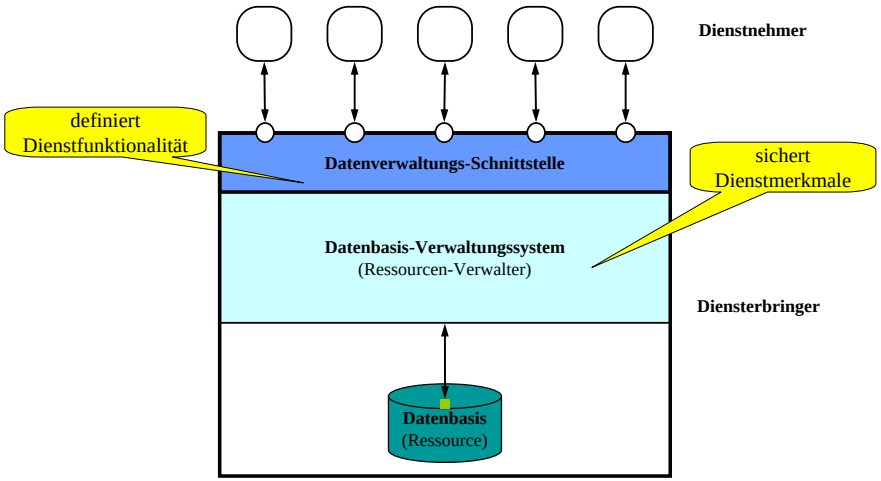
Wo stehen wir?

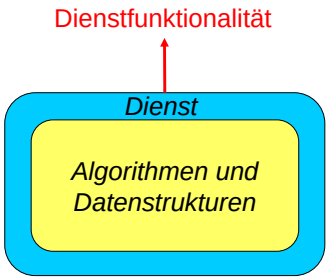
Letzte Woche:

- viele Begriffe: Prozess, Dienst, Dienstnehmer, Dienstgeber, Dienstfunktionalität, Dienstmerkmal
- Datenbankdienst:
  - Dienstfunktionalität ← Was der Dienst tut
  - Dienstmerkmale ↑ Randbedingungen

Datenverwaltungssystem



Dienstfunktionalität der Datenverwaltung



Dienstfunktionen für das
- Entgegennehmen,
- Abspeichern,
- Ändern,
- Löschen,
- Auswählen,
- Bereitstellen

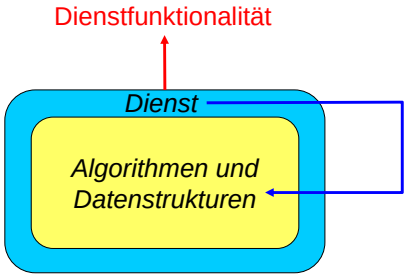
von Daten.

- Dienstfunktionen besitzen allgemeingültigen Charakter,
- Wirtschaftlichkeitsgründe diktieren „breitbandige“ Einsatzfähigkeit

⇒ **Generisches Vorgehen**.

Dienstfunktionalität und Dienstmerkmale (2)

Ressourcen-Verwalter



Dienstmerkmale

Bedeutungstreue
Gemeinsames Verständnis der Daten bei allen am Austausch beteiligten Partnern

Dauerhaftigkeit
Zugriff auf Daten nach ihrer Entstehung zu jeder Zeit in der Zukunft

Allgegenwart
Datenzugriff zu jeder Zeit von jedem Ort

Leistung
- Effizienz: Minimaler Verbrauch innerer Ressourcen
- Verfügbarkeit: Rechtzeitiger Zugang zu den Ressourcen

Skalierbarkeit
Stetes Wachstum ohne Einbußen bei Funktionalität und Leistung

Robustheit
Zuverlässiges Erbringen der Dienste unter Störungen und Fehlern

Sicherheit
Keine Verluste, Störungen, Ausfälle durch unbedachte oder böswillige Eingriffe

Wo stehen wir?

Letzte Woche:

- viele Begriffe: Prozess, Dienst, Dienstnehmer, Dienstgeber, Dienstfunktionalität, Dienstmerkmal
- Datenbankdienst:
  - Dienstfunktionalität
  - Dienstmerkmale: Bedeutungstreue

Heute:

- noch mal: Bedeutungstreue
- weitere Dienstmerkmale: Dauerhaftigkeit, Leistung, Skalierbarkeit, Robustheit
- Architektur von Datenbanksystemen

Beispiel

Zugrundeliegender polymorpher Typ (Schablone für monomorphe Typen)

```
Datenbasistyp  Buchungsdatenbasis

Typ   Flughafen ::= datensatz  [
      FlughCode: string, Stadt: string, Land: string,
      Name: string, Zeitzone: int ]
Typ   Flugzeugtyp ::= datensatz  [
      FtypId: string, Name: string, First: int,
      Business: int, Economy: int ]
Typ   Flug ::= datensatz  [
      FlugNr: string, von: string, nach: str
      FtypId: string, Wochentag: string,
      Abflugszeit: time, Ankunftszeit: time,
      Entfernung: int ]
Typ   Flughäfen ::= menge  {Flughafen} key FlughCode
Typ   Flugzeugtypen ::= menge  {Flugzeugtyp} key FtypId
Typ   Flüge ::= menge  {Flug} key FlugNr
```

monomorphe Konsistenzbedingung

Weitere denkbare Konsistenzbedingungen

```
Datenbasistyp  Buchungsdatenbasis

Typ   Flughafen ::= datensatz  [
      FlughCode: string, Stadt: string, Land: string,
      Name: string, Zeitzone: int ]
Typ   Flugzeugtyp ::= datensatz  [
      FtypId: string, Name: string, Fir
      Business: int, Economy: int ]
Typ   Flug ::= datensatz  [
      FlugNr: string, von: string, nach: string,
      FtypId: string, Wochentag: string,
      Abflugszeit: time, Ankunftszeit: time,
      Entfernung: int ]
Typ   Flughäfen ::= menge  {Flughafen} key FlughCode
Typ   Flugzeugtypen ::= menge  {Flugzeugtyp} key FtypId
Typ   Flüge ::= menge  {Flug} key FlugNr
```

Land → *Zeitzone*

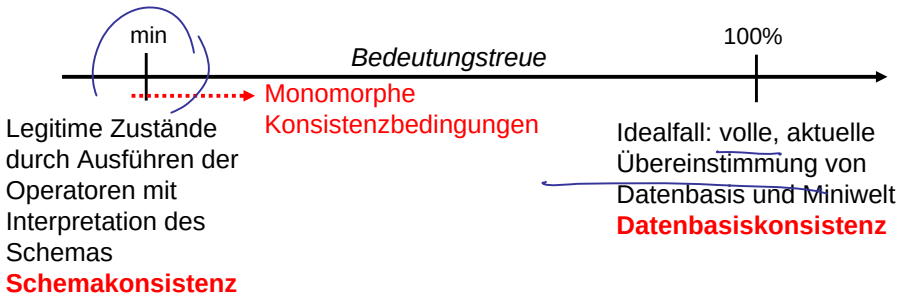
Wert unter *von* kommt als Wert unter *FlughCode* vor

(*von*,*nach*) → *Entfernung*

Zustandsübergänge (1)

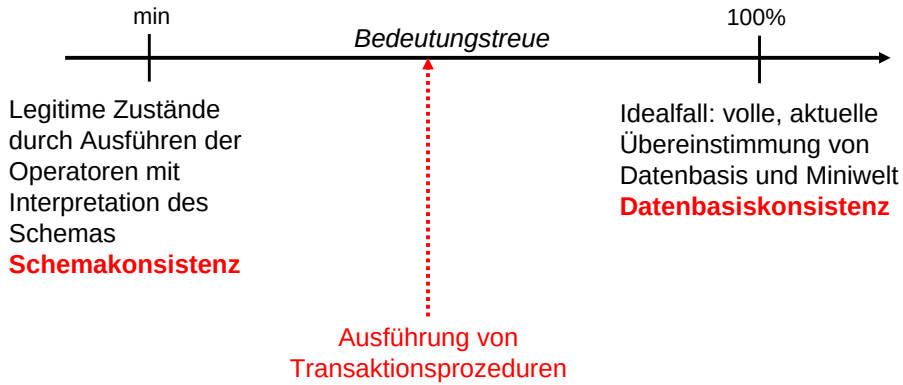
- Ein Datenverwaltungssystem gewährleistet (Schema-) Konsistenz, wenn seine Dienstfunktionen stets einen (schema-)konsistenten Zustand seiner Datenbasis wieder in einen (schema-)konsistenten Zustand überführen.
- Gehorcht die Datenbasis einer wie immer gearteten Schemakonsistenz vor Ausführen einer Dienstfunktion, so gehorcht sie ihr auch zum Abschluss der Ausführung.

Konsistente Zustände (1)



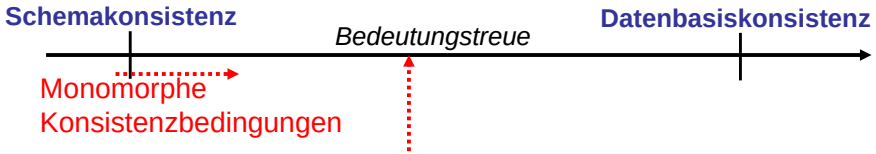
Datenmodell legt Regeln fest, nach denen Zustandsräume konstruiert werden können.
Schema legt bestimmten Zustandsraum fest.

Konsistente Zustände (2)



Konsistenz: Grad an Bedeutungstreue einer Datenbasis

Zustandsübergänge (2)



Transaktionsprozedur: Dienstprozedur als frei definierte Abfolge von Operationen, die Zustandsübergänge einschränkt: Prozedurale Kontrolle der Konsistenz

- Will man also Konsistenz durchsetzen: Als Dienstfunktionen sind ausschließlich Transaktionsprozeduren zulässig.
  - Die Bedeutungstreue ist an das Ende der Prozedur gebunden, während der Ausführung, also am Ende einzelner Operationen, muss sie nicht zutreffen.
  - Daher dürfen Konsistenzbedingungen erst am Ende durchgesetzt werden.

Beispiel

Mustert eine Fluggesellschaft einen Flugzeugtyp aus, so ist dessen Datensatz mit der entsprechenden FtypId aus der Menge der Flugzeugtypen zu entfernen, gleichzeitig sind alle Flug-Datensätze, die sich auf diese FtypId beziehen, auf den neuen Flugzeugtyp umzustellen.

Beispiel

- Betrachten Sie noch einmal den Internet-Landkarten-Shop. Dort sollen auch Landkarten diverser ausländischer Verlage angeboten werden. Von diesen Verlagen beziehen Sie die Karten über unterschiedliche Importeure. Einer Ihrer bisherigen Importeure wurde von einem Importeur aufgekauft, mit dem Sie bisher keine Geschäftsbeziehungen hatten. Alle Verlage, die bisher vom alten Importeur abgedeckt wurden, werden jetzt durch den neuen Importeur bedient.
  - Welche Konsistenzbedingung muss sinnvollerweise hinsichtlich der Importeure gelten?
  - Welche Operationen müssen Sie auf der Datenbank ausführen?
  - Zu welchen Zeitpunkten befindet sich die Datenbank in einem inkonsistenten Zustand?

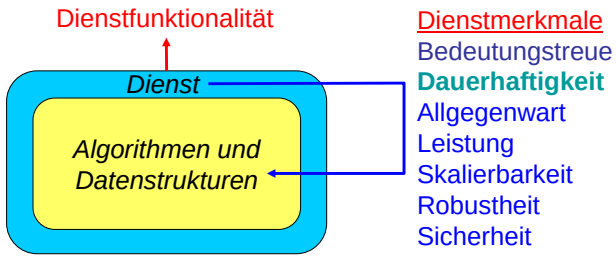
Zusammenfassung Bedeutungstreue

- Schema definiert Menge der im Betrieb zulässigen Zustände und Zustandsübergänge.
- Charakterisierung des Zustandsraumes durch:
  - Auswahl konkreter Typen aus dem Typsystem des Datenmodells,
  - weitere Eingrenzung durch Konsistenzbedingungen
  - und (implizit) durch Transaktionsprozeduren.
- Charakterisierung der Zustandsübergänge durch:
  - Spezialisierung der polymorphen Operatoren des Datenmodells auf die im Schema auftretenden Typen,
  - weitere Eingrenzung durch Transaktionsprozeduren.
- **Datendefinitionssprache** (DDL) dient zur Formulierung des Schemas.
- **Datenmanipulationssprache** (DML) dient zum nachfolgenden Umgang mit der Datenbasis.

Durchsetzen von Konsistenz (2)

- **Datenbasistransaktion** (auch kurz: Transaktion): Ausführung einer Transaktionsprozedur.
- Folgerung: Konsistenz ist erst am Ende einer Transaktion definiert.
- Somit wird auch das Durchsetzen von Konsistenzbedingungen einer Transaktionsverwaltung übertragen. Eine Transaktion überprüft also am Ende, ob sich eine Konsistenzbedingung erfüllen lässt bzw. erfüllt ist. Trifft dies nicht zu, so wird eine Korrekturaktion ausgelöst oder ein Fehler signalisiert.
- Beispiel: In einer Transaktion wird ein insert-Operator gezwungen, gegen die Schlüsselbedingung zu verstoßen.
- **Anwendungstransaktion**: Konsistente Ausführung eines Leistungsprozesses.

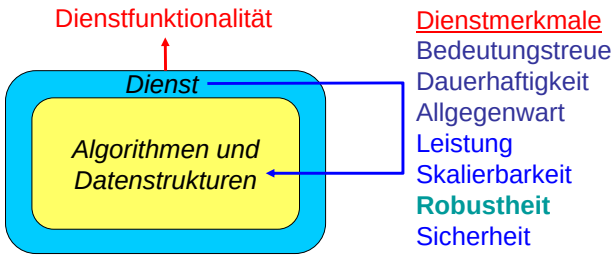
Dauerhaftigkeit in der Datenverwaltung



- Verbringen der Daten auf nichtflüchtiges Speichermedium genügt nicht.
- Dauerhafte Daten müssen zudem konsistent sein.
- Das sind sie, wenn sie von abgeschlossenen Transaktionen stammen.

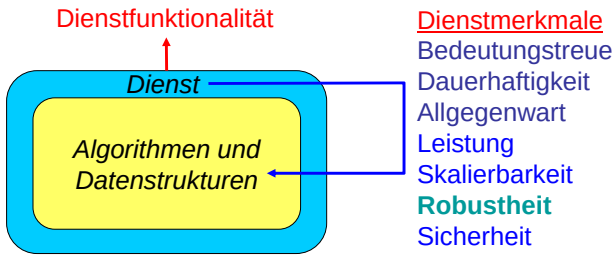
Dauerhaftigkeit ist die andauernde Nichtflüchtigkeit gewisser, in jedem Fall konsistenter Datenbasiszustände (Unverletzlichkeit der Datenbasis).

Robustheit 1: Persistenz



- Erreichen eines konsistenten nichtflüchtigen Zustands: **Persistenz**
- Folgerung 1: Dauerhaftigkeit ist potenziell unbegrenzte Lebensdauer eines persistenten Zustandes.
- Folgerung 2: Transaktionsabschluss ist ein Persistenz-Ereignis.

Robustheit 2: Resistenz



- Wahrung von Persistenz (und damit auch Konsistenz) auch unter dem Einfluss von Störungen: **Resistenz**
- Störung des Dienstnehmer-Dienstgeber-Verhältnisses: **Fehler-Resistenz**.
- Störung des Dienstnehmer-Dienstnehmer-Verhältnisses: **Konflikt-Resistenz**.

Transaktionen (1)

- Präzisierung von Robustheit:
  - Datenbasis führt — ungeachtet möglicher Störungen — nur solche Zustandsübergänge aus, die explizit durch Aufruf von Transaktionsprozeduren angefordert wurden.
- Impliziert zwei Garantien:
  - **Persistenz**: Effekte von abgeschlossenen Zustandsübergängen gehen nicht durch Störungen verloren, d.h., Datenbasiszustand ändert sich nicht „einfach so“.
  - **Resistenz**: Angeforderte Zustandsübergange werden entweder wie verlangt durchgeführt, oder wenn Zustandsübergang wegen Störung nicht abgeschlossen werden kann, strebt Datenbasis definierten konsistenten Zustand, z.B. Ausgangszustand, an.

Beispiel-Transaktion

- Gewünschter Zustandsübergang: Ausmusterung eines Flugzeugtyps.
- Transaktionsprozedur:
  - Entferne zugehörigen Datensatz aus Tabelle Flugzeugtypen.
  - Ersetze in Tabelle Flüge alle Vorkommen der alten FtypId durch FtypId des Ersatzmodells.
- Beachte: Datenbasis-Zustand ist zwischen Schritt 1 und 2 inkonsistent, davor und danach konsistent.
- Resistenz garantiert, dass bei Störung entweder Zustand vor Schritt 1 oder nach Schritt 2 erreicht wird.
- Persistenz garantiert, dass nach Abschluss von Schritt 2 Ergebnis nicht mehr verloren gehen kann.

Transaktionen (2)

- Transaktionsbegriff fasst Konsistenz, Persistenz und Resistenz zusammen:
- **Transaktion**: resistente Ausführung eines konsistenten Zustandsübergangs (Operator oder Transaktionsprozedur) mit Persistenz des Endzustands.

Fehler-Resistenz (1)

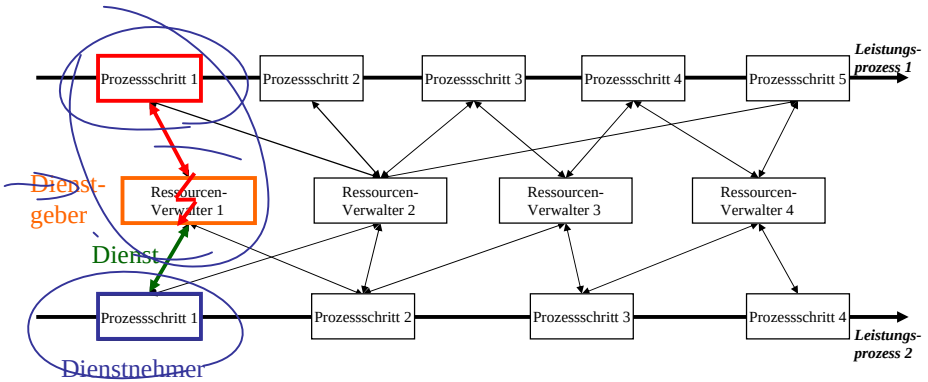
Beispiele für Störungen:

- Fehlerhafte Eingaben; Programmierfehler; unbeabsichtigte Wechselwirkung von Dienstleistungen für unterschiedliche Dienstnehmer; Programmfehler in der Datenverwaltungssoftware; Ausfall von Hintergrundspeichern, Datenträgern oder dem Rechner.

Fehler-Resistenz (2)

- So lange es zu keinen Störungen kommt, wird die Datenbasis einfach fortentwickelt. Tritt eine Störung auf, so wird ein früherer persistenter Zustand angestrebt.
- Wann ein persistenter Zustand vorliegt, muss dazu ausdrücklich dem Datenverwaltungssystem bekannt gemacht werden.
- Verbreitet: Da eine Transaktion ein persistentes Ergebnis bewirkt, ist der Abschluss einer Transaktion ein Persistenz-Ereignis. Die Transaktionsprozedur ist so zu entwerfen, dass man sie für beendet erklärt genau dann, wenn ihre Ergebnisse nicht mehr verloren gehen dürfen.
- Kommt die Transaktion nicht zum Abschluss, so ist der jüngste persistente Zustand der Zustand unmittelbar vor Ausführung der Transaktion. Man wird daher diesen Zustand wieder herstellen.

Konflikt-Resistenz (1)



Störfaktor für die Konsistenz: Bemühen sich mehrere Dienstnehmer-Transaktionen zur selben Zeit um dieselbe Ressource, so kann es zu einer unerwünschten Wechselwirkung zwischen ihnen kommen (**Konflikt**).

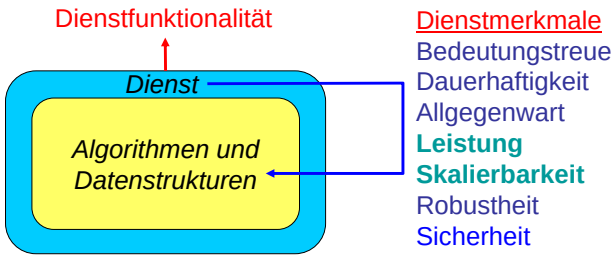
Konflikt-Resistenz (2)

- **Konkurrenz** um gemeinsame Daten (engl.: concurrency für Nebenläufigkeit): Konfliktverhalten von Transaktionen.
- **Benötigt**: Protokoll, das Konflikte zwischen *nebenläufigen* Dienstnehmer-Transaktionen eines Datenverwaltungssystems unterbindet (**Synchronisations-Protokoll**).
- **Korrektheitskriterium**: Wahrung der Konsistenz durch: Konkurrierende Datenbasistransaktionen sind dann korrekt synchronisiert, wenn jede so abläuft als ob sie ohne Konkurrenz wäre, insbesondere also dem Dienstnehmer keinen inkonsistenten Datenbasiszustand präsentiert und bei Abschluss einen persistenten Datenbasiszustand erreicht.

Persistenz und Dauerhaftigkeit

- Persistenz schafft die Voraussetzungen für Dauerhaftigkeit.
- Unbegrenzt lange Speicherung der Daten verschärft die Forderung nach Konsistenz, denn es steigt die Wahrscheinlichkeit, dass durch Störungen oder Fehler die Daten in ihrem Inhaltsbezug korrumpiert werden.
- Beispiele: Ausfall von Hintergrundspeichern, Datenträgern oder dem Rechner; externe Ereignisse wie Feuer, Wasser, Klima, Alterung mit physischer Vernichtung der Datenbasis.

Skalierbarkeit/Leistung in der Datenverwaltung



- Enge Wechselwirkung zwischen Skalierbarkeit und Leistung
- Daher gemeinsame Betrachtung: **Performanz**

Wechselwirkung Skalierbarkeit/Leistung

Aspekte der Leistung:
- **Effizienz**: Möglichst geringer Ressourcenverbrauch durch die Dienste.
- **Verfügbarkeit**: Bedarfsaktueller Zugang zu den Ressourcen.

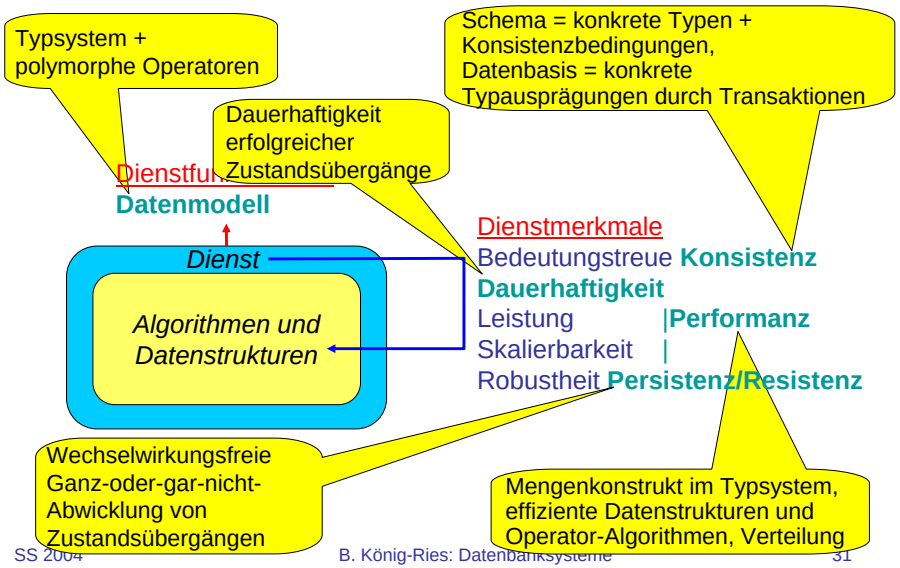
Aspekte der Skalierbarkeit:
- Wachstum der Datenbasis ⇔ Wachstum der Transaktionslast

Skalierbarkeit

- Forderung:
  - Leistung des DBMS darf mit wachsender DB-Größe und wachsender Nutzungsintensität (Anzahl und Komplexität der angeforderten Transaktionen) nicht kollabieren.
- Maßzahlen für Leistung:
  - **Durchsatz** = Anzahl abgeschlossener Transaktionen pro Sekunde,
  - **Antwortzeit** = Zeitspanne zwischen Beginn und Ende einer Transaktion.
- Realisierung durch:
  - effiziente Datenstrukturen für Datenbasis-Zustand,
  - effiziente Algorithmen für Operatoren,
  - Verteilung des DBMS auf mehrere Rechner.

Kapitel 1: Datenbankdienste

Dienstfunktionalität und Dienstmerkmale



Kapitel 2: Referenzarchitektur für Datenbanksysteme

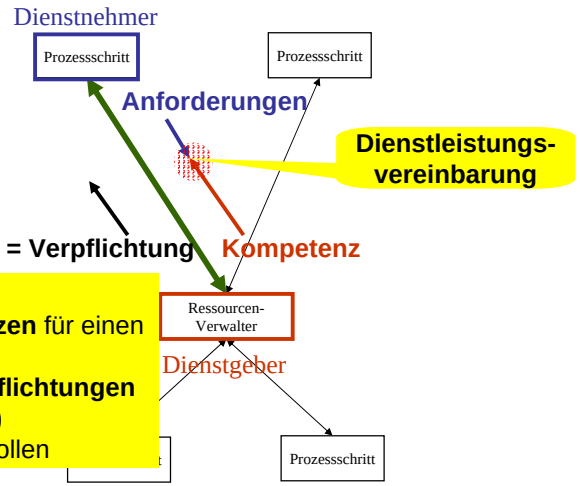
Kapitel 2: Referenzarchitektur für Datenbanksysteme

Methodischer Architekturentwurf

Architekturentwurf für Datenbanksysteme

Referenzarchitektur

Was ist ein Dienst?

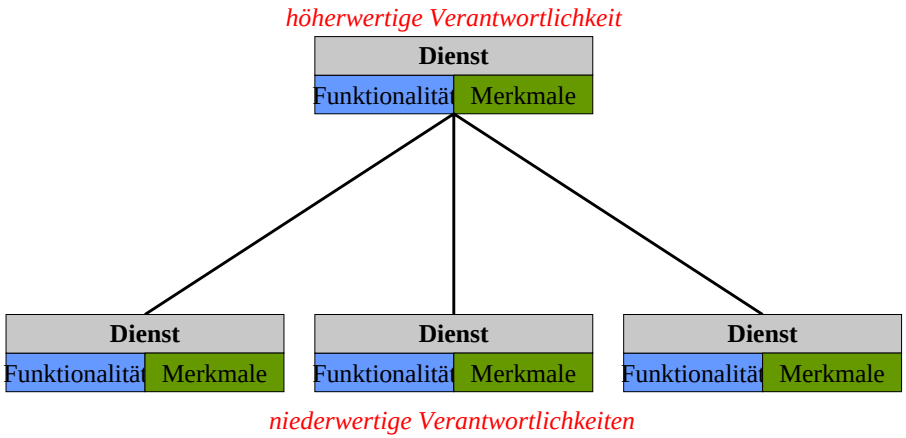


Dienstentwurf:

- Ermittle die **Kompetenzen** für einen breiten Abnehmerkreis
- Lege fest, welche **Verpflichtungen (Verantwortlichkeiten)** übernommen werden sollen

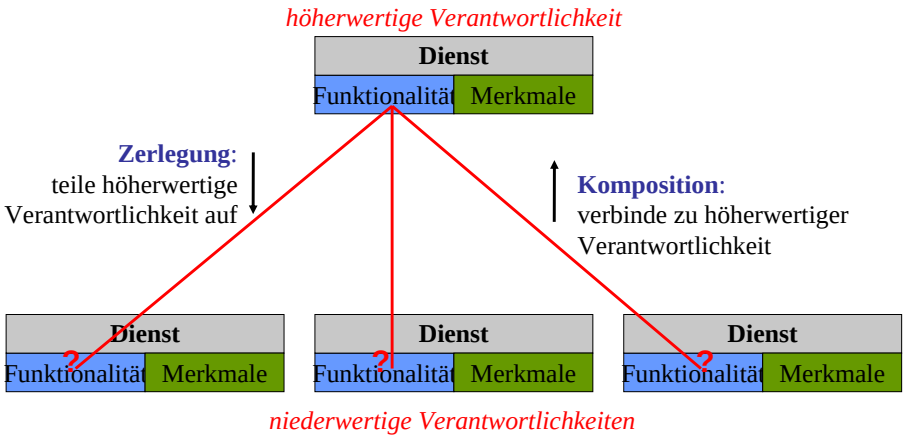
Entwurfsthese (1): Diensthierarchien

Grundlage: Teile-und-herrsche

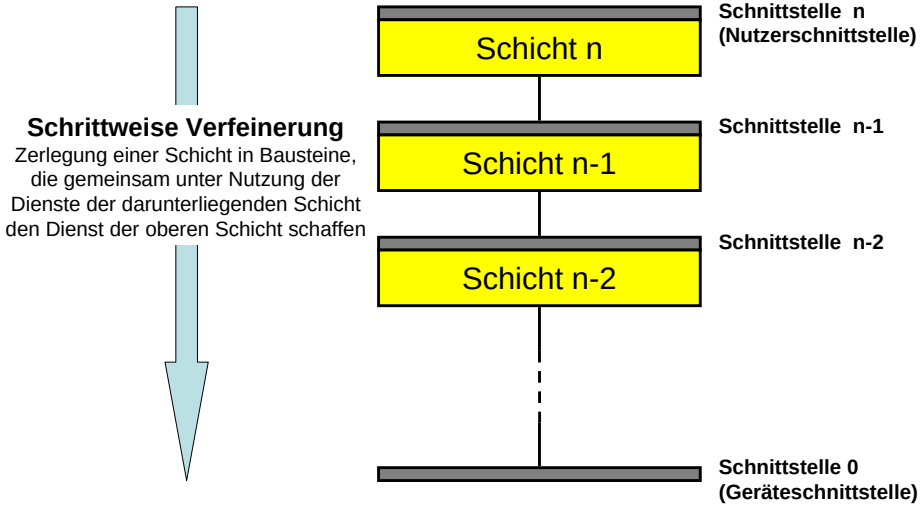


Entwurfsthese (2): Diensthierarchien

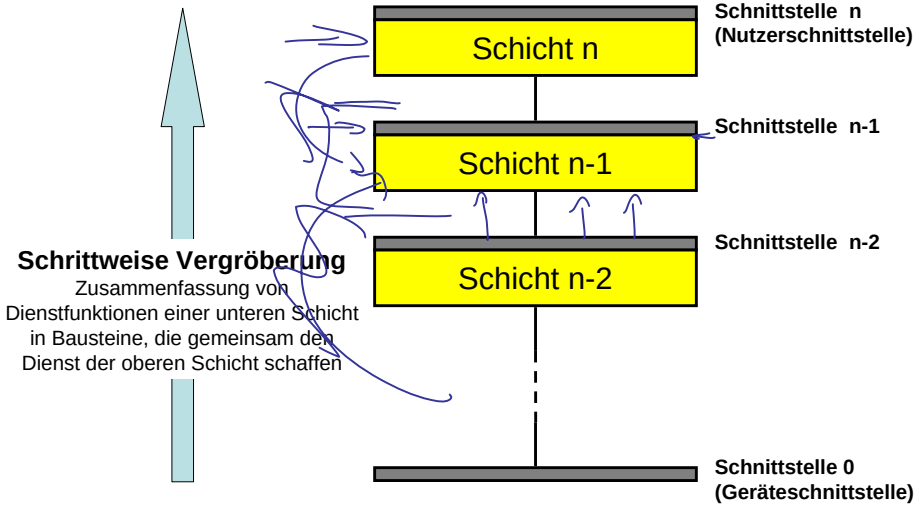
Funktionale Zerlegung



Entwurfsthese (3): Systemschichtung



Entwurfsthese (3): Systemschichtung



Korrektheit von Schichtenarchitekturen (1)

Grundsatz:
Sei D_i Dienst von Schicht *i*. D_{i-1} bildet die vollständige und alleinige Grundlage für die Realisierung von Dienst D_i.

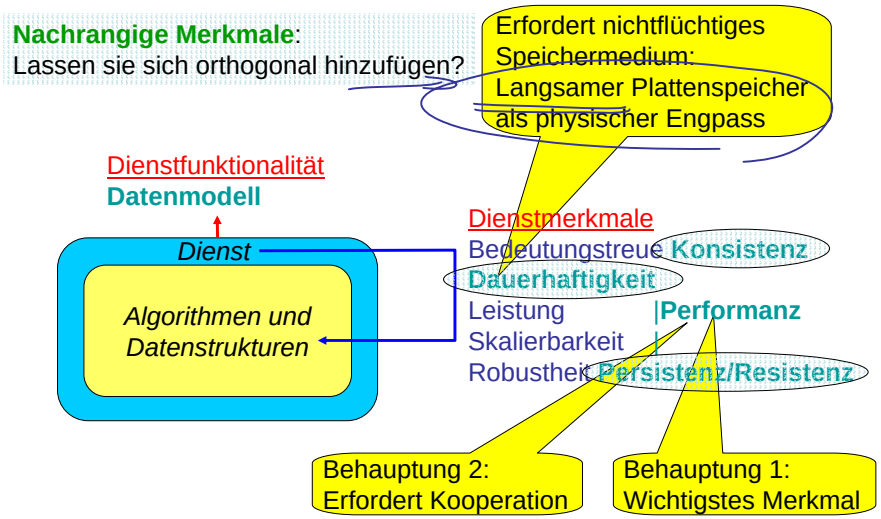
Begründung :

- Keine unkontrollierte Fortpflanzung der Änderungen in einer Schicht nach oben.
- Beweis der Korrektheit der Realisierung der Dienste D_i eines Verwalters M_i lokal führbar, weil man die Korrektheit von D_{i-1} unterstellen kann.

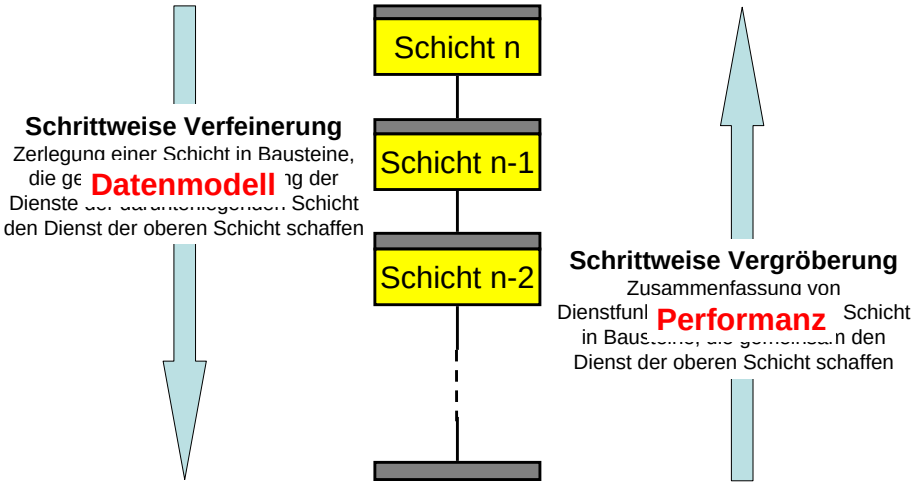
Kapitel 2: Referenzarchitektur für Datenbanksysteme

Methodischer Architekturentwurf
Architekturentwurf für Datenbanksysteme
Referenzarchitektur

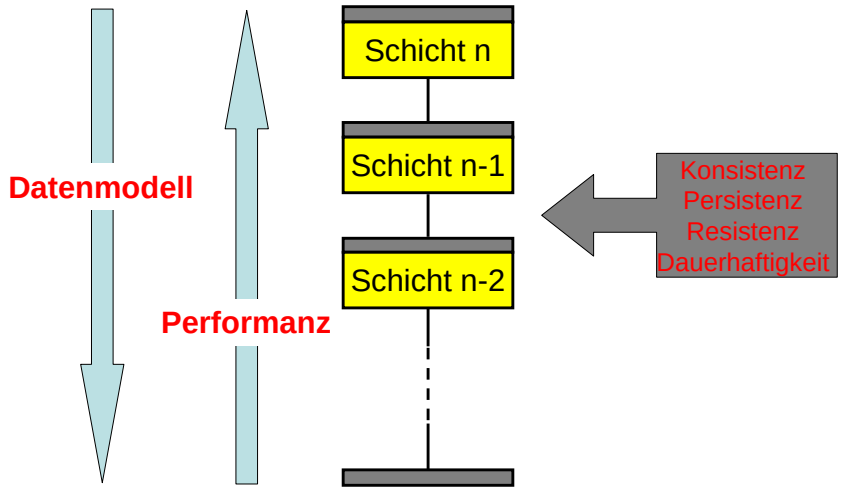
Ausgangspunkt der Schichtung



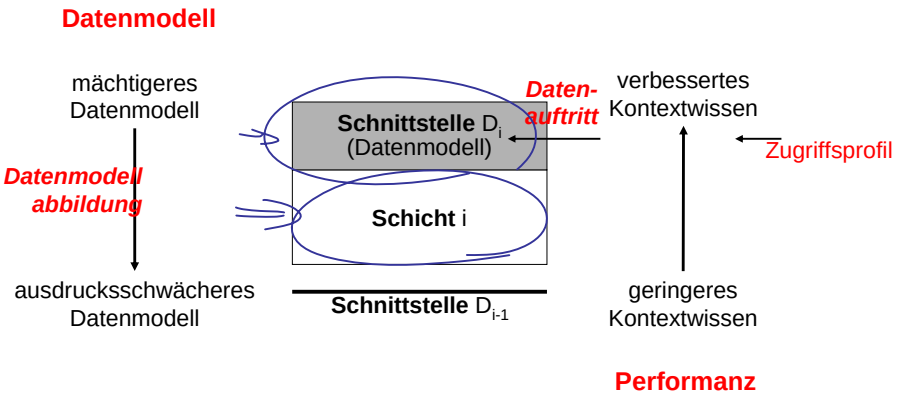
Entwurf von DBMS (1)



Entwurf von DBMS (2)



Entwurf einer Schicht



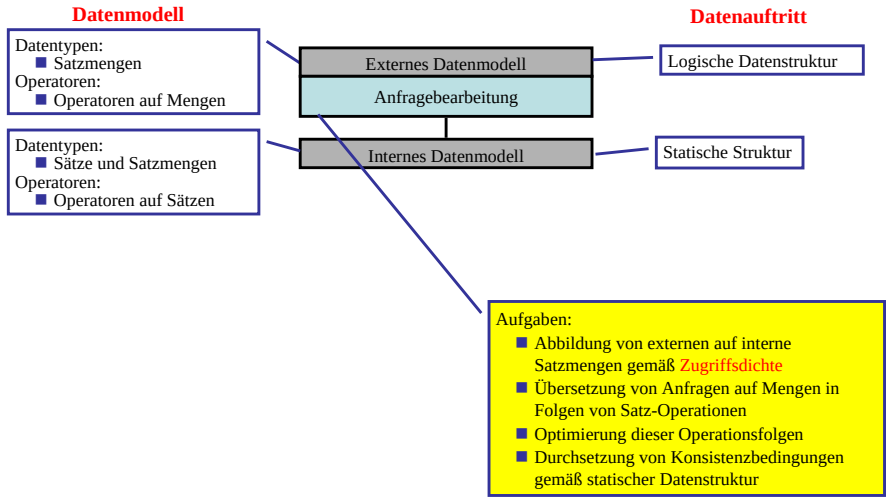
Kapitel 12: Referenzarchitektur für Datenbanksysteme

Methodischer Architekturentwurf

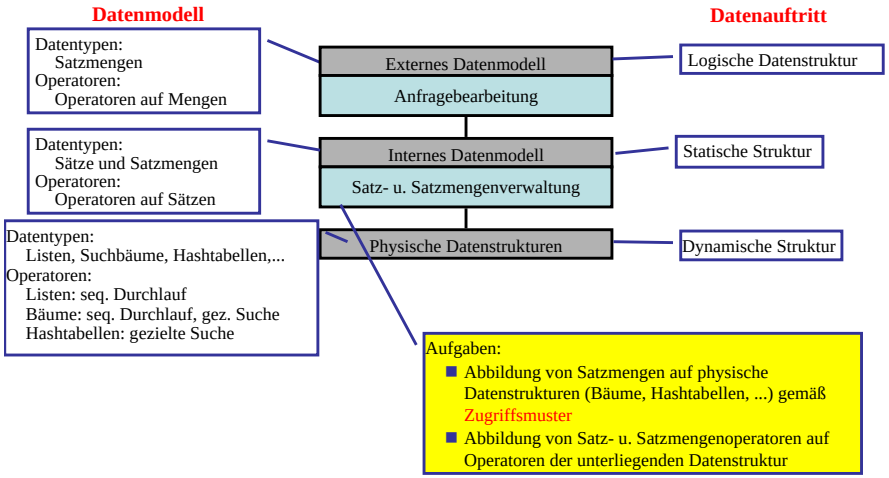
Architekturentwurf für Datenbanksysteme

Referenzarchitektur

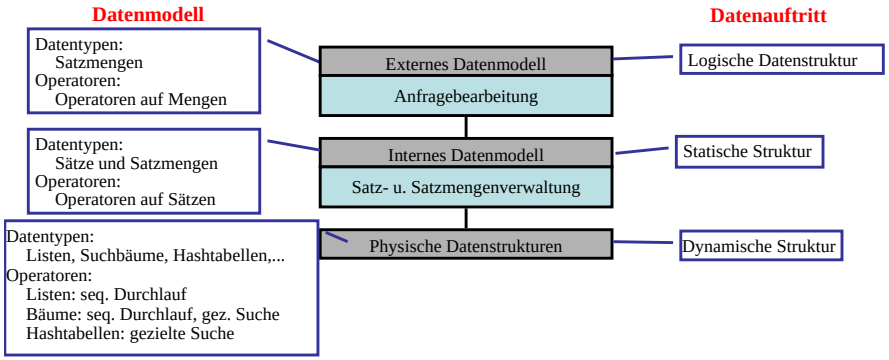
Datenbasis-Verwalter (1)



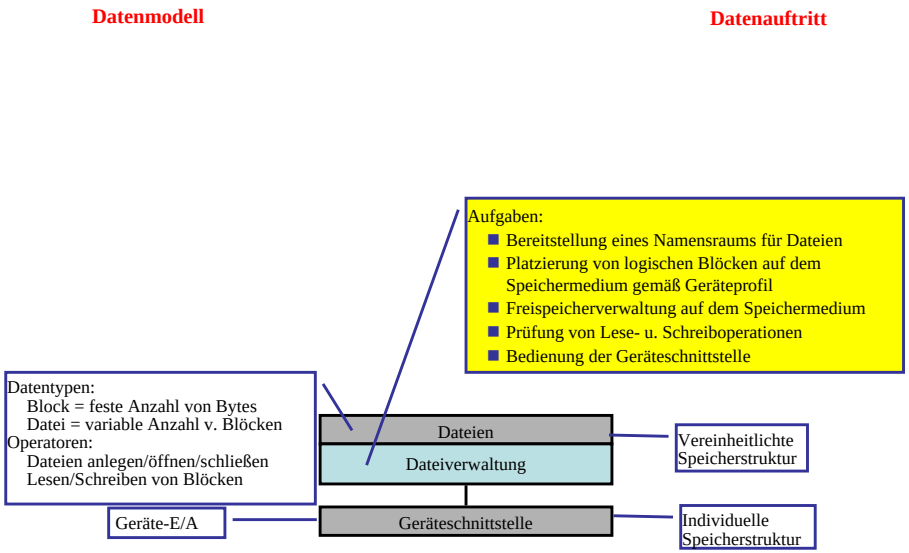
Datenbasis-Verwalter (2)



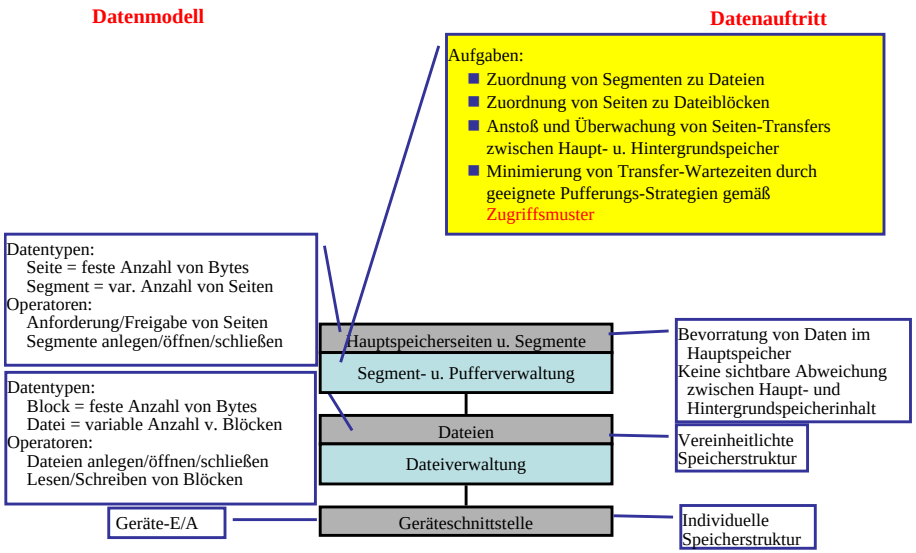
Datenbasis-Verwalter (2)



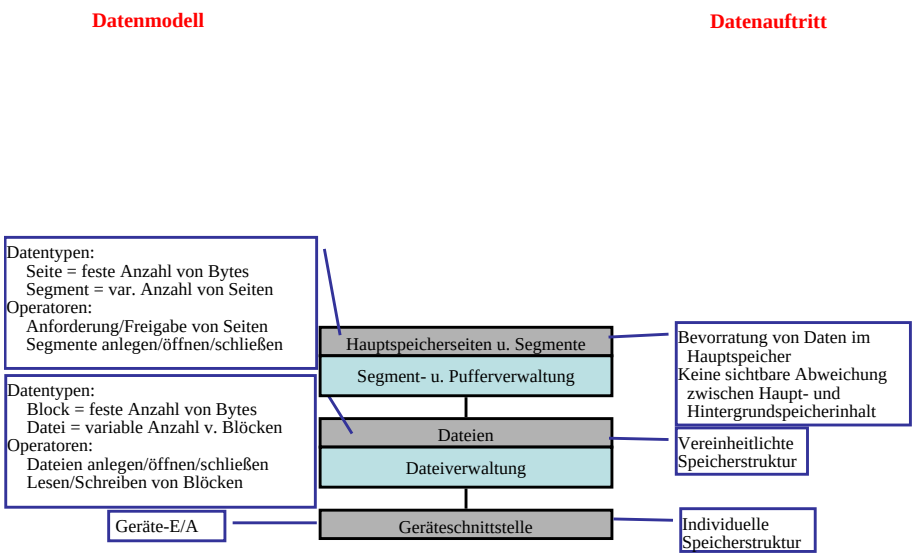
Datenbasis-Verwalter (3)



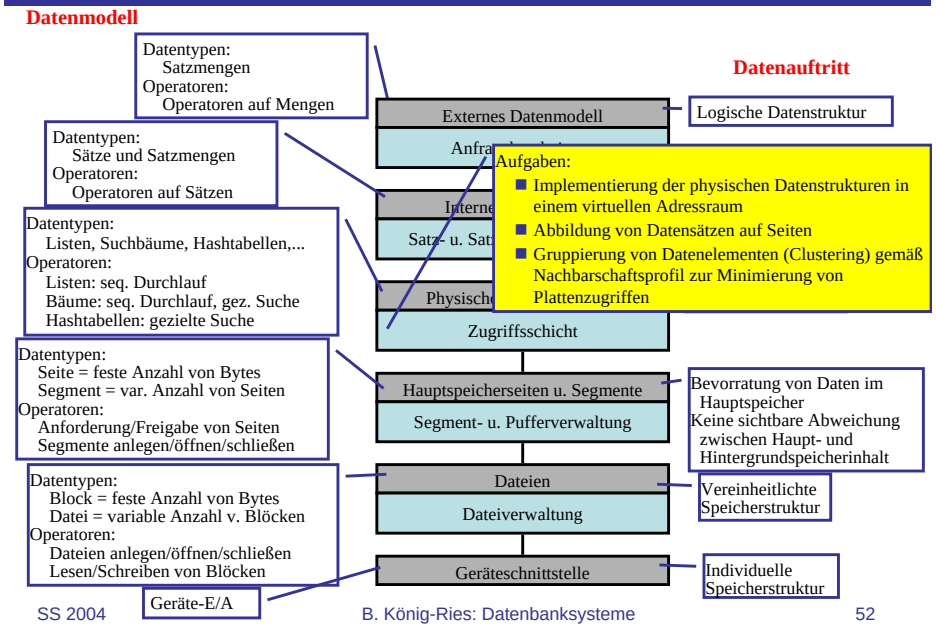
Datenbasis-Verwalter (4)



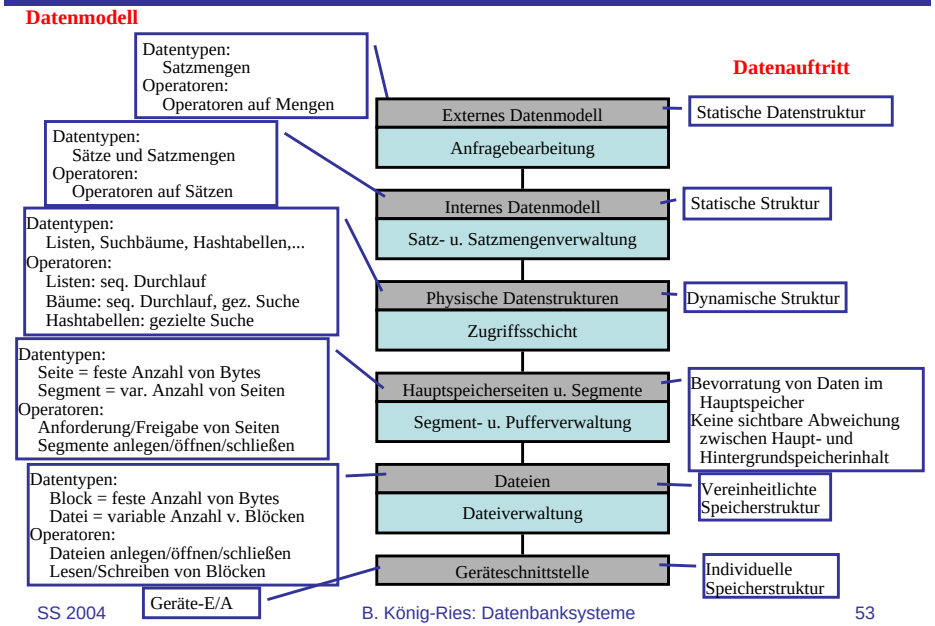
Datenbasis-Verwalter (4)



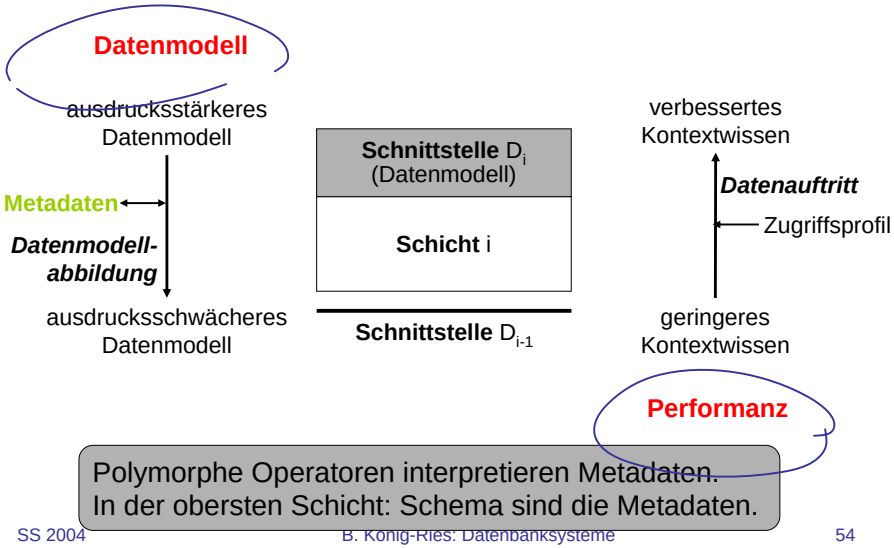
Datenbasis-Verwalter (5)



Datenbasis-Verwalter (5)



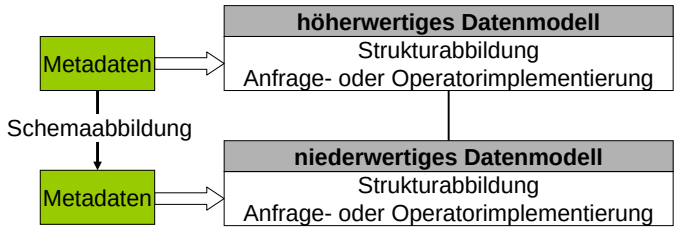
Schemakonsistenz (1)



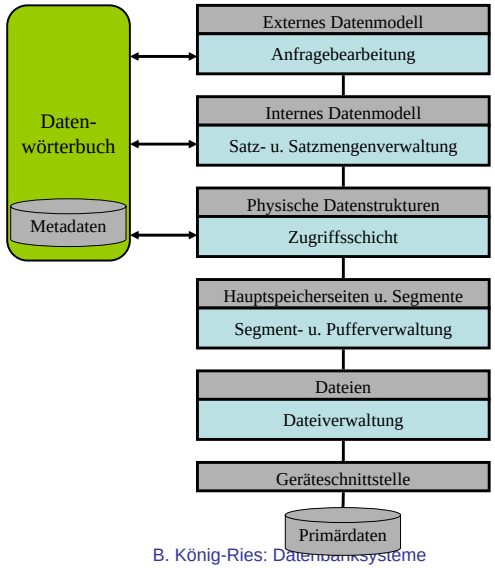
Polymorphe Operatoren interpretieren Metadaten.
In der obersten Schicht: Schema sind die Metadaten.

Schemakonsistenz (2)

Verallgemeinerung auf alle semantiktragenden Schichten



Schemakonsistenz (3)



Konsistenz, Persistenz, Resistenz (1)

- **Konsistenz**: Transaktion als Ausführung einer Transaktionsprozedur
- **Persistenz**: Herstellen der Dauerhaftigkeit erst bei erfolgreichem Transaktionsabschluss
- **Fehler-Resistenz**: Üblich: Zurücksetzen auf den Transaktionsanfang.
- **Konflikt-Resistenz**: Isolation der Transaktionen untereinander.

mach Fehlern ← **Recovery-Verwalter**

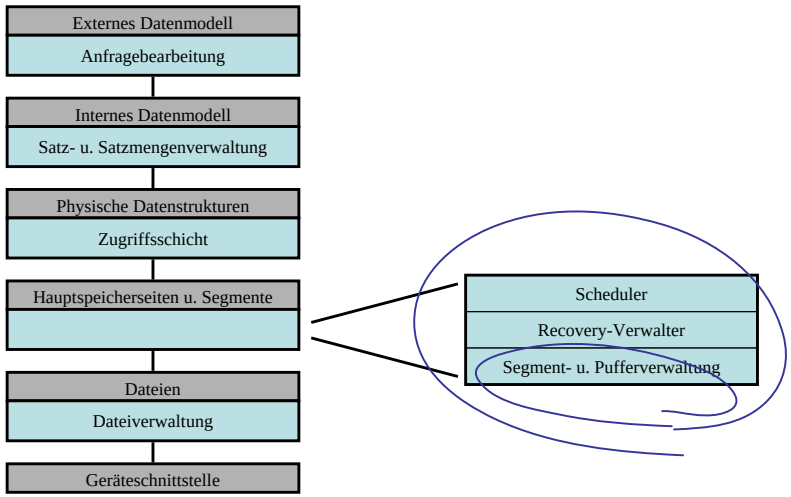
Isolation ← **Scheduler**

Konsistenz, Persistenz, Resistenz (2)

- Platzierung von Scheduler und Recovery-Verwalter:
  - Recovery-Verwalter benötigt Wissen um Transporte zwischen Haupt- und Hintergrundspeicher
  - Konsequenz: Integration mit Segmentverwaltung
  - Scheduler sollte mit denselben Einheiten wie der Recovery-Verwalter umgehen, daher Ansiedlung dort
- Transaktions-Koordinator:
  - Entgegennahme von Start-, Ende- und Abbruch-Anforderungen
  - Vergabe von Transaktionskennungen
  - Buchführung über Stand der Transaktion
  - Weiterreichen von Operationen mit Kennungen an Scheduler

→ extern oder intern

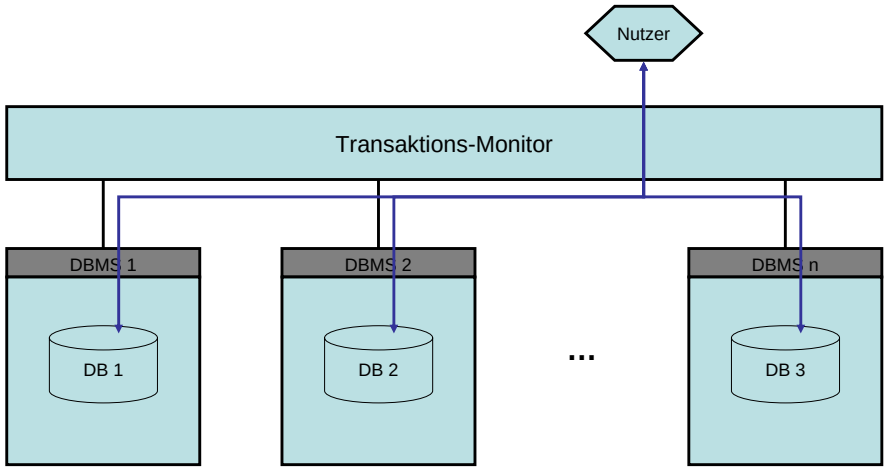
Konsistenz, Persistenz, Resistenz (3)



Konsistenz, Persistenz, Resistenz (4)

- Platzierung des Transaktions-Koordinators:
  - Variante 1: DBMS-intern
    - Integration mit Scheduler und Recovery-Manager
    - Transaktions-Beginn, -Ende und -Abbruch werden von Anfrageschicht durchgereicht
    - Nur Bearbeitung lokaler Datenbasis-Transaktionen möglich
  - Variante 2: Externes System (sog. **Transaktions-Monitor**):
    - Völlig eigenständiges System, vorgelagert zu DBMS
    - Kommunikation mit Recovery-Manager des DBMS über standardisierte Schnittstellen (**X/OPEN DTP**-Standard)
    - Bearbeitung von Anwendungs-Transaktionen möglich
    - Sinnvoll bei verteilten Informationssystemen

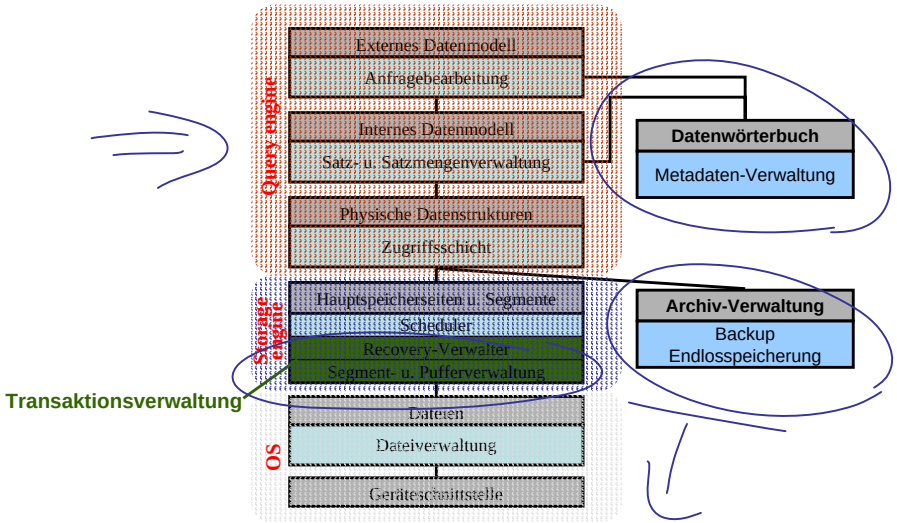
Konsistenz, Persistenz, Resistenz (5)



Dauerhaftigkeit

- Sichern der Persistenz nach Abschluss der Transaktion:
  - Kann offenkundig nicht Angelegenheit der Transaktionsverwaltung sein.
  - Darf normalen Betrieb nicht behindern.
  - Daher eigene Komponente: **Archiv-Verwaltung**

Hauptkomponenten eines DBMS



Kapitel 3: Das relationale Modell

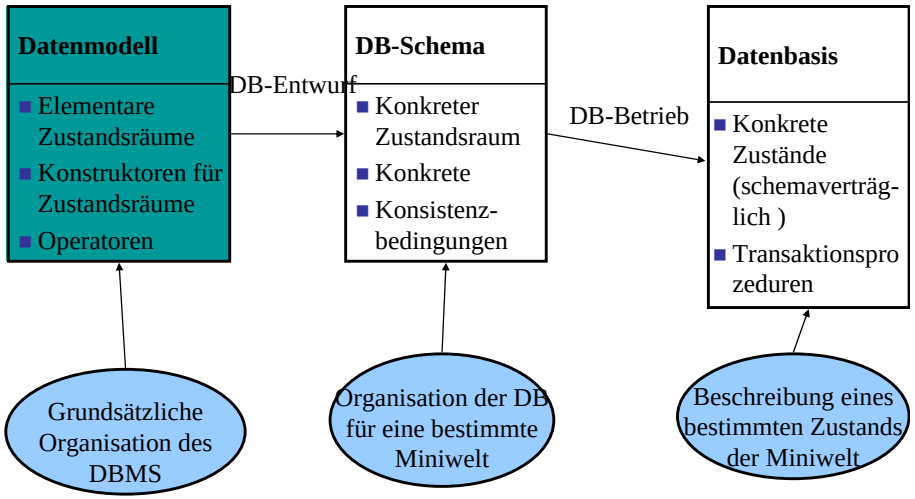
Einführung
Typsystem und Konsistenzbedingungen
relationale Algebra
Interaktive Anfragesprache SQL
Schema und Sichten
Sicherheit

viele Folien: © Prof. Lockemann, IPD, Uni Karlsruhe

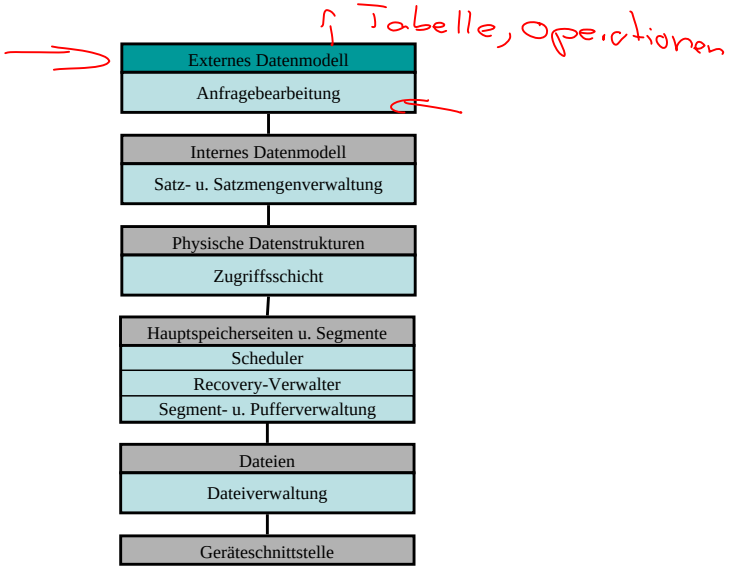
Kapitel 3: Das relationale Modell

viele Folien: © Prof. Lockemann, IPD, Uni Karlsruhe

Einordnung (1)



Einordnung (2)



Praxis der Datenmodelle

- Entwicklung eines Datenmodells verlangt Kompromisse:
  - Universelle Anwendbarkeit vs. Zuschnitt auf spezielle Modellierungsbedürfnisse
  - Ausdrucksmächtigkeit vs. Effizienz der Implementierung
- Heute eingesetzte Datenmodelle:
  - Hierarchisches und Netzwerk-Datenmodell (Altanwendungen)
  - Relationales Datenmodell (Sieg des Kompromisses)
  - Objektorientiertes Datenmodell (für anspruchsvollere Modellierungsbedürfnisse)
  - Objektrelationales Modell (Synthese der besten Eigenschaften)
  - XML (für Datenaustausch, Internetanwendungen)