Dresden Database
Systems Group

1. Einführung

Architecture of database systems

Motivation – What? Why?

Petabyte Age

Dresden Database Systems Group

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.

Petabyte Age

Dresden Database Systems Group

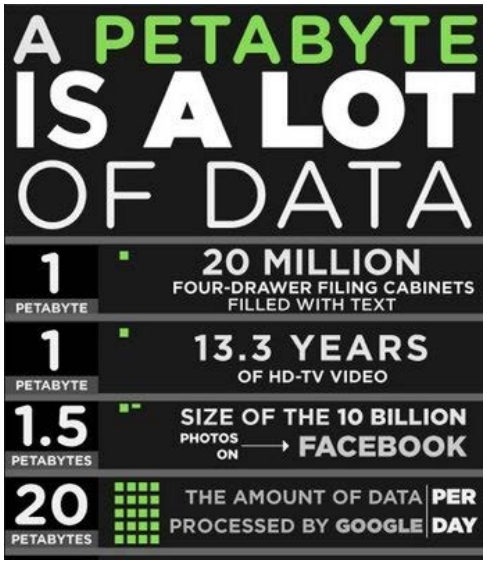
The Web is a huge source of information: search engines (Google, Yahoo!) collect and store billions of documents

- 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



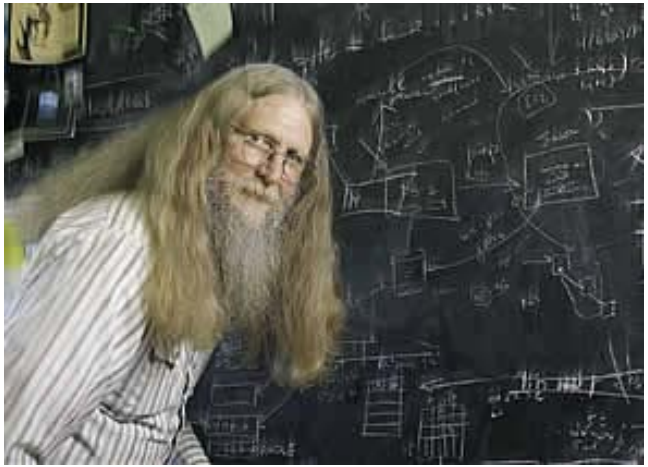
Other data produces

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

Warum Datenbanken ???

Dresden Database Systems Group

Relational databases are the foundation of western civilization



Bruce Lindsay,
IBM Fellow @ IBM Almaden Research Center

Konzepte der Datenbanktechnologie

Dresden Database Systems Group

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

Def. Datenbanksystem (Literatur)

Dresden Database Systems Group

Begriffseinführung:

"Datenbanksystem"

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

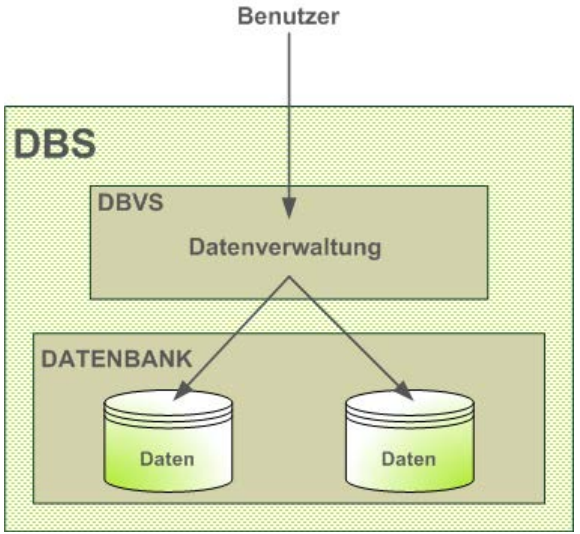
Komponenten

- 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 Bereichen

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

Darstellung



Landschaft aktueller Datenbanksysteme

Dresden Database Systems Group

Kommerzielle Systeme

Frei verfügbare Systeme

Allgemeine Herausforderungen

Datenbanksysteme aus der Sicht der Anwendung

Dresden Database Systems Group

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)

Dresden Database Systems Group

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)

Anfragesprachen

Dresden Database Systems Group

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 prozeduralen DB Sprachen wird spezifiziert, **WIE** das DBS zu suchen hat
- In deskresptiven DB-Sprachen wird spezifiziert, **WAS** das DBS zu suchen hat

Anfragesprache SQL

Dresden Database Systems Group

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)
)

- Schemadefinition der Daten
- Primärschlüssel & Fremdschlüssel

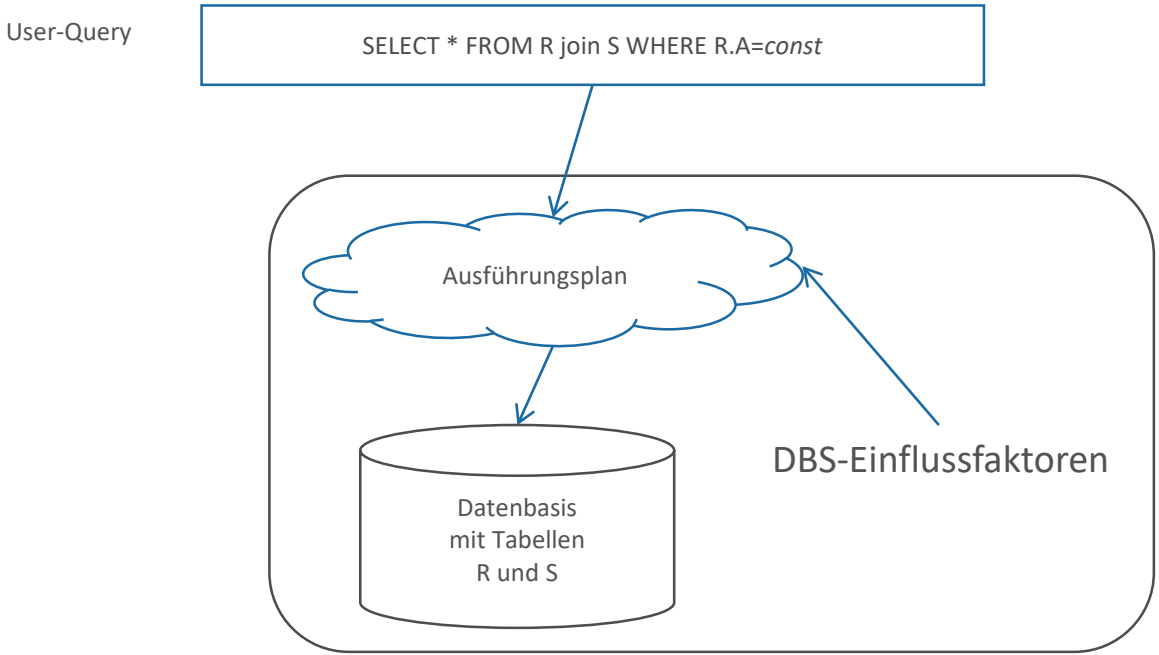
DML (Data Manipulation Language)

- 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)

SQL Transformation

Dresden Database Systems Group

Herausforderung (Ausführungsplan)



Beispiel zur SQL-Optimierung

Dresden Database Systems Group

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

Optimierungsarten

Dresden Database Systems Group

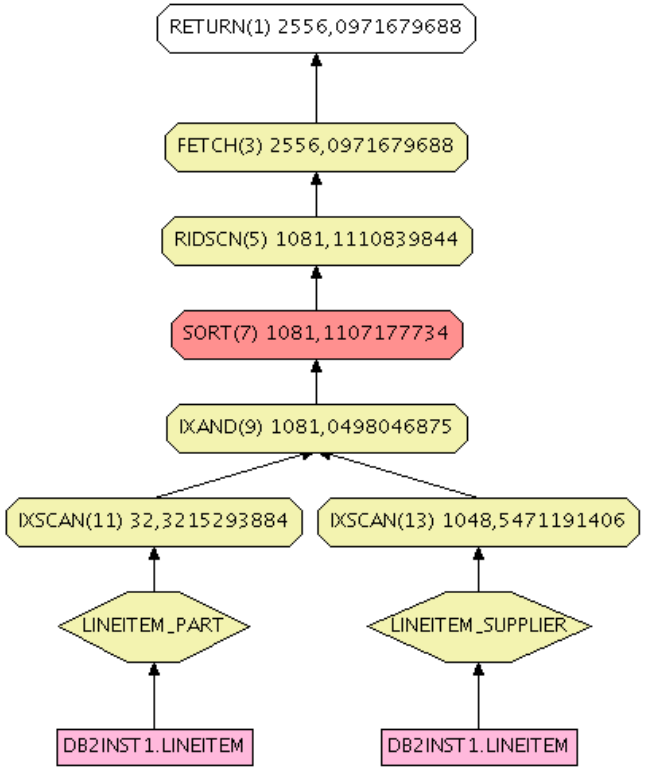
Logische Optimierung

- 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

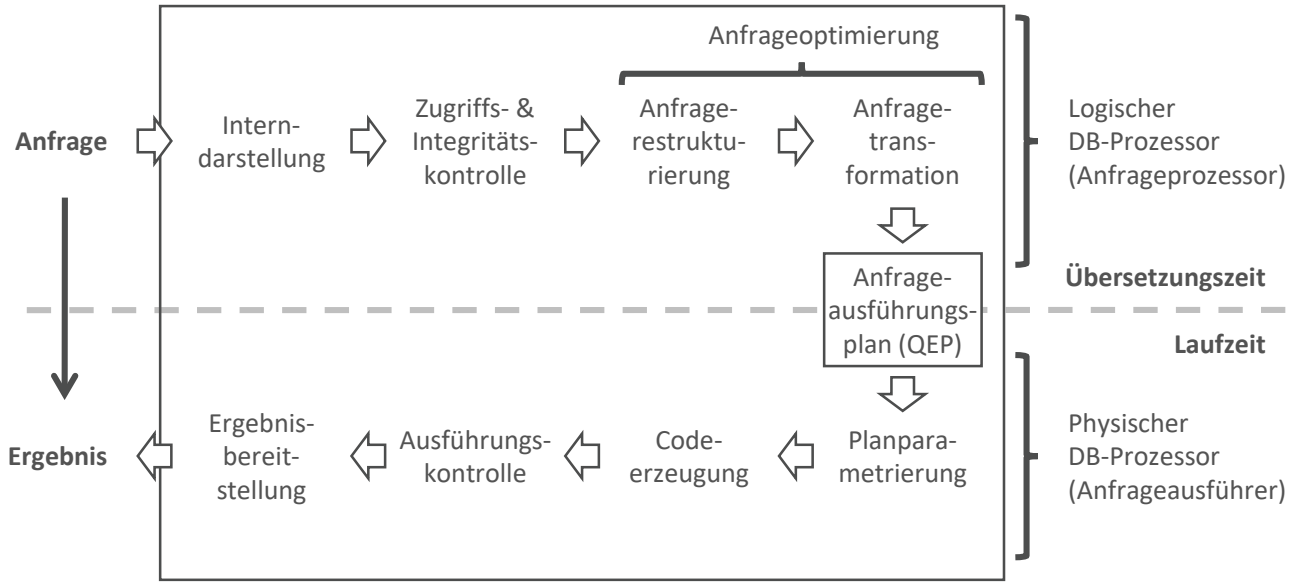
Physische Optimierung

- 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

Beispiel eines Anfragegraphens



Phasen der Anfrageverarbeitung



Charakterisierung von Transaktionen

Dresden Database Systems Group

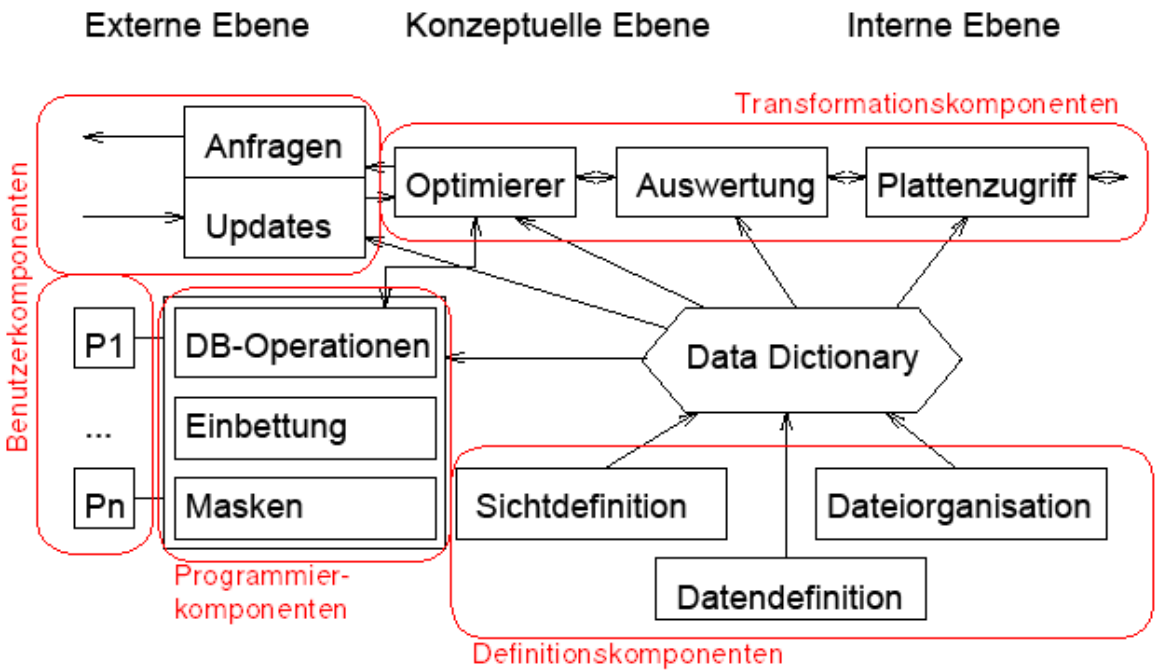
Charakterisierung

- 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)

Bestandteile eines Datenbanksystems

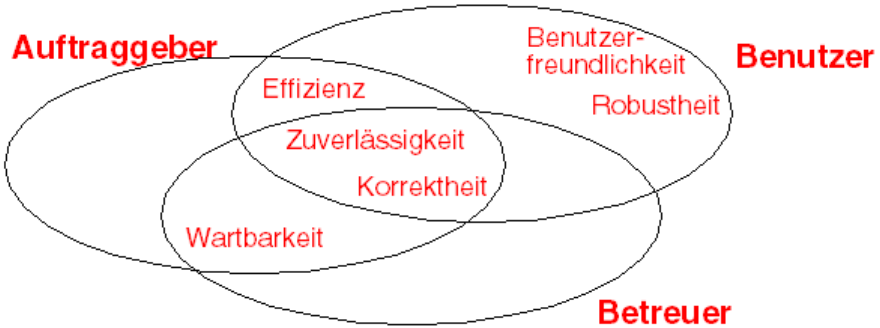


Schichtenmodell

DBS als Beispiel eines generischen Softwaresystems

Dresden Database Systems Group

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

Allgemeine Betrachtungen zur Schichtenbildung

Dresden Database Systems Group

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

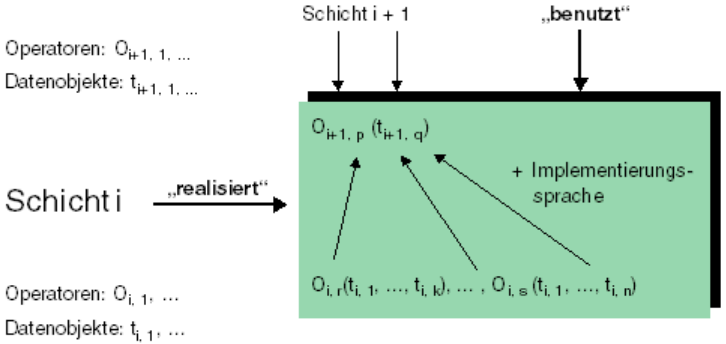
Aufbauprinzip einer Schicht

Dresden Database Systems Group

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 der Schichtenbildung

Dresden Database Systems Group

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.

Generischer Systementwurf

Dresden Database Systems Group

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)

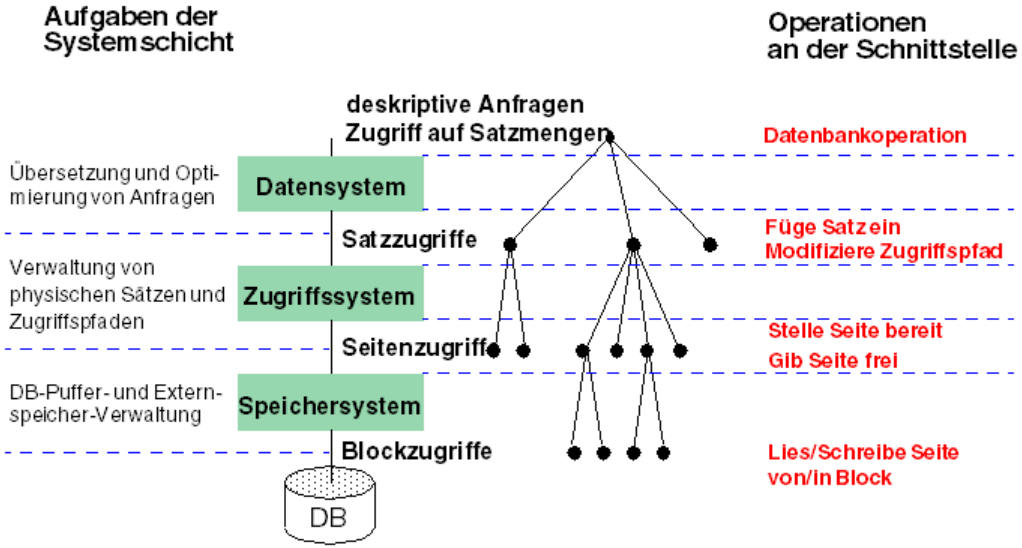
MERKE

- 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

Schichtenmodell und Kontrollfluss einer Operation



Übersicht des vereinfachten Schichtenmodells

Dresden Database Systems Group

Speichersystem

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

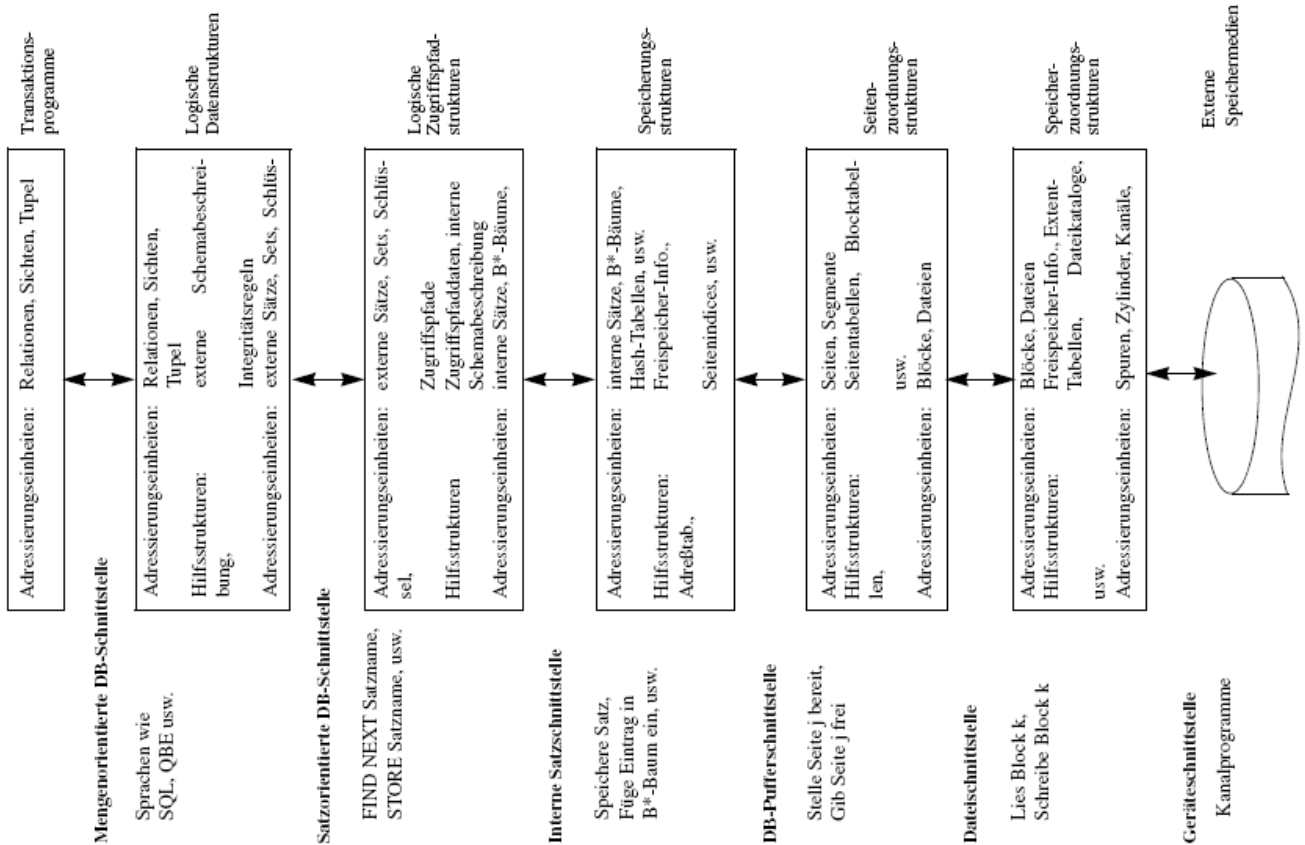
Zugriffssystem

- 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, ..)

5-Schichtenmodell eines DBS



5-Schichten-Architektur

Dresden Database Systems Group

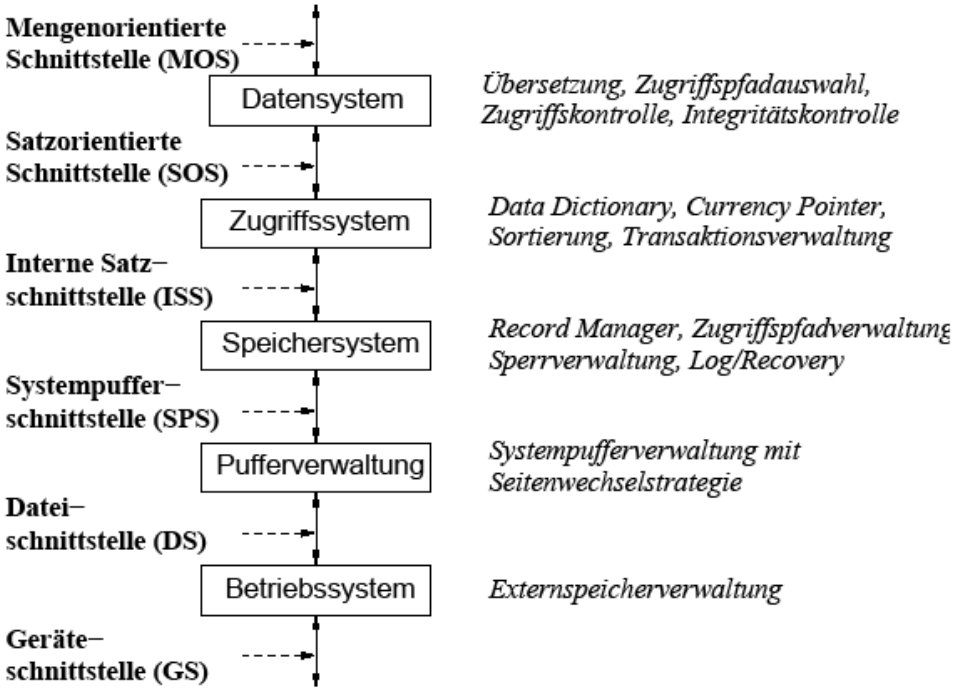
Übersicht über die Schnittstellen

- 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

5-Schichten-Architektur (2)

Dresden Database Systems Group

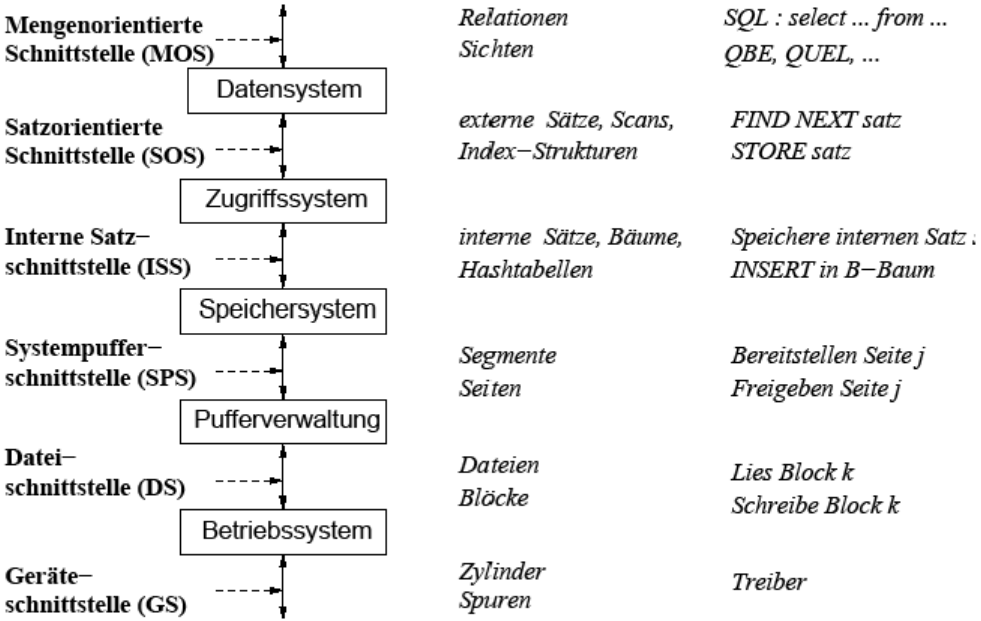
Übersicht über Funktionen



5-Schichten-Architektur (3)

Dresden Database Systems Group

Übersicht über Datenobjekte und Objektstrukturen



5-Schichten-Architektur (4)

Dresden Database Systems Group

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

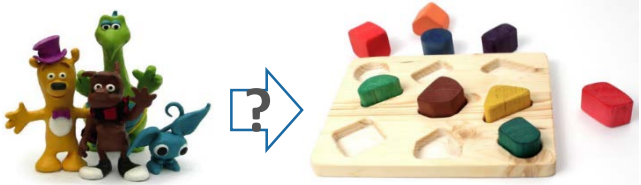
DS: Dateischnittstelle -> Umsetzung auf Geräteschnittstelle

Aktuelle Herausforderungen/Entwicklungen

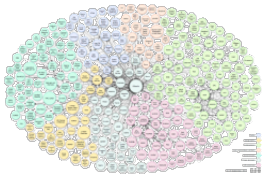
Aktuelle Herausforderungen

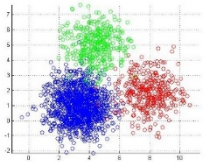
Dresden Database Systems Group

Schema-Flexible Storage...

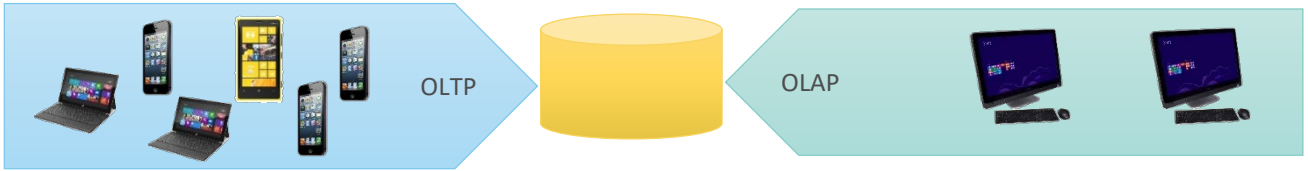


Mixed Procedural/Declarative Code...

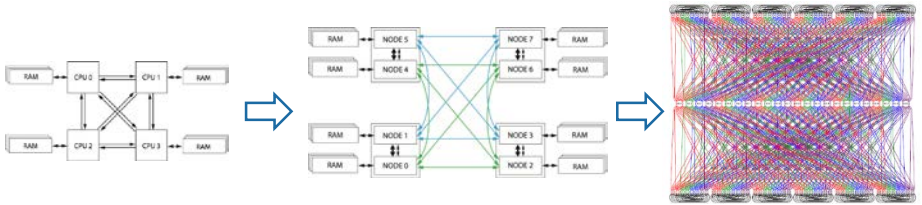




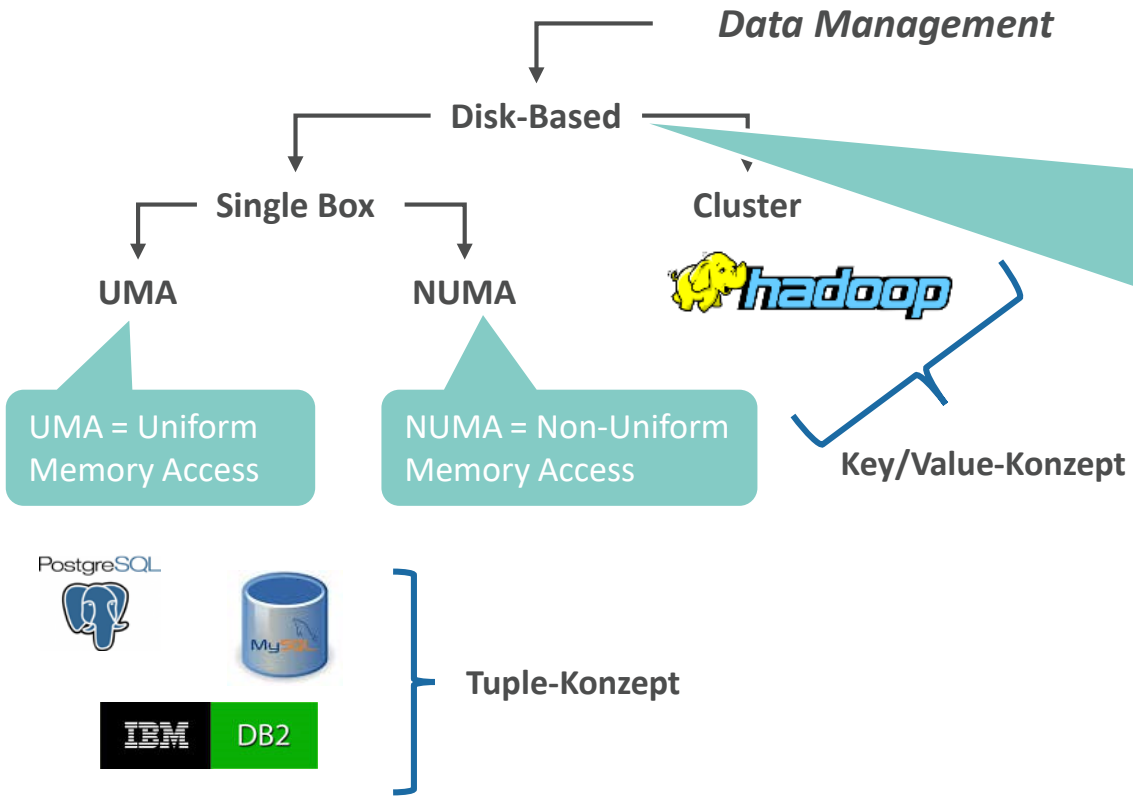
Balanced Read/Write Performance...



Hardware-Awareness...



DBS-Umfeld

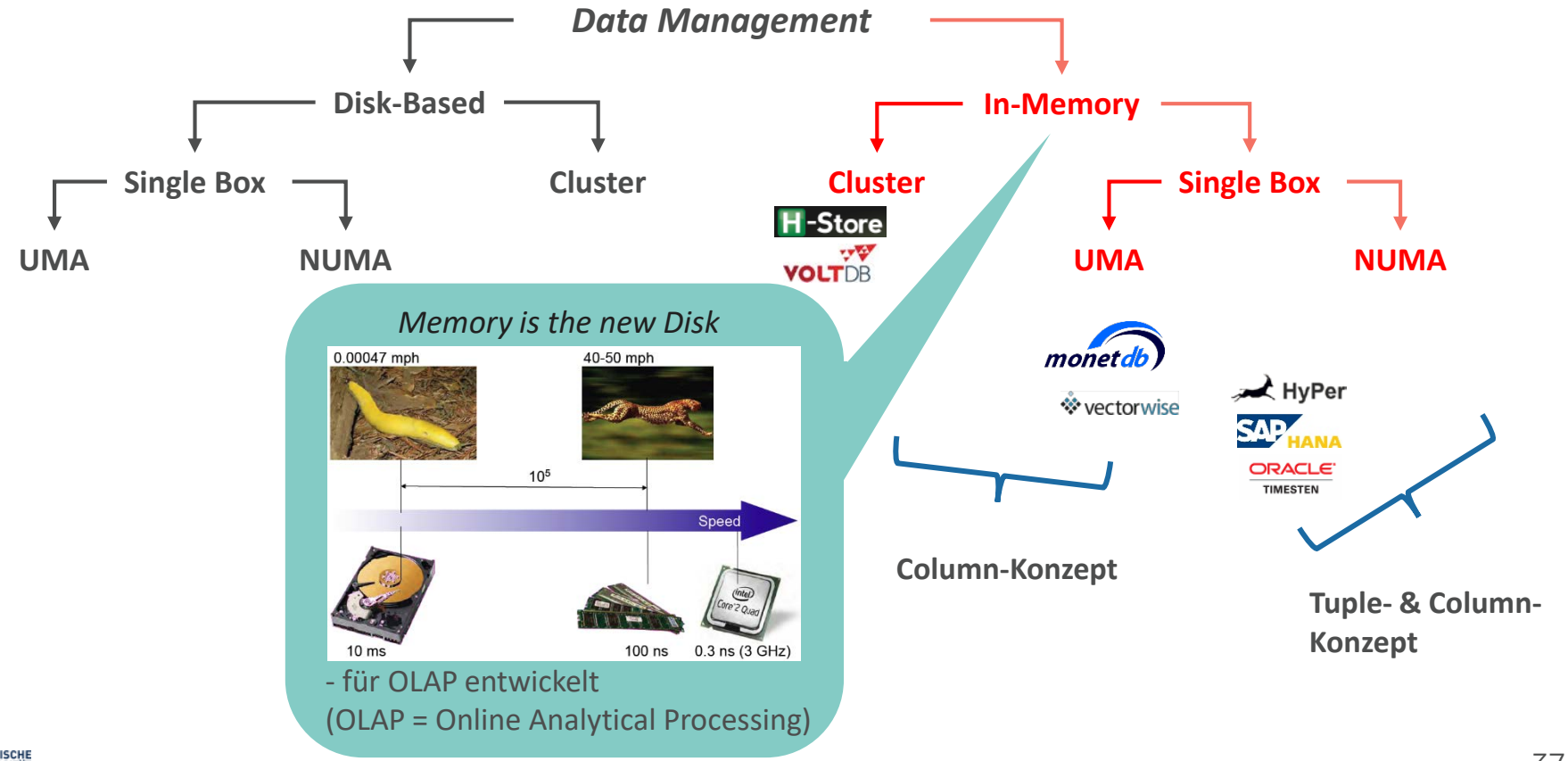


Klassischer Ansatz

- Daten werden primär auf Festplatte gespeichert und zur Verarbeitung in den Hauptspeicher geladen
- Festplattenzugriff eine teure Aktion, Daten sind aber persistent
- Insbesondere für OLTP entwickelt (OLTP = Online Transaction Processing)

DBS-Umfeld

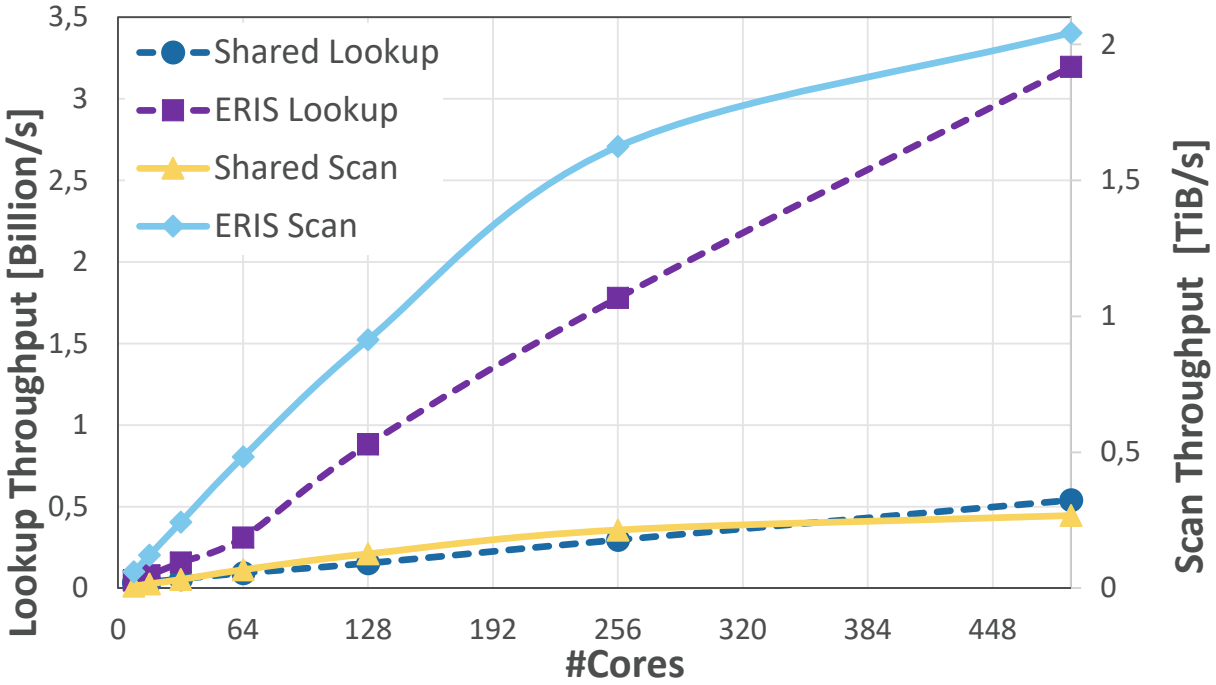
Dresden Database Systems Group



Dresden Database Systems Group

In-Memory UMA versus NUMA

- UMA-Ansatz → Shared
- NUMA-Ansatz → ERIS



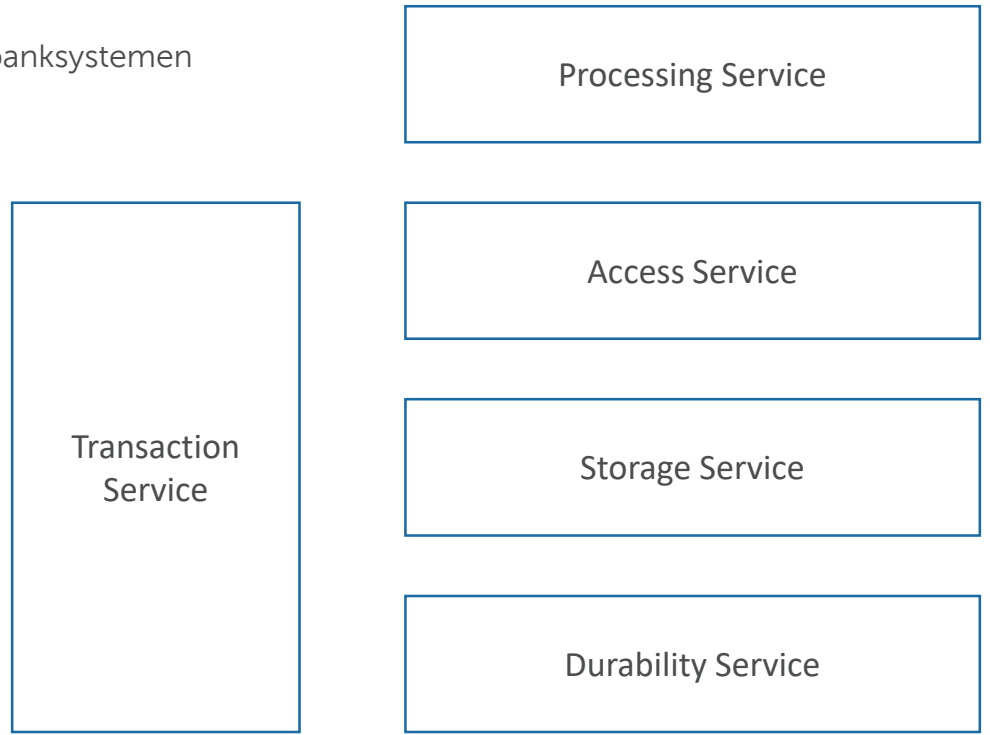
Aufbau Architektur von Datenbanksystemen

Dresden Database Systems Group

Aktuell

- Viel Bewegung in der Architektur von Datenbanksystemen
- Reflektion der Bewegung in der Vorlesung

- Aufbau der Architektur über Services



Aufbau Architektur von Datenbanksystemen (2)

Dresden Database Systems Group

Aktuell

- Viel Bewegung in der Architektur von Datenbanksystemen
- Reflektion der Bewegung in der Vorlesung

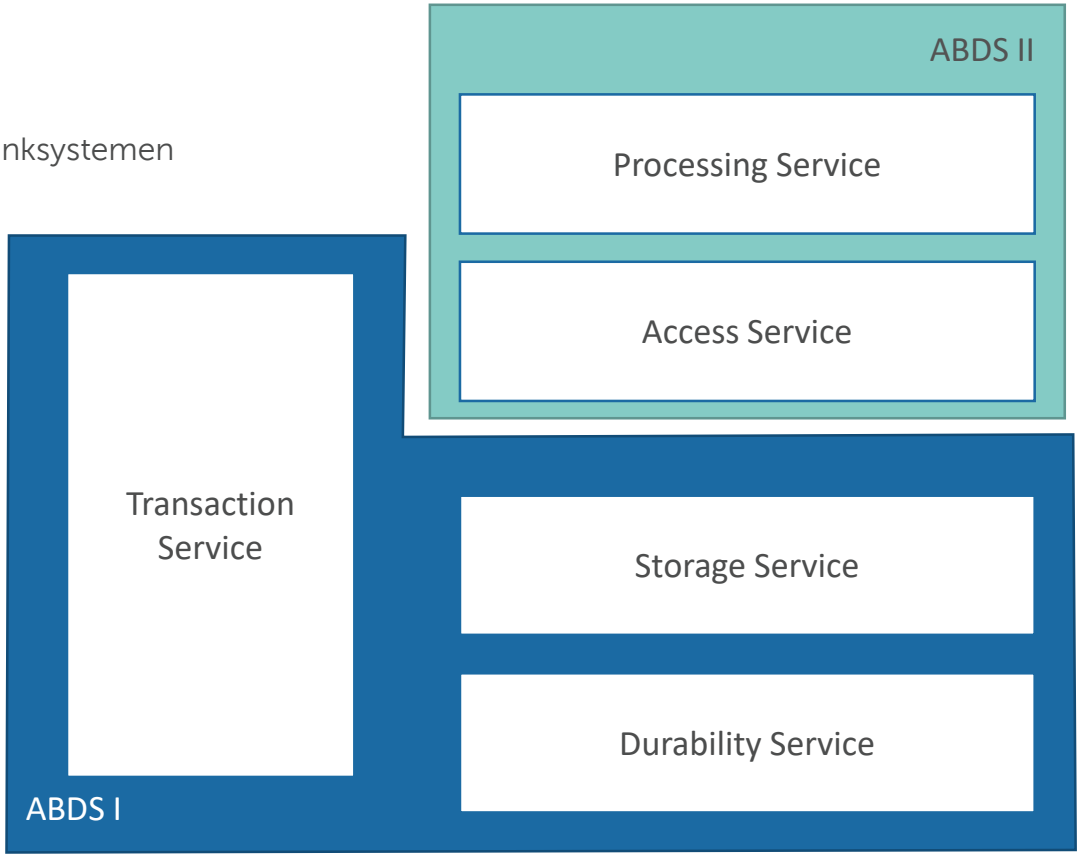
- Aufbau der Architektur über Services

ADBS I

- Speicherung der Daten
- Realisierung der Dauerhaftigkeit
- Transaktionsverarbeitung

ADBS II

- Zugriffskonzepte
- Verarbeitungskonzepte



Dresden Database Systems Group

Inhalt der Vorlesung