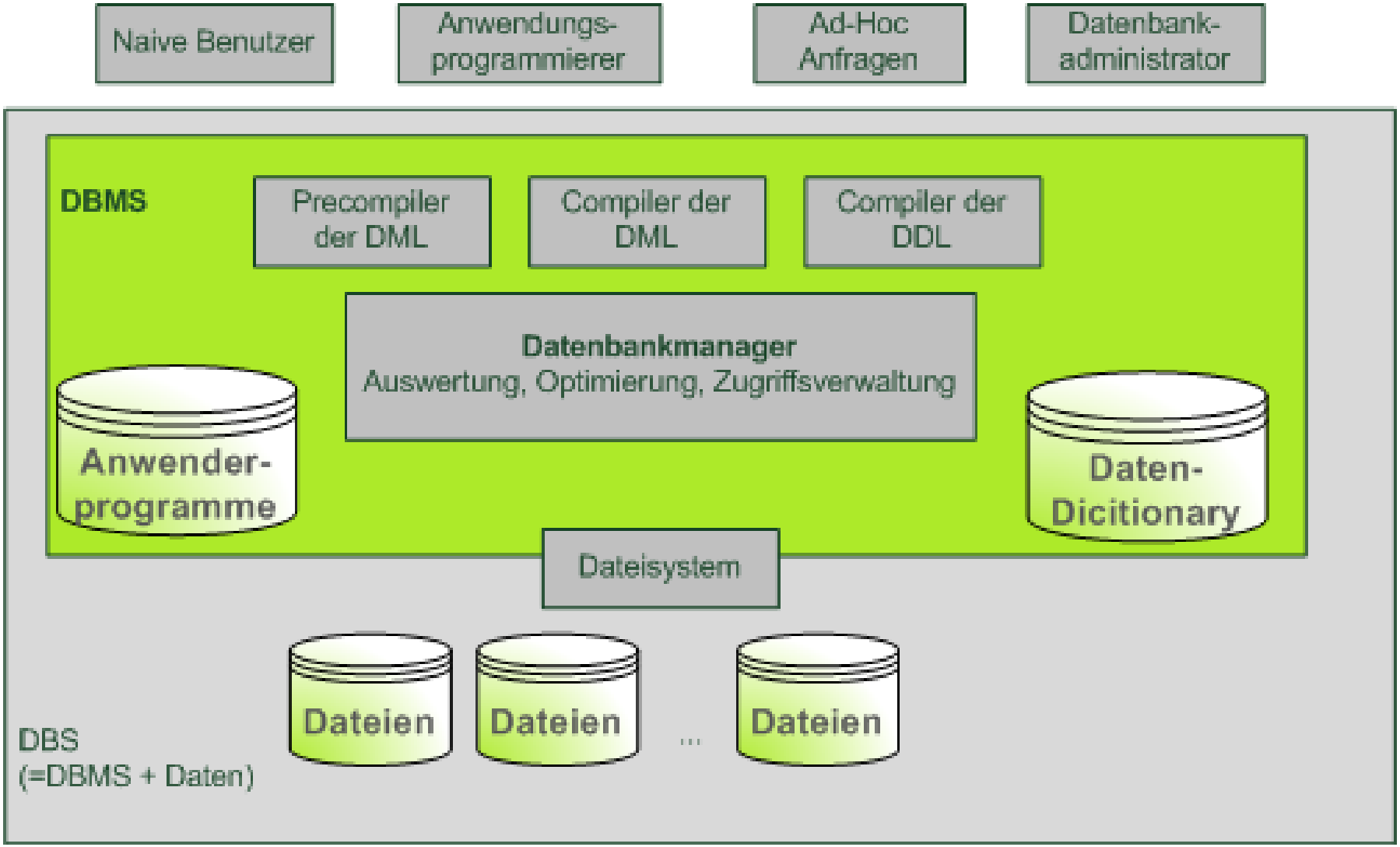

Anfrageoptimierung
Anfrage
Intern-darstellung
Zugriffs- & Integritäts-kontrolle
Anfrage-restruktu-rierung
Anfrage-trans-formation
Logischer DB-Prozessor (Anfrageprozessor)
Anfrage-ausführungs-plan (QEP)
Übersetzungszeit
Laufzeit
Ergebnis
Ergebnis-bereit-stellung
Ausführungs-kontrolle
Code-erzeugung
Planpara-metrierung
Physischer DB-Prozessor (Anfrageausführer)

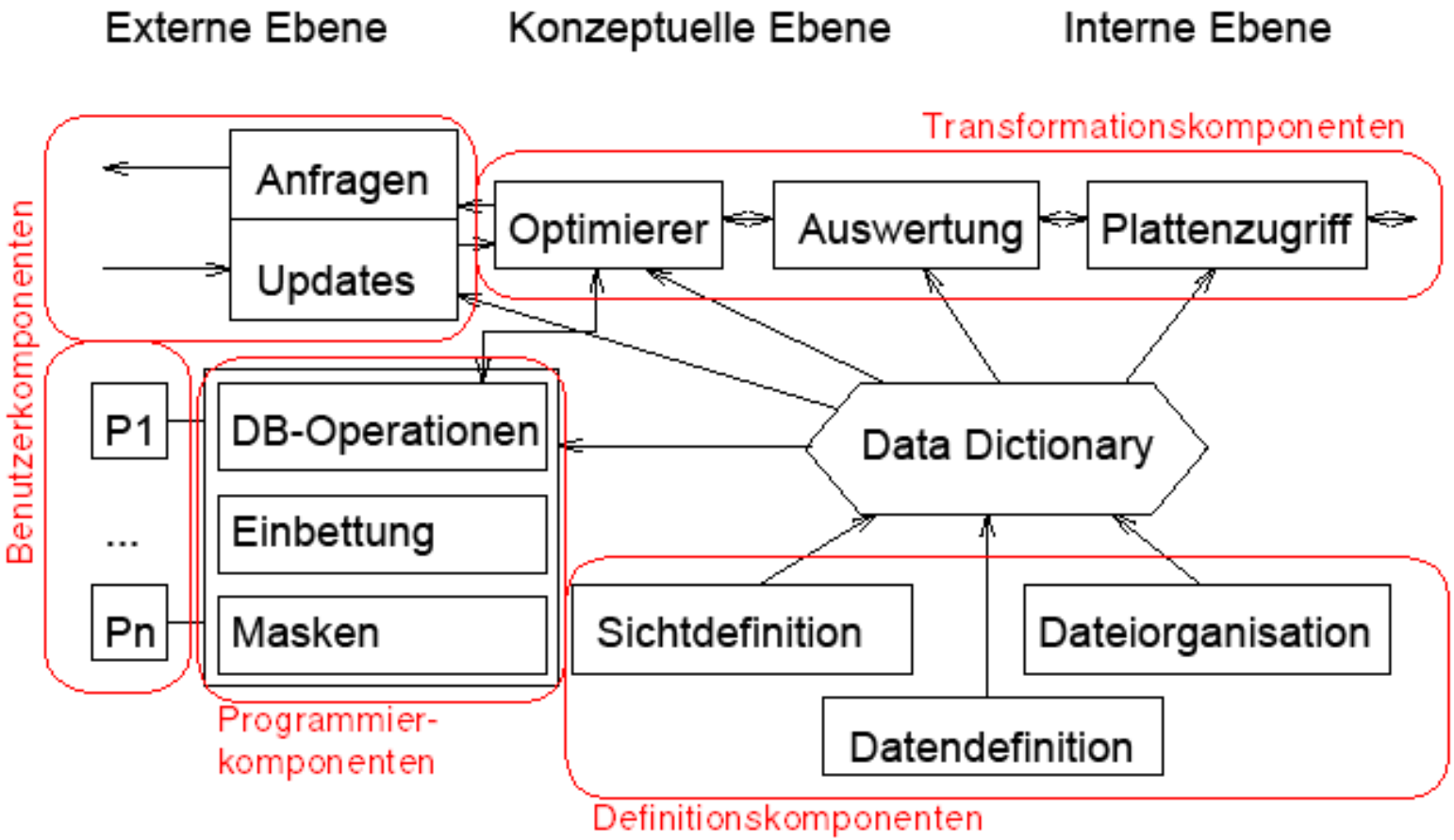
Charakterisierung

- eine Datenbanktransaktion stellt eine logische Arbeitseinheit dar
- Zusammenfassen von aufeinanderfolgenden DB-Operationen, die eine Datenbank von einem konsistenten Zustand in einen neuen konsistenten Zustand überführen (physische und logische Konsistenz)
- DB-Operationen sind gleichzusetzen mit SQL-Befehlen
- Innerhalb einer Transaktion können logisch inkonsistente Zustände auftreten

Der Transaktionsmanager der DB garantiert ACID-Eigenschaften

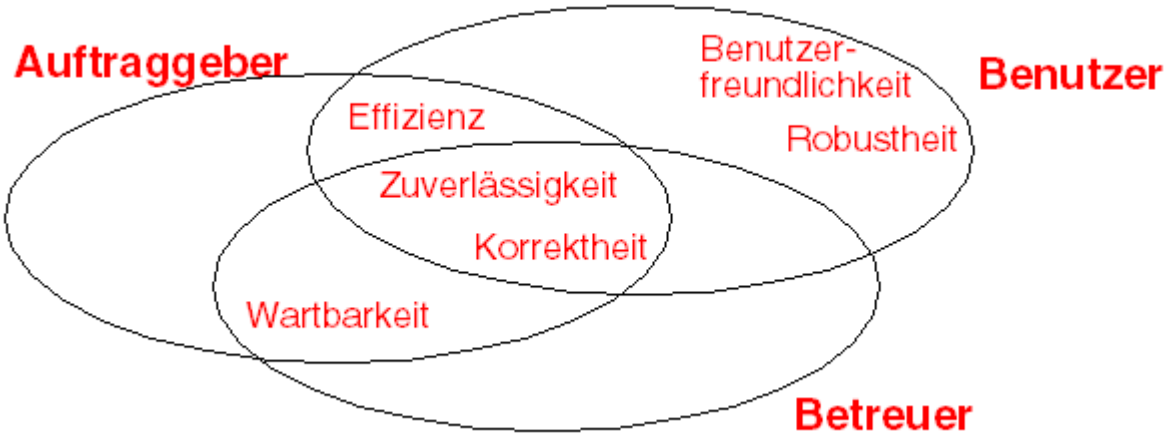
- entweder vollständige Ausführung einer Transaktion ...
- oder Wirkungslosigkeit der gesamten Transaktion (und damit aller beteiligten Operationen)


Naive Benutzer
Anwendungs-programmierer
Ad-Hoc Anfragen
Datenbank-administrator
DBMS
Precompiler der DML
Compiler der DML
Compiler der DDL
Datenbankmanager
Auswertung, Optimierung, Zugriffsverwaltung
Anwender-programme
Daten-Dicitionary
Dateisystem
Dateien
Dateien
...
Dateien
DBS
(=DBMS + Daten)


Externe Ebene
Konzeptuelle Ebene
Interne Ebene
Transformationskomponenten
Benutzerkomponenten
Anfragen
Updates
Optimierer
Auswertung
Plattenzugriff
P1
...
Pn
DB-Operationen
Einbettung
Masken
Data Dictionary
Sichtdefinition
Dateiorganisation
Datendefinition
Programmier-komponenten
Definitionskomponenten

Schichtenmodell

Innere und äußere Qualitätskriterien



Bekannte Konzepte der Implementierung

- Geheimnisprinzip (Information Hiding)
- Hierarchische Strukturierung durch Schichten
- eine Schicht realisiert einen bestimmten Dienst, den sie an der Schnittstelle “nach oben” höheren Schichten zur Verfügung stellt
- eine Schicht nimmt Dienste unterliegender Schichten in Anspruch

Beobachtung

- Schichten entstehen sukzessive durch Strukturierung oder Erweiterung des Anwendungsprogrammes
- Schichtenbildung ist eine Möglichkeit, um die bei der Software-Entwicklung geforderten Ziele (Wartbarkeit, Erweiterbarkeit, etc.) zu erreichen

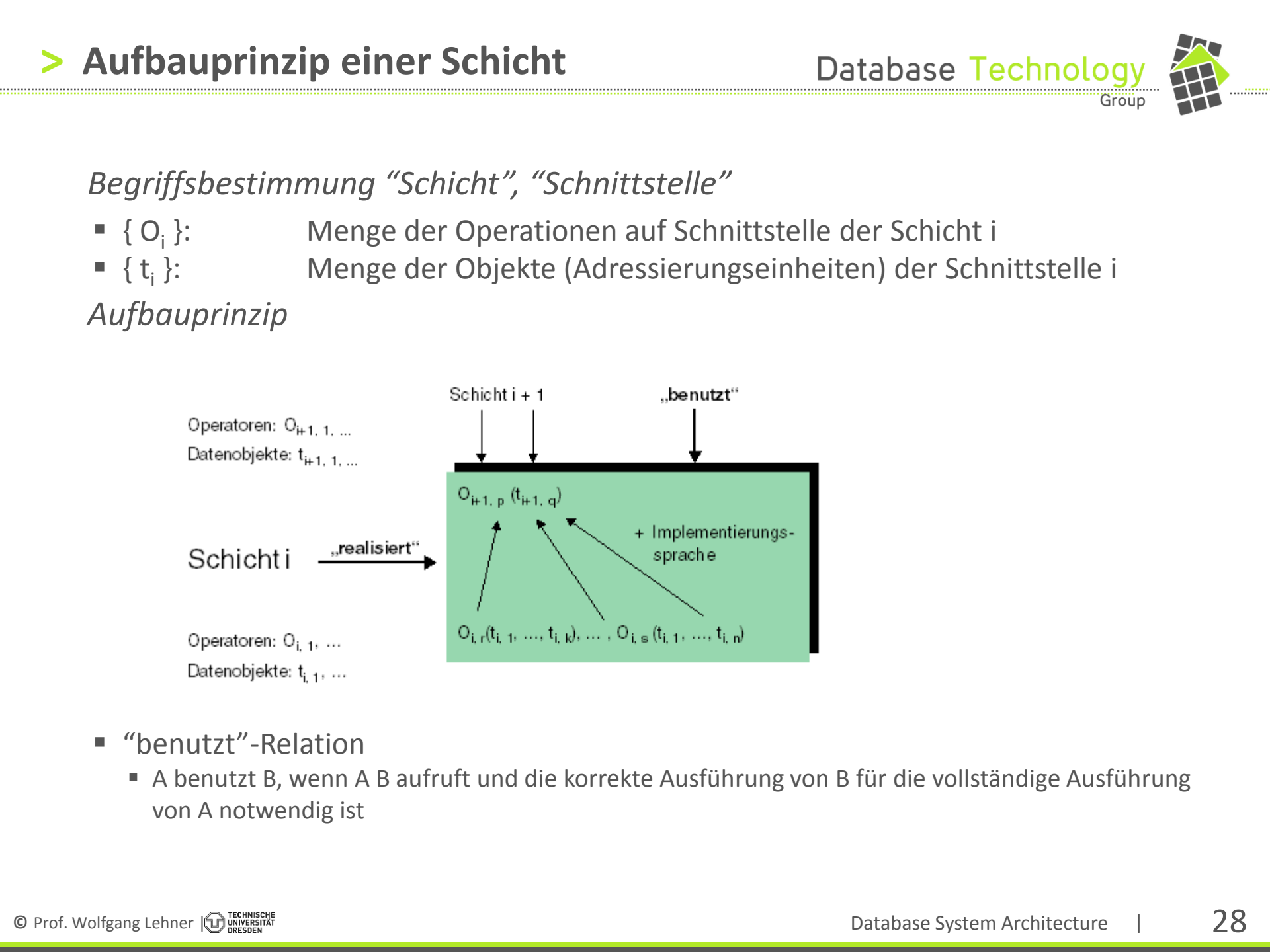
“Veredelungshierarchie”

- entlang der Schichten wird nach oben hin “abstrahiert”
- “Eine Hauptaufgabe der Informatik ist systematische Abstraktion”

H. Wedekind

Problem: Anzahl der Schichten in einem Softwaresystem

- n = 1: monolithisches System -> keine Vorteile der Schichtenbildung
- n sehr groß: -> hoher Koordinierungsaufwand
- Daumenregel: n typischerweise zwischen 3 und 10



Begriffsbestimmung "Schicht", "Schnittstelle"

- { O_i }: Menge der Operationen auf Schnittstelle der Schicht i
- { t_i }: Menge der Objekte (Adressierungseinheiten) der Schnittstelle i

Aufbauprinzip

- "benutzt"-Relation
  - A benutzt B, wenn A B aufruft und die korrekte Ausführung von B für die vollständige Ausführung von A notwendig ist

Vorteile als Konsequenzen der Nutzung hierarchischer Strukturen

- Höhere Ebenen (Systemkomponenten) werden einfacher, weil sie tiefere Ebenen (Systemkomponenten) benutzen können.
- Änderungen auf höheren Ebenen haben keinen Einfluss auf tiefere Ebenen.
- Höhere Ebenen können abgetrennt werden, tiefere Ebenen bleiben trotzdem funktionsfähig.
- Tiefere Ebenen können getestet werden, bevor die höheren Ebenen lauffähig sind.

Jede Hierarchieebene kann als abstrakte oder virtuelle Maschine aufgefasst werden

- Programme der Schicht i benutzen als abstrakte Maschine die Programme der Schicht i-1, die als Basismaschine dienen.
- Abstrakte Maschine der Schicht i dient wiederum als Basismaschine für die Implementierung der abstrakten Maschine der Schicht i+1

Eine abstrakte Maschine entsteht aus der Basismaschine durch Abstraktion

- Einige Eigenschaften der Basismaschine werden verborgen.
- Zusätzliche Fähigkeiten werden durch Implementierung höherer Operationen für die abstrakte Maschine bereitgestellt.

Schichtenbildung

- Modularisierung, Übersichtlichkeit, ...
- Erstellung generischer Softwareeinheiten

Generische Softwaremodule

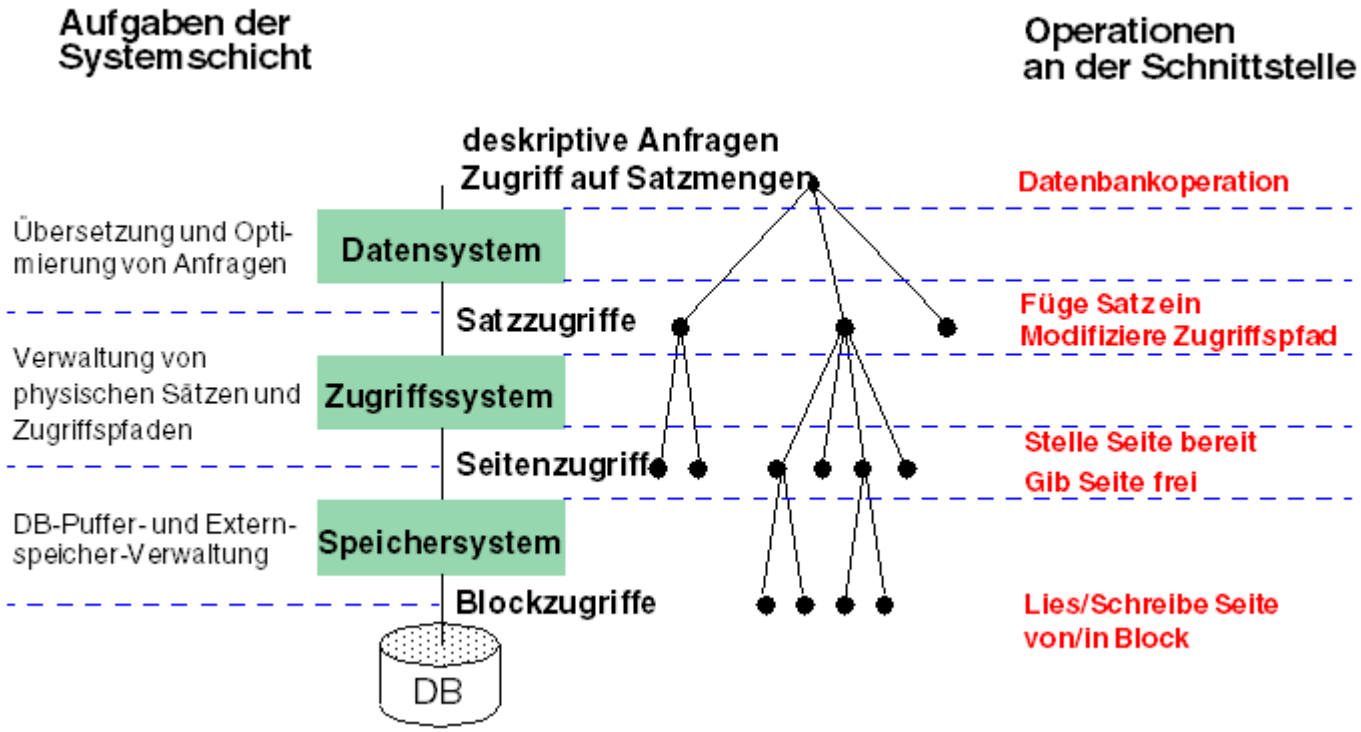
- NICHT für eine bestimmte Anwendung (z.B. CD-Verwaltung)
- SONDERN für eine Anwendungsklasse (z.B. Verwaltungssysteme allgemein)

MERKE

- Datenbanksysteme sind von der konkreten Anwendung unabhängige Softwaresysteme

Problem

- die spezifischen Datenstrukturen oder Eigenschaften einer Anwendung müssen einem Datenbanksystem
  - explizit mitgeteilt werden (Schemaentwurf) -> siehe Grundlagen
  - ABER NICHT PROGRAMMIERT werden


Aufgaben der Systemschicht
Operationen an der Schnittstelle
deskriptive Anfragen
Zugriff auf Satzmengen
Datenbankoperation
Übersetzung und Optimierung von Anfragen
Datensystem
Satzzugriffe
Füge Satz ein
Modifiziere Zugriffspfad
Verwaltung von physischen Sätzen und Zugriffspfaden
Zugriffssystem
Seitenzugriffe
Stelle Seite bereit
Gib Seite frei
DB-Puffer- und Externspeicher-Verwaltung
Speichersystem
Blockzugriffe
Lies/Schreibe Seite von/in Block
DB

Speichersystem

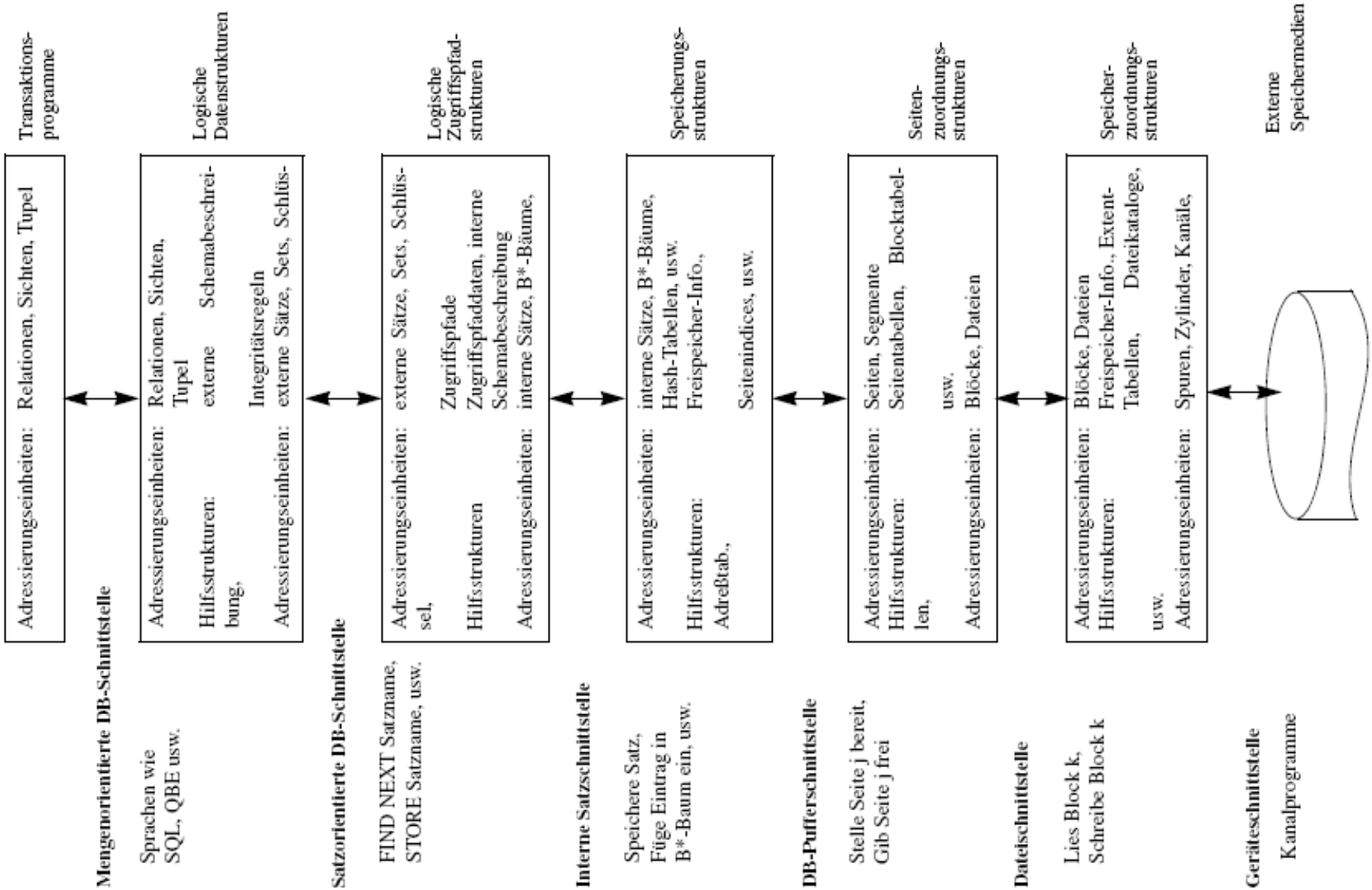
- Abbildung von Seiten und Segmenten auf Blöcke und Dateien
- Abbildung von Sätzen auf Seiten
- Satzadressierung, Einbringstrategien, Systempuffer

Zugriffssystem

- Eindimensionale Zugriffspfade (B*-Baum, Hashing, Invertierung, ...)
- Mehrdimensionale Zugriffspfade (Quad-Tree, K-d-Baum, Multi-Key Hashing, ...)

Datensystem (Anfrageverarbeitung)

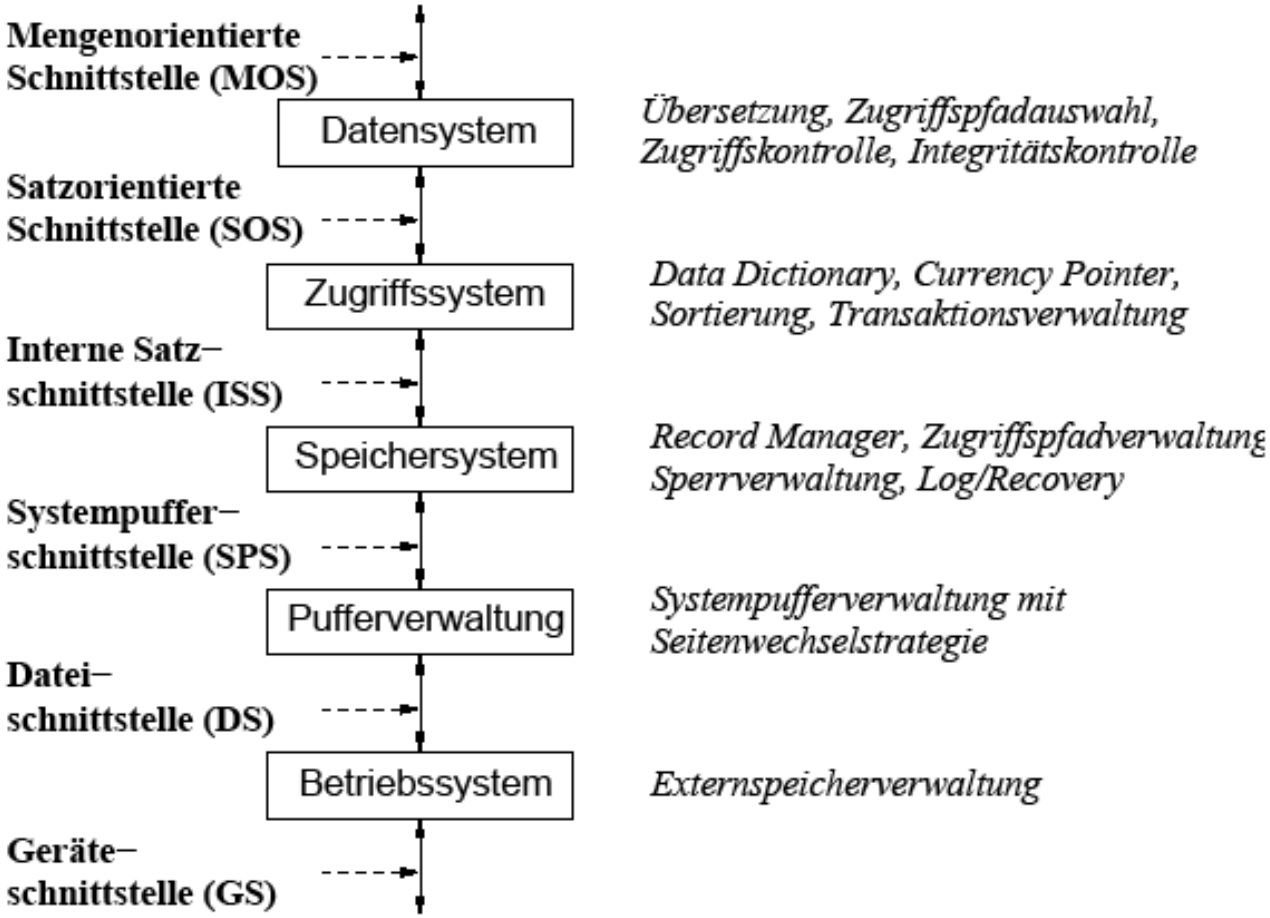
- Relationales Datenmodell und relationale Algebra
- Phasen der Anfrageverarbeitung
- Optimierungsstrategien in der Anfrageverarbeitung (predicate-pushdown, gb-pullup, ..)


Transaktions-programme
Logische Datenstrukturen
Logische Zugriffspfad-strukturen
Speicherungs-strukturen
Seiten-zuordnungs-strukturen
Speicher-zuordnungs-strukturen
Externe Speichermedien
Adressierungseinheiten: Relationen, Sichten, Tupel
Mengenorientierte DB-Schnittstelle
Sprachen wie SQL, QBE usw.
Adressierungseinheiten: Relationen, Sichten, Tupel
Hilfsstrukturen: externe Schemabeschrei-bung, Integritätsregeln
Adressierungseinheiten: externe Sätze, Sets, Schlüs-
Satzorientierte DB-Schnittstelle
FIND NEXT Satzname, STORE Satzname, usw.
Adressierungseinheiten: externe Sätze, Sets, Schlüs-sel, Zugriffspfade
Hilfsstrukturen Zugriffspfaddaten, interne Schemabeschreibung
Adressierungseinheiten: interne Sätze, B*-Bäume,
Interne Satzschnittstelle
Speichere Satz, Füge Eintrag in B*-Baum ein, usw.
Adressierungseinheiten: interne Sätze, B*-Bäume, Hash-Tabellen, usw.
Hilfsstrukturen: Freispeicher-Info., Adreßtab.,
Seitenindices, usw.
DB-Pufferschnittstelle
Stelle Seite j bereit, Gib Seite j frei
Adressierungseinheiten: Seiten, Segmente
Hilfsstrukturen: Seitentabellen, Blocktabel-len, usw.
Adressierungseinheiten: Blöcke, Dateien
Dateischnittstelle
Lies Block k, Schreibe Block k
Adressierungseinheiten: Blöcke, Dateien
Hilfsstrukturen: Freispeicher-Info., Extent-Tabellen, Dateikataloge, usw.
Adressierungseinheiten: Spuren, Zylinder, Kanäle,
Geräteschnittstelle
Kanalprogramme

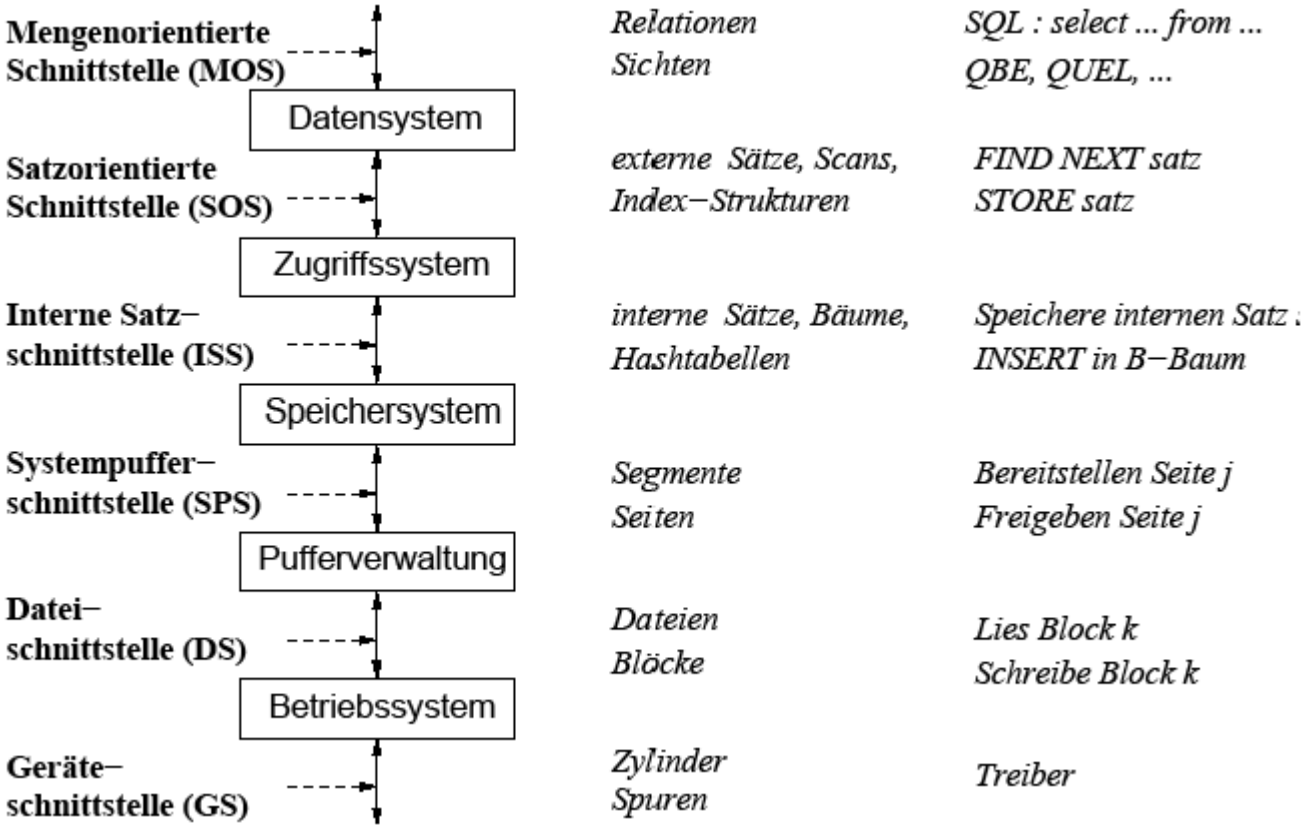
Übersicht über die Schnittstellen

- Mengenorientierte Schnittstelle
  - deklarative DML auf Tabellen, Sichten, Zeilen
- Satzorientierte Schnittstelle
  - Sätze, logische Dateien, logische Zugriffspfade
  - navigierender Zugriff
- Interne Satzschnittstelle
  - Sätze, Zugriffspfade
  - Manipulation von Sätzen und Zugriffspfaden
- Pufferschnittstelle
  - Seiten, Seitenadressen
  - Freigeben und Bereitstellen
- Datei- oder Seitenschnittstelle
  - Hole Seite, Schreibe Seite
- Geräteschnittstelle: Spuren, Zylinder, Armbewegungen

Übersicht über Funktionen



Übersicht über Datenobjekte und Objektstrukturen



MOS: mengenorientierte Schnittstelle

- deklarative Datenmanipulationssprache auf Tabellen und Sichten (etwa SQL)

SOS: satzorientierte Schnittstelle

- navigierender Zugriff auf interner Darstellung der Relationen
- manipulierte Objekte: typisierte Datensätze und interne Relationen sowie logische Zugriffspfade (Indexe)
- Aufgaben des Datensystems: Übersetzung und Optimierung von SQL-Anfragen

ISS: interne Satzschnittstelle

- interne Tupel einheitlich verwalten, ohne Typisierung
- Speicherstrukturen der Zugriffspfade (konkrete Operationen auf B-Bäumen und Hash-Tabellen), Mehrbenutzerbetrieb mit Transaktionen

SPS: Systempufferschnittstelle

- Umsetzung auf interne Seiten eines virtuellen linearen Adressraums
- Typische Operationen: Freigeben und Bereitstellen von Seiten, Seitenwechselstrategien, Sperrverwaltung, Schreiben des Protokolls

DS: Dateischnittstelle -> Umsetzung auf Geräteschnittstelle

Schichtenansatz

- “scheint” geeignet zur Strukturierung großer Anwendungssysteme
- Aufbau mehrschichtiger Software-Systeme

Datenbanksysteme

- Speichersystem
- Zugriffssystem
- Datensystem

Optional: 5-schichtiger Architekturansatz

- basierend auf Idee von Senko 1973; Weiterentwicklung von Härder 1987
- Umsetzung im Rahmen des IBM-Prototyps System R
- genaue Beschreibung der Transformationskomponenten (Dienste und Schnittstellen)