1. Anforderungen und Beschreibungsmodelle

- **Nutzung von Dateisystemen**

- **Anforderungen an ein DBS**
  - Kontrolle über die operationalen Daten
  - Leichte Handhabung der Daten
  - Kontrolle der Datenintegrität
  - Hoher Grad an Datenunabhängigkeit

- **Schichtenmodelle für DBS**
  - Statisches Schichtenmodell: „Erklärungsmodell“
  - Schichtenweise Abbildungen
  - Rolle von n als Anzahl der Schichten
  - Dynamisches Verhalten

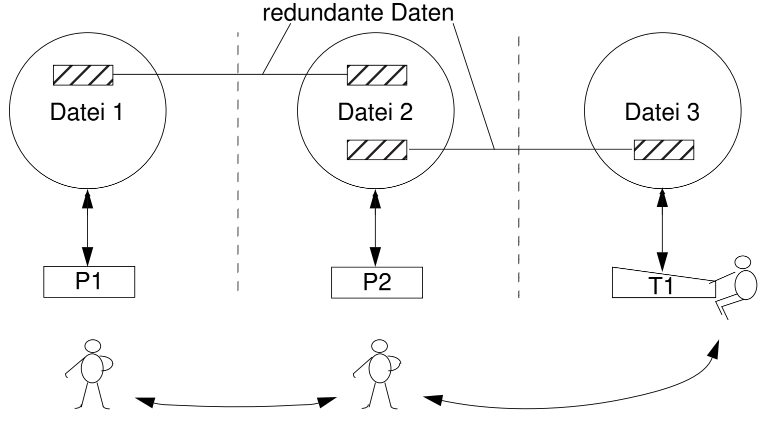
- **Drei-Schema-Architektur nach ANSI-SPARC**
  - Externes, konzeptionelles und internes Schema
  - Beschreibungsebenen für DB-Anwendungen

- **Dynamischer Ablauf von DBS-Operationen**

Nutzung von Dateisystemen

- **Permanente Datenhaltung innerhalb von BS-Dateien:**
  Warum keine direkte Nutzung von Dateien zur Datenhaltung in IS?

- **Betriebssystem/Dateisystem bietet Funktionen für**
  - Erzeugen / Löschen von Dateien
  - Zugriffsmöglichkeiten auf Blöcke/Sätze der Datei
  - einfache Operationen zum Lesen/Ändern/Einfügen/Löschen von Sätzen (dynamisches Wachstum)



Kommunikation notwendig für Änderungen

- **Probleme/Nachteile**
  - Datenredundanz und Inkonsistenz
  - Inflexibilität
  - **Mehrbenutzerbetrieb, Fehlerfall**
  - **Integritätssicherung**
  - Mißbrauch der Daten

➥ **Wer ist verantwortlich?**

- **DBS nutzt gewöhnlich Dateien** des Betriebssystems /Dateisystems zur permanenten Datenhaltung (**Externspeicherabstraktion**)

Anforderungen an ein DBS

1. Kontrolle über die operationalen Daten

- **Alle Daten können/müssen gemeinsam benutzt werden**
  - keine verstreuten privaten Dateien
  - Querauswertungen aufgrund inhaltlicher Zusammenhänge
  - symmetrische Organisationsformen
    (keine Bevorzugung einer Verarbeitungs- und Auswertungsrichtung)
  - Entwicklung neuer Anwendungen auf der existierenden DB
  - Erweiterung/Anpassung der DB (Änderung des Informationsbedarfs)

  ➥ ***Anwendungsneutralität beim DB-Entwurf*** **:**
  Was zeichnet ein gutes DB-Schema aus?

- **Eliminierung der Redundanz**
  - keine wiederholte Speicherung in unterschiedlicher Form
    für verschiedene Anwendungen
  - Vermeidung von Inkonsistenzen
  - zeitgerechter Änderungsdienst,
    keine unterschiedlichen Änderungsstände

  ➥ ***Redundanzfreiheit aus der Sicht der Anwendungen***

- **Datenbankadministrator (DBA):**
  zentrale Verantwortung für die operationalen Daten

Anforderungen an ein DBS (2)

2. Leichte Handhabbarkeit der Daten

- **Einfache Datenmodelle**
  - Beschreibung der logischen Aspekte der Daten
  - Benutzung der Daten ohne Bezug auf systemtechnische Realisierung

- **Logische Sicht der Anwendung**
  - zugeschnitten auf ihren Bedarf
  - lokale Sicht auf die DB

- **Leicht erlernbare Sprachen**
  - deskriptive Problemformulierung
  - hohe Auswahlmächtigkeit
  - Unterstützung der Problemlösung des Anwenders im Dialog

- **Durchsetzung von Standards**
  - unterschiedliche DBS bieten einheitliche Schnittstelle
  - Portierbarkeit von Anwendungen
  - erleichterter Datenaustausch

- **Erweiterung der Benutzerklassen**
  - Systempersonal
  - Anwendungsprogrammierer
  - Anspruchsvolle Laien
  - Parametrische Benutzer/Gelegentliche Benutzer

Relationenmodell - Beispiel

DB-Schema

FB

FBNR	FBNAME	DEKAN

PROF

PNR	PNAME	FBNR	FACHGEB

STUDENT

MATNR	SNAME	FBNR	STUDBEG

PRÜFUNG

PNR	MATNR	FACH	DATUM	NOTE

Ausprägungen

FB	FBNR	FBNAME	DEKAN
	FB 9	WIRTSCHAFTSWISS	4711
	FB 5	INFORMATIK	2223

PROF	PNR	PNAME	FBNR	FACHGEB
	1234	HÄRDER	FB 5	DATENBANKSYSTEME
	5678	WEDEKIND	FB 9	INFORMATIONSSYSTEME
	4711	MÜLLER	FB 9	OPERATIONS RESEARCH
	6780	NEHMER	FB 5	BETRIEBSSYSTEME

STUDENT	MATNR	SNAME	FBNR	STUDBEG
	123 766	COY	FB 9	1.10.95
	225 332	MÜLLER	FB 5	15. 4.87
	654 711	ABEL	FB 5	15.10.94
	226 302	SCHULZE	FB 9	1.10.95
	196 481	MAIER	FB 5	23.10.95
	130 680	SCHMID	FB 9	1. 4.97

PRÜFUNG	PNR	MATNR	FACH	PDATUM	NOTE
	5678	123 766	BWL	22.10.97	4
	4711	123 766	OR	16. 1.98	3
	1234	654 711	DV	17. 4.97	2
	1234	123 766	DV	17. 4.97	4
	6780	654 711	SP	19. 9.97	2
	1234	196 481	DV	15.10.97	1
	6780	196 481	BS	23.12.97	3

Relationenmodell - Beispiel (2)

- **Deskriptive DB-Sprachen**
  - hohes Auswahlvermögen und Mengenorientierung
  - leichte Erlernbarkeit auch für den DV-Laien
  - RM ist symmetrisches Datenmodell, d.h., es gibt keine bevorzugte Zugriffs- oder Auswertungsrichtung

- **Anfragebeispiele:**

Q1: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

```
SELECT    *
FROM      STUDENT
WHERE     FBNR = 'FB5' AND STUDBEG < '1.1.90'
```

Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

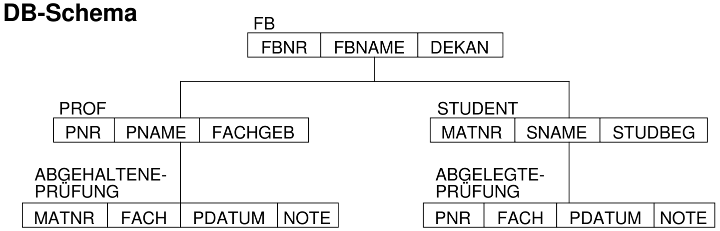
```
SELECT    *
FROM      STUDENT
WHERE     FBNR = 'FB5' AND MATNR IN
          (SELECT   MATNR
           FROM     PRÜFUNG
           WHERE    FACH = 'DV' AND NOTE ≤ '2')
```

Q3: Finde die Durchschnittsnoten der DV-Prüfungen für alle Fachbereiche mit mehr als 1000 Studenten.

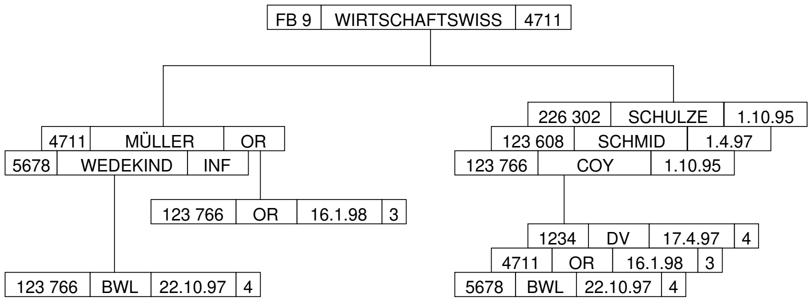
```
SELECT    F.FBNR, AVG (P.NOTE)
FROM      PRÜFUNG P, STUDENT F
WHERE     P.FACH = 'DV' AND P.MATNR = F.MATNR
GROUP BY  F.FBNR
HAVING    (SELECT   COUNT(*)
           FROM     STUDENT S
           WHERE    S.FBNR = F.FBNR) > 1000
```

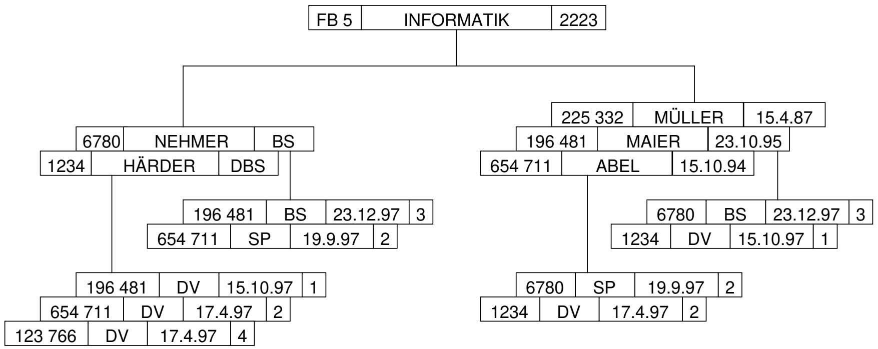
Hierarchisches Datenmodell - Beispiel

DB-Schema



Ausprägungen





Hierarchisches Datenmodell - Beispiel (2)

- **Prozedurale DB-Sprachen**
  - Programmierer als Navigator
  - satzweiser Zugriff über vorhandene Zugriffspfade
  - bei hierarchischen Datenmodellen
    - unnatürliche Organisation bei komplexen Beziehungen (n:m)
    - Auswertungsrichtung ist vorgegeben

- **Anfragebeispiele:**

Q4: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 1990 begonnen haben.

```
              GU FB (FBNR = 'FB5');
NEXT_STUDENT: GNP STUDENT (STUDBEG < '1.1.90');
              IF END_OF_PARENT THEN EXIT;
              PRINT STUDENT RECORD;
              GOTO NEXT_STUDENT;
```

Q5: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
              GU FB (FBNR = 'FB5');
NEXT_STUDENT: GNP STUDENT;
              IF END_OF_PARENT THEN EXIT;
              GNP ABGELEGTE_PRÜFUNG   (FACH = 'DV') AND
                                      (NOTE ≤ '2');
              IF END_OF_PARENT THEN GOTO NEXT_STUDENT;
              PRINT STUDENT RECORD;
              GOTO NEXT_STUDENT;
```

Anforderungen an ein DBS (3)

3. Kontrolle der Datenintegrität

(system enforced integrity, siehe auch ACID-Eigenschaften[1])

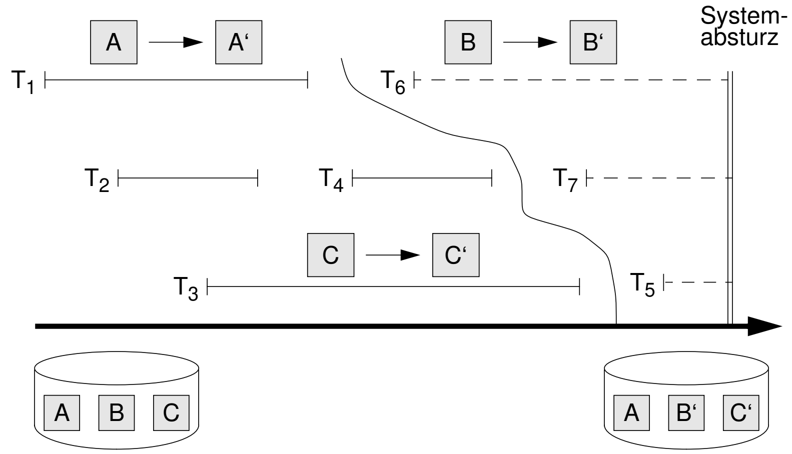
- **Automatisierte Zugriffskontrollen (Datenschutz)**
  - separat für jedes Datenobjekt
  - unterschiedliche Rechte für verschiedene Arten des Zugriffs
  - **Idealziel**: „least priviledge principle“
- **Erhaltung der logischen Datenintegrität**
  - Beschreibung der „Richtigkeit“ von Daten durch Prädikate und Regeln
  - „Qualitätskontrollen“ bei Änderungsoperationen
  - aktive Maßnahmen des DBS erwünscht (ECA-Regeln)
- **Erhaltung der physischen Datenintegrität**
  - Periodisches Erstellen von Datenkopien
  - Führen von Änderungsprotokollen für den Fehlerfall (Logging)
  - Bereitstellen von Wiederherstellungsalgorithmen im Fehlerfall (Recovery)

    ➥ **Garantie** nach erfolgreichem Neustart:
    jüngster transaktionskonsistenter DB-Zustand
- **Notwendigkeit des kontrollierten Mehrbenutzerbetriebs**
  - logischer Einbenutzerbetrieb für jeden von n parallelen Benutzern (Leser + Schreiber)
  - geeignete Synchronisationsmaßnahmen zur gegenseitigen Isolation
  - angepaßte Synchronisationseinheiten (z. B. Sperrgranulate) mit abgestuften Zugriffsrechten

    ➥ **Ziel**: möglichst geringe gegenseitige Behinderung

1. „May all your transactions commit and never leave you in doubt“ (J. Gray)

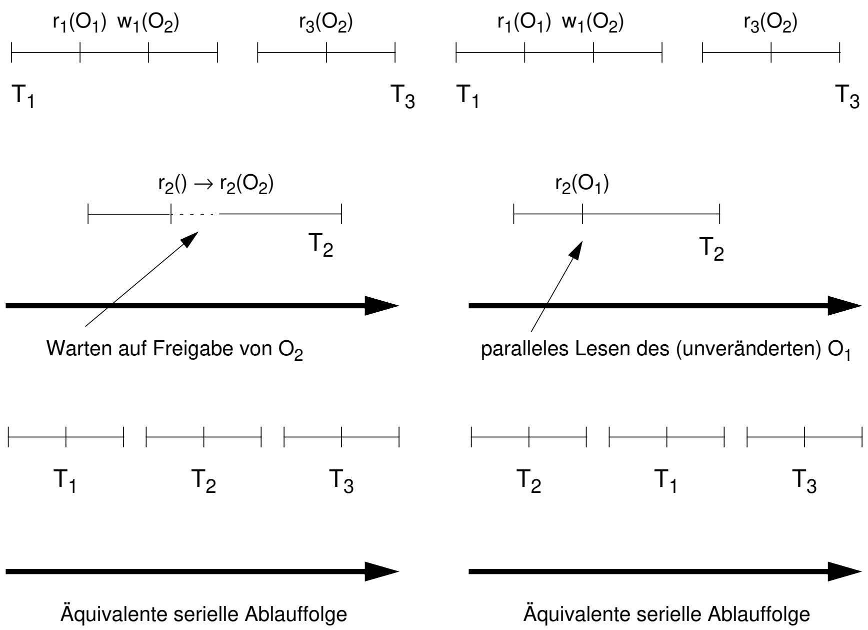
Physische Datenintegrität - jüngster transaktionskonsistenter Zustand



- **DBMS garantiert physische Datenintegrität**
  - bei jedem Fehler (z. B. Ausfall des Rechners, Absturz des Betriebssystems oder des DBMS, Fehlerhaftigkeit einzelner Transaktionsprogramme) wird eine „korrekte“ Datenbank rekonstruiert
  - nach einem (Teil-)Absturz ist immer der jüngste transaktionskonsistente Zustand der DB zu rekonstruieren, in dem alle Änderungen von Transaktionen enthalten sind, die vor dem Zeitpunkt des Fehlers erfolgreich beendet waren (T_1 bis T_4) und sonst keine
  - automatische Wiederherstellung nach Neustart des Systems
- **Maßnahmen beim Wiederanlauf (siehe auch Beispiel)**
  - Ermittlung der beim Absturz aktiven Transaktionen (T_5, T_6, T_7)
  - Rücksetzen (UNDO) der Änderungen der aktiven Transaktionen in der Datenbank (B‘ → B)
  - Wiederholen (REDO) der Änderungen von abgeschlossenen Transaktionen, die vor dem Absturz nicht in die Datenbank zurückgeschrieben waren (A → A‘)

Logischer Einbenutzerbetrieb

- Beim **logischen Einbenutzerbetrieb** hat jede der parallel aktiven Transaktionen den Eindruck, als liefe sie alleine ab, d. h., logisch bilden alle Transaktionen eine serielle Ablauffolge
- **Synchronisationskomponente** des DBMS umfaßt alle Maßnahmen zur Sicherstellung der Ablaufintegrität (Isolation der parallelen Transaktionen)
- **Formale Definition:** Eine parallele Ablauffolge von Transaktionen ist genau dann korrekt synchronisiert, wenn es eine zu dieser Ablauffolge äquivalente (bezüglich ihrer Lese- und Schreibabhängigkeiten (r, w)) serielle Ablauffolge gibt, so daß jede Transaktion T_i in der seriellen Reihenfolge dieselben Werte liest und schreibt wie im parallelen Ablauf. (Dabei ist jede Permutation der T_i-Folge gleichermaßen zulässig, siehe Beispiel).



Anforderungen an ein DBS (4)

4. Hoher Grad an Datenunabhängigkeit

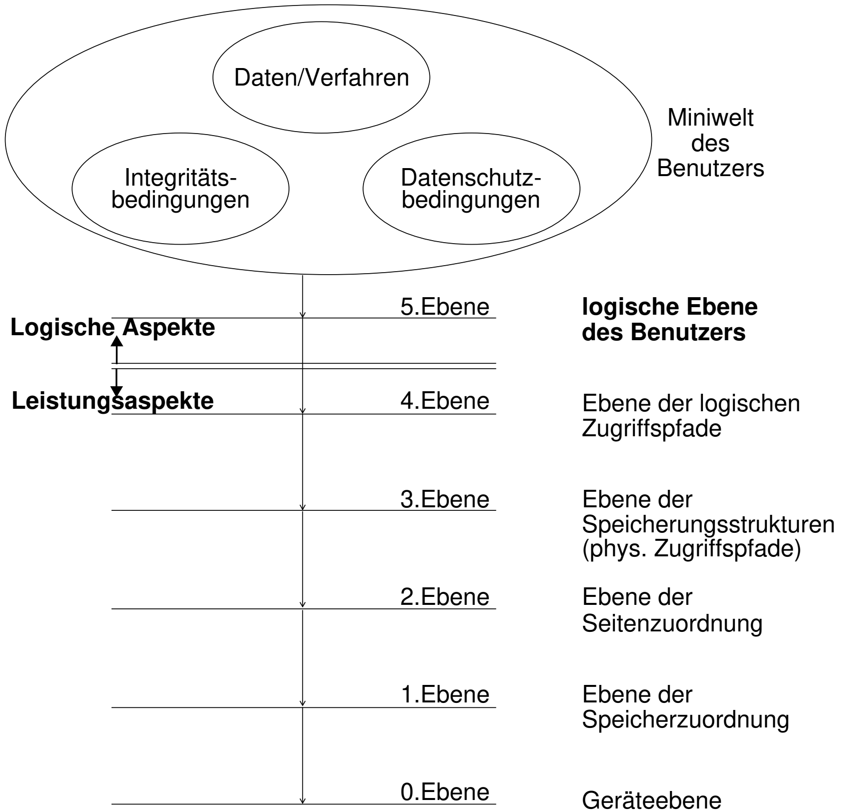
- **Konventionelle Anwendungsprogramme (AP) mit Dateizugriff**
  - Nutzung von Kenntnissen der Datenorganisation und Zugriffstechnik
  - gutes Leistungsverhalten, aber . . . ?
- **Datenabhängige Anwendungen sind äußerst unerwünscht**
  - verschiedene Anwendungen brauchen verschiedene Sichten auf dieselben Daten
  - Änderungen im Informationsbedarf sowie bei Leistungsanforderungen erzwingen Anpassungen bei Speicherungsstrukturen und Zugriffsstrategien

    ➥ deshalb: *möglichst starke Isolation der APs von den Daten*
    sonst: extremer Wartungsaufwand für die APs
- **Realisierung verschiedener Arten von *Datenunabhängigkeit*:**
  - **Geräte**unabhängigkeit
  - **Speicherungsstruktur**unabhängigkeit

    ➥ **Minimalziel:** physische Datenunabhängigkeit (durch das BS/DBS)
  - **Zugriffspfad**unabhängigkeit
  - **Datenstruktur**unabhängigkeit

    ➥ logische Datenunabhängigkeit
    (vor allem durch das Datenmodell!)

Ebenen beim Entwurf eines DBS[1]



- **„outside-in"-Ansatz** (*top-down*)
  - Die Ausdrucksmächtigkeit des Datenmodells und seine Konzepte sowie die postulierten Betriebseigenschaften bestimmen die Anforderungen, die an das zu entwerfende DBS zu stellen sind
  - Beim Entwurf erfolgt eine mehrstufige Strukturverfeinerung, bis die konkrete Implementierungsstruktur abgeleitet ist

1. „Eine Hauptaufgabe der Informatik ist systematische Abstraktion" (H. Wedekind)

Verschiedene Sichten auf DBS-Daten

- **Logischen Datenstrukturen eines Anwendungsbeispiels**

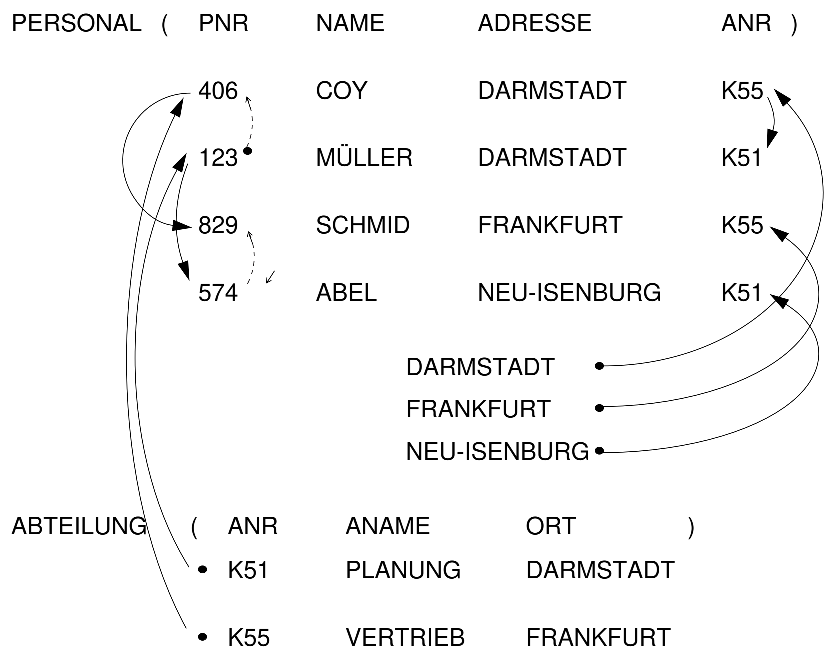
PERSONAL (PNR NAME ADRESSE ANR)

PNR	NAME	ADRESSE	ANR
406	COY	DARMSTADT	K55
123	MÜLLER	DARMSTADT	K51
829	SCHMID	FRANKFURT	K55
574	ABEL	NEU-ISENBURG	K51

ABTEILUNG (ANR ANAME ORT)

ANR	ANAME	ORT
K51	PLANUNG	DARMSTADT
K55	VERTRIEB	FRANKFURT

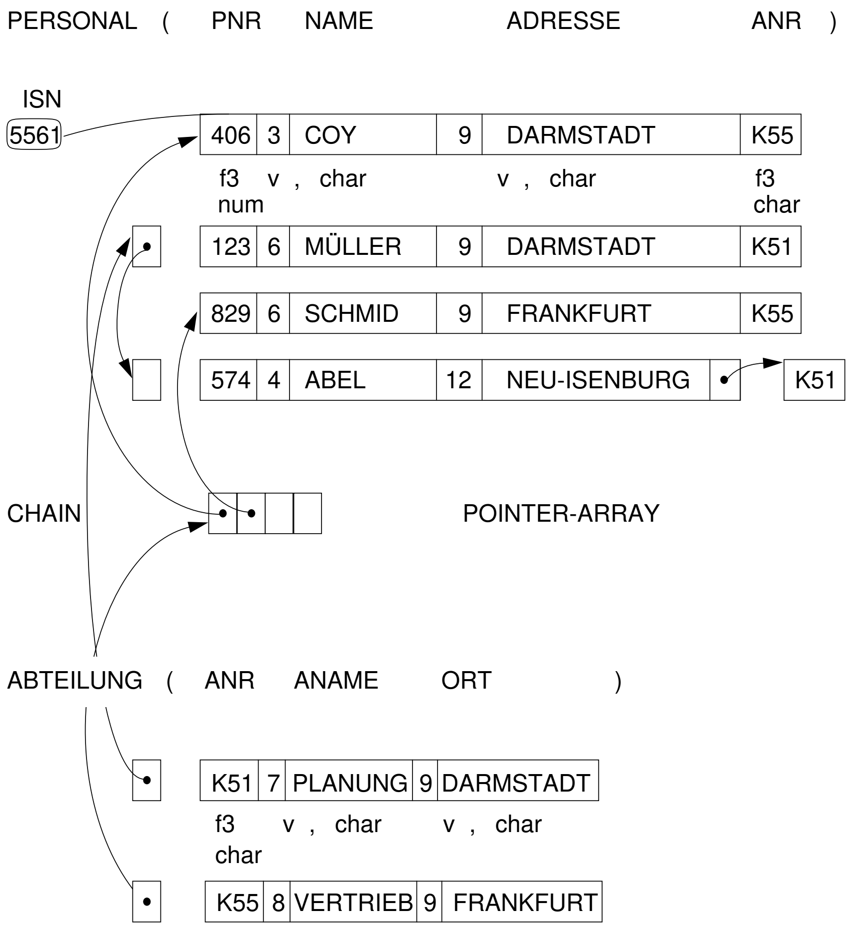
- **Sicht auf die logischen Zugriffspfade**



Logische Zugriffspfade: 1. OWNER - MEMBER
2. Sortierreihenfolge PNR ASC
3. Search Key (Invertierung ADRESSE)

Verschiedene Sichten auf DBS-Daten (2)

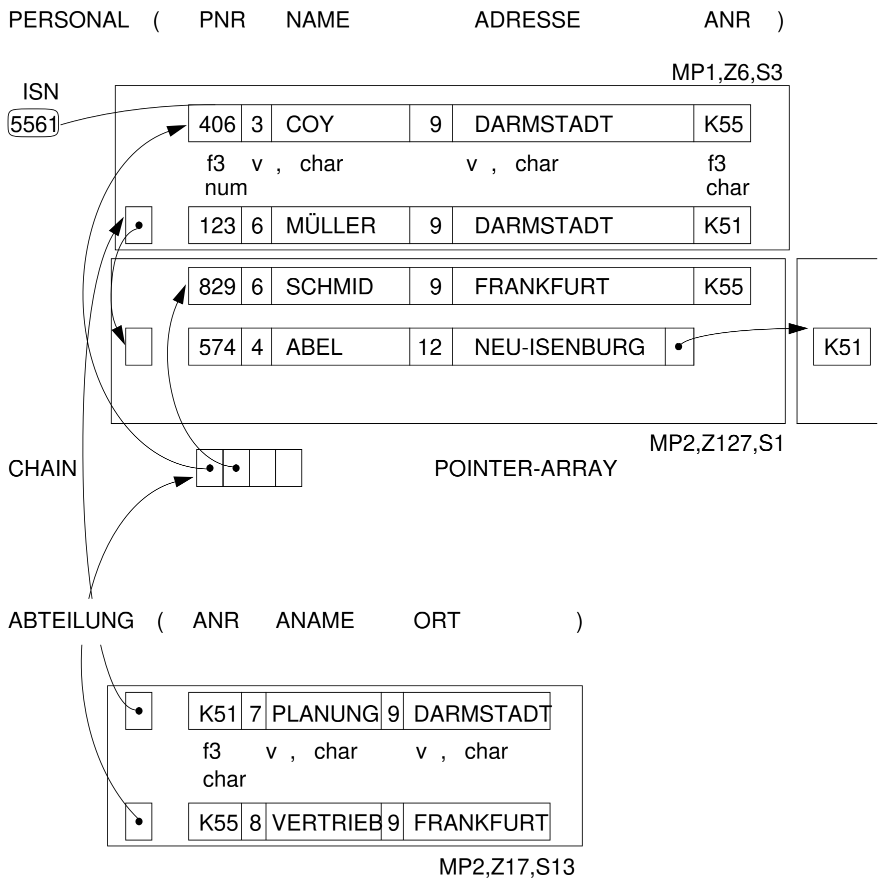
- **Sicht auf die Speicherungsstrukturen**



Speicherungsstrukturen: 1. Formate
2. Datentypen
3. Implementierungstechniken

Verschiedene Sichten auf DBS-Daten (3)

- **Sicht auf die Speicherzuordnungsstrukturen**



Speicherzuordnungsstrukturen: 1. physische Blocklänge
2. spanned record facility

Gerätemerkmale: 1. Eigenschaften der Speichermedien
2. Magnetplatten-Zuordnungen

Schichtenmodelle für DBS

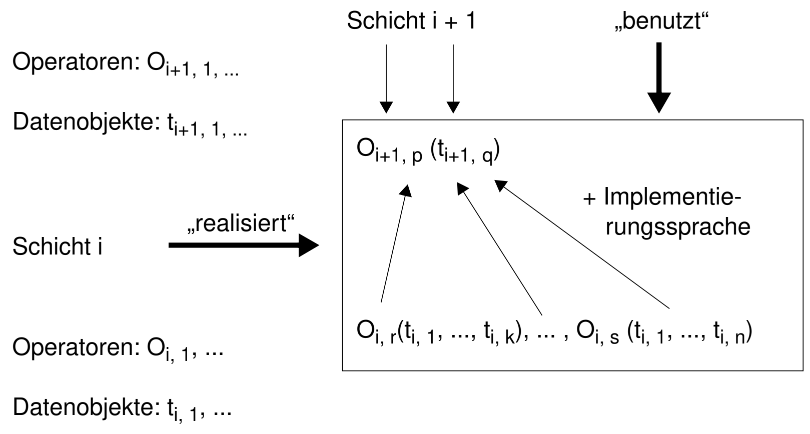
- **Ziel: Architektur eines datenunabhängigen DBS**
- **Systementwurf**
  - Was sind die geeigneten Beschreibungs- und Kommunikationstechniken?
    Sie sind notwendigerweise informal.
  - Was ist auf welcher Beschreibungsebene sichtbar?
    Es ist angemessene Abstraktion erforderlich![1]
  - Wie kann eine Evolution des Systems erfolgen?
    Es muß eine Kontrolle der Abhängigkeiten erfolgen!
- **Aufbau in Schichten:**
  - „günstige Zerlegung" des DBS in „nicht beliebig viele" Schichten
  - optimale Bedienung der Aufgaben der darüberliegenden Schicht
  - implementierungsunabhängige Beschreibung der Schnittstellen

  ➥ Es gibt keine Architekturlehre für den Aufbau großer SW-Systeme
- **Empfohlene Konzepte:**
  - Geheimnisprinzip (*Information Hiding*)
  - hierarchische Strukturierung
  - generische Auslegung der Schnittstellen:
    Nur bestimmte Objekttypen mit charakteristischen Operationen sind vorgegeben, jedoch nicht ihre anwendungsbezogene Spezifikation und Semantik

1. „Die durch Abstraktion entstandenen Konstrukte der Informatik als Bedingungen möglicher Information sind zugleich die Bedingungen der möglichen Gegenstände der Information in den Anwendungen"
(H. Wedekind in Anlehnung an eine Aussage Kants aus der „Kritik der reinen Vernunft")
Vereinfacht ausgedrückt: Informatiker erfinden (konstruieren) abstrakte Konzepte; diese ermöglichen (oder begrenzen) wiederum die spezifischen Anwendungen.

Schichtenmodelle für DBS (2)[1]

- **Aufbauprinzip:**



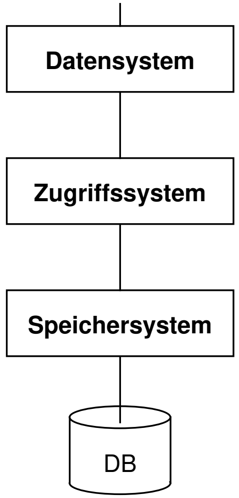
- **„benutzt"-Relation:**
  A benutzt B, wenn A B aufruft und die korrekte Ausführung von B für die vollständige Ausführung von A notwendig ist
- **Anzahl der Schichten**
  - n = ?
  - Entwurfskomplexität/Schicht fällt mit wachsendem n
  - Laufzeitaufwand des DBS steigt mit wachsendem n

1. Härder, T., Rahm, E.: Datenbanksysteme - Konzepte und Techniken der Implementierung, Springer-Verlag, 1999, Kap. 1

Schichtenmodelle für DBS (3)

- **Vereinfachtes Schichtenmodell**

Aufgaben der Systemschicht		**Art der Operationen an der Schnittstelle**
		deskriptive Anfragen Zugriff auf Satzmengen
Übersetzung und Optimierung von Anfragen	Datensystem	
		Satzzugriffe
Verwaltung von physischen Sätzen und Zugriffspfaden	Zugriffssystem	
		Seitenzugriffe
DB-Puffer- und Externspeicher-Verwaltung	Speichersystem	

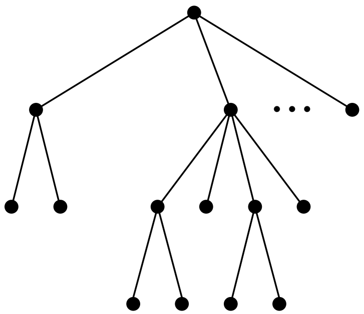


- **Dynamischer Kontrollfluß einer Operation an das DBS**

Datensystem

Zugriffssystem

Speichersystem



DBS-Operationen (API)

Füge Satz ein
Modifiziere Zugriffspfad

Stelle Seite bereit
Gib Seite frei

Lies / Schreibe Seite

Schichtenmodelle für DBS (4)

- Vorteile als Konsequenzen der Nutzung **hierarchischer Strukturen und der „benutzt"-Relation**
  - Höhere Ebenen (Systemkomponenten) werden einfacher, weil sie tiefere Ebenen (Systemkomponenten) benutzen können
  - Änderungen auf höheren Ebenen sind ohne Einfluß auf tieferen Ebenen
  - Höhere Ebenen können abgetrennt werden, tiefere Ebenen bleiben trotzdem funktionsfähig
  - Tiefere Ebenen können getestet werden, bevor die höheren Ebenen lauffähig sind

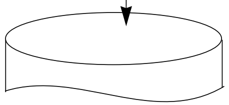
- **Jede Hierarchieebene kann als abstrakte oder virtuelle Maschine aufgefaßt werden**
  - Programme der Schicht i benutzen als abstrakte Maschine die Programme der Schicht i-1, die als Basismaschine dienen
  - Abstrakte Maschine der Schicht i dient wiederum als Basismaschine für die Implementierung 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

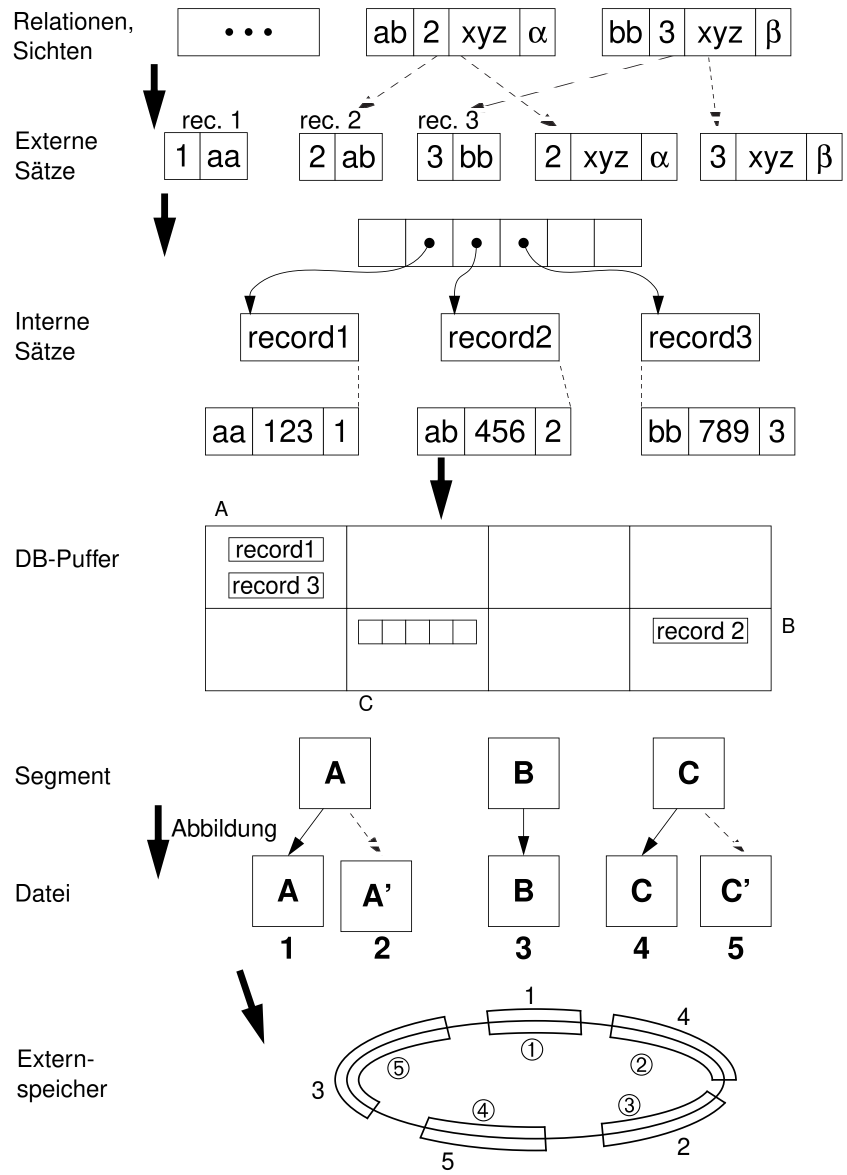
- Programme einer bestimmten Schicht können die der nächsten tieferen Schicht genau so benutzen**, als sei die untere Schicht Hardware**

Fünf-Schichten-Modell eines DBS - statisch

	Adressierungseinheiten: Relationen, Sichten, Tupel	Transaktionsprogramme
Mengenorientierte DB-Schnittstelle		
Sprachen wie SQL, QBE usw.	Adressierungseinheiten: Relationen, Sichten, Tupel Hilfsstrukturen: externe Schemabeschreibung, Integritätsregeln Adressierungseinheiten: externe Sätze, Sets, Schlüssel, Zugriffspfade	Logische Datenstrukturen
Satzorientierte DB-Schnittstelle		
FIND NEXT Satzname, STORE Satzname, usw.	Adressierungseinheiten: externe Sätze, Sets, Schlüssel, Zugriffspfade Hilfsstrukturen: Zugriffspfaddaten, interne Schemabeschreibung Adressierungseinheiten: interne Sätze, B*-Bäume, Hash-Tabellen, usw.	Logische Zugriffspfadstrukturen
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. Adressierungseinheiten: Seiten, Segmente	Speicherungsstrukturen
DB-Pufferschnittstelle		
Stelle Seite j bereit, Gib Seite j frei	Adressierungseinheiten: Seiten, Segmente Hilfsstrukturen: Seitentabellen, Blocktabellen, usw. Adressierungseinheiten: Blöcke, Dateien	Seitenzuordnungsstrukturen
Dateischnittstelle		
Lies Block k, Schreibe Block k	Adressierungseinheiten: Blöcke, Dateien Hilfsstrukturen: Freispeicher-Info., Extent-Tabellen, Dateikataloge, usw. Adressierungseinheiten: Spuren, Zylinder, Kanäle, usw.	Speicherzuordnungsstrukturen
Geräteschnittstelle		
Kanalprogramme		Externe Speichermedien



Fünf-Schichten-Modell - schichtenweise Abbildungen



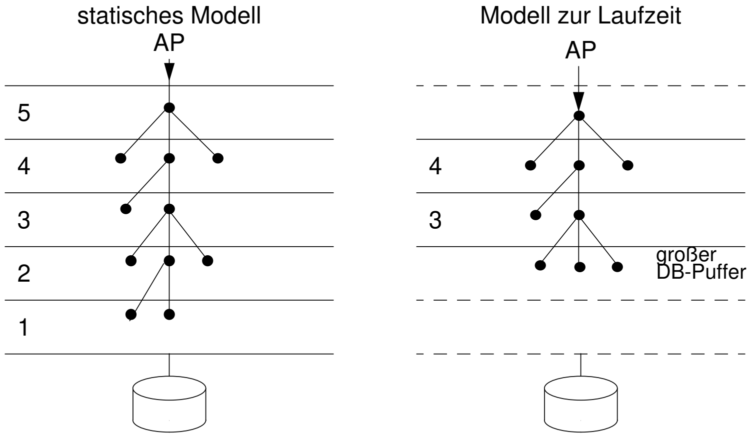
n-Schichtenmodell - Rolle von n

- **n = 1: Monolith**

- **Wachsendes n (n < 10)**
  - Reduktion der Komplexität der einzelnen Schichten (leichtere Systemevolution)
  - Leistungsverluste, da mehr Schnittstellen zu überqueren sind.[1]
    Bei jedem Übergang:
    - Kapselung/Parameterüberprüfung
    - Datentransport
      - Kopiervorgänge nach oben!
      - Propagieren von Änderungen nach unten!
    - nicht-lokale Fehlerbehandlung ist schwieriger (Verstehen von Fehlermeldungen?)
  - Abnehmende Möglichkeit der Optimierung

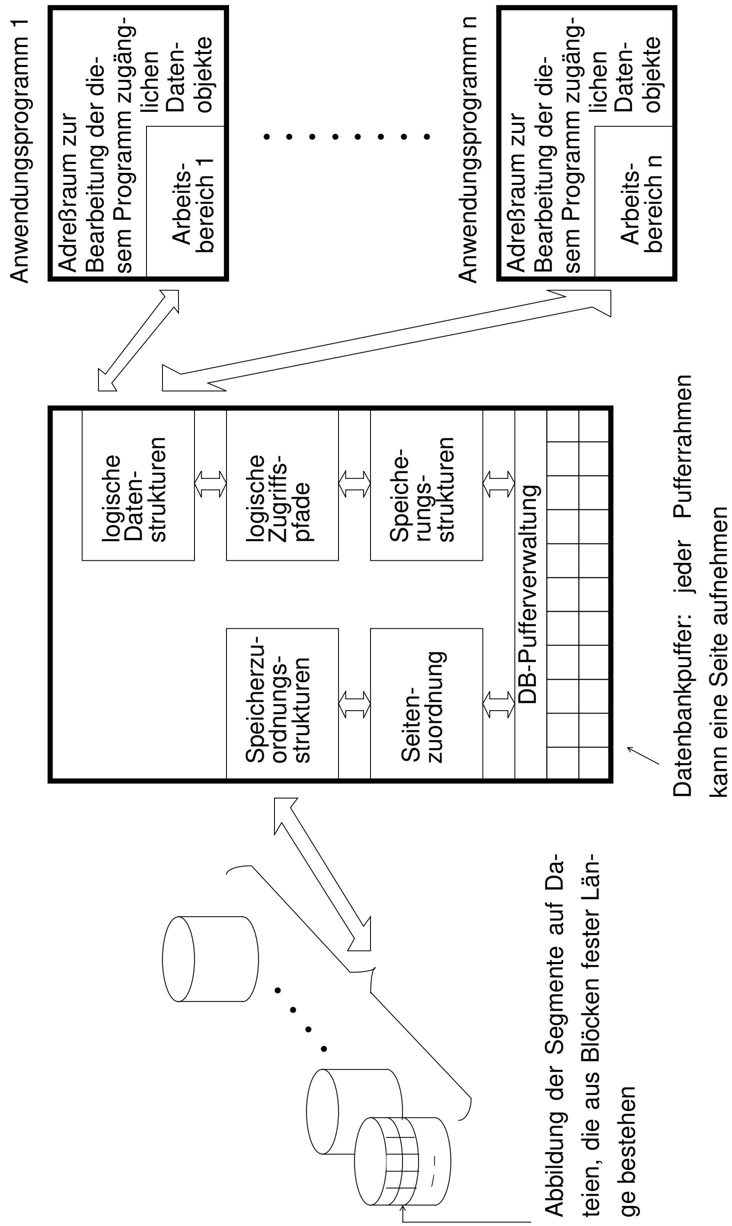
  ➥ Kompromiß bei der Wahl von n!

- **n = 5**



1. Für DBS gilt besonders: „Leistung ist nicht alles, aber ohne Leistung ist alles nichts!“

Schichtenmodell - Laufzeitaspekte



Datenunabhängigkeit im Überblick

Benutzung von:	**Was wird verborgen?**
Mengenorientierte DB-Schnittstelle	Positionsanzeiger und explizite Beziehungskonstrukte im Schema
Satzorientierte DB-Schnittstelle	Zahl und Art der physischen Zugriffspfade; interne Satzdarstellung
Interne Satzschnittstelle	DB-Pufferverwaltung; Recovery-Vorkehrungen
DB-Pufferschnittstelle	Dateiabbildung, Recovery-Unterstützung durch das BS
Dateischnittstelle	Technische Eigenschaften und Betriebsdetails der externen Speichermedien

Architektur eines DBS - weitere Komponenten

- **Entwurfsziel:**
  DBS sollen von ihrem Aufbau und ihrer Einsatzorientierung her in hohem Maße generische Systeme sein. Sie sind so zu entwerfen, daß sie flexibel durch Parameterwahl und ggf. durch Einbindung spezieller Komponenten für eine vorgegebene Anwendungsumgebung zu konfigurieren sind

- **Rolle der Metadaten**
  - Metadaten enthalten Informationen über die zu verwaltenden Daten
  - Sie beschreiben also diese Daten (Benutzerdaten) näher hinsichtlich Inhalt, Bedeutung, Nutzung, Integritätsbedingungen, Zugriffskontrolle usw.
  - Die Metadaten lassen sich unabhängig vom DBVS beschreiben (siehe internes, konzeptionelles und externes Schema)

  ➥ Dadurch erfolgt das „Zuschneidern eines DBS“ auf eine konkrete Einsatzumgebung. Die Spezifikation, Verwaltung und Nutzung von Metadaten bildet die Grundlage dafür, daß DBS hochgradig „generische“ Systeme sind

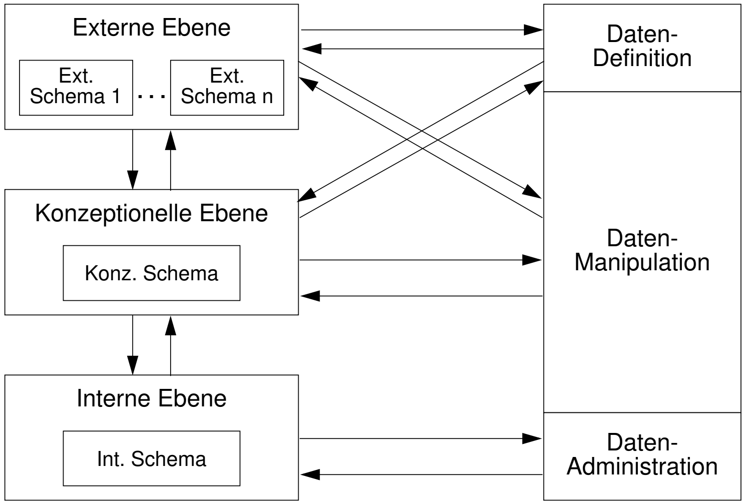
- **Verwaltung der Daten, die Daten beschreiben:**
  - Metadaten fallen in allen DBS-Schichten an
  - Metadatenverwaltung, DB-Katalog, Data-Dictionary-System, DD-System, ...

- **Transaktionsverwaltung**
  - **Realisierung der ACID-Eigenschaften**
    (Synchronisation, Logging/Recovery, Integritätssicherung)

Drei-Schema-Architektur[1]
nach ANSI-SPARC

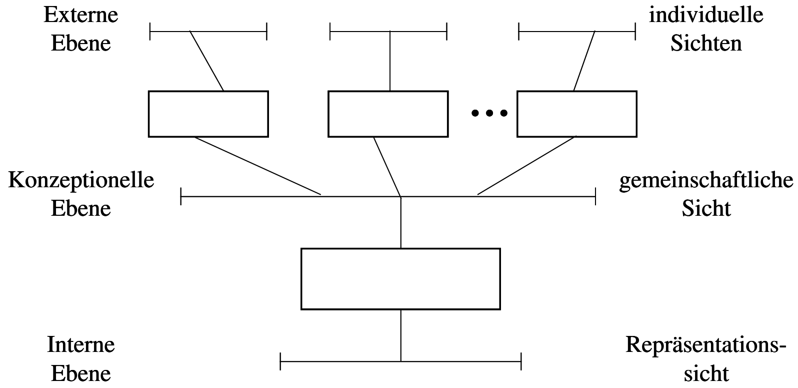
- **Verschiedene Betrachtungsweisen**
  - bisher: Realisierungssicht
    - ➥ Schichtenweiser Aufbau, Datenunabhängigkeit
  - jetzt: Benutzungssicht
    - ➥ Beschreibungsebenen, um aus dem „generischen“ ein „einsatzfähiges“ DBS (DBS-Installation) zu machen

- **3-Ebenen-Architektur nach ANSI/SPARC**



1. Tsichritzis, D. C., Klug, A.: The ANSI/X3/Sparc DBMS Framework Report of the Study Group on Database Management Systems, in: Information Systems 3:3, 1978, 173-191

Drei-Schema-Architektur (2)



Grobarchitektur für Schnittstellen nach ANSI/SPARC

- **Konzeptionelles Schema:**
  - (zeitvariante) globale Struktur; neutrale und redundanzfreie Beschreibung in der Sprache eines spezifischen Datenmodells
  - Beschreibungssprache: DDL (*Data Definition Language*)

- **Externes Schema:**
  - Definition von zugeschnittenen Sichten auf Teile des konzeptionellen Schemas für spezielle Anwendungen (Benutzer)

- **Internes Schema:**
  - legt physische Struktur der DB fest (physische Satzformate, Zugriffspfade etc.)
  - Beschreibungssprache: SSL (*Storage Structure Language*)

- **Gibt es noch weitere DB-Aspekte, die zu beschreiben sind ?**

Drei-Schema-Architektur (3)

- **Stark vereinfachtes Beispiel für die Datenbeschreibung**

Extern (PL/1)	Extern (COBOL)
DCL 1 PERS, 2 PNR CHAR(6), 3 GEH FIXED BIN(31) 4 NAME CHAR(20)	01 PERSC. 02 PERSNR PIC X(6). 02 ABTNR PIC X(4).

Konzeptionelles Schema

ANGESTELLTER

ANG_NUMMER	CHARACTER	(6)
NAME	CHARACTER	(20)
ABT_NUMMER	CHARACTER	(4)
GEHALT	NUMERIC	(5)

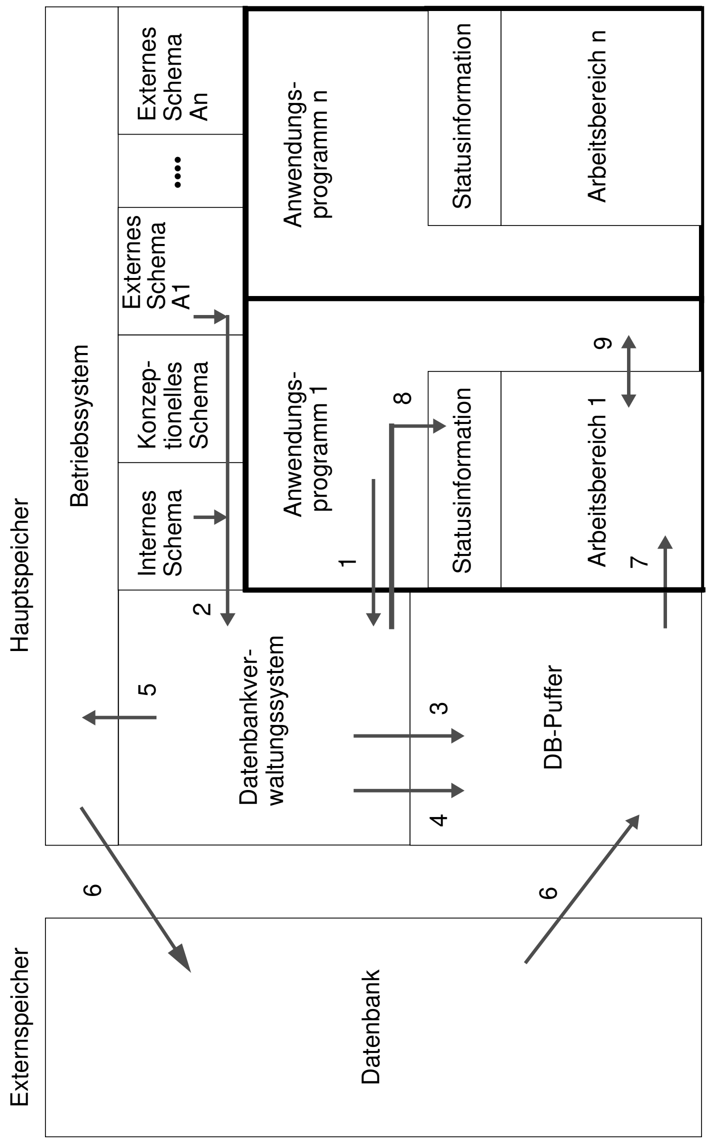
Internes Schema

SPEICHERUNG_PERS LENGTH=40

PREFIX	TYPE=BYTE(6), OFFSET=0
PNR	TYPE=BYTE(6), OFFSET=6, INDEX=PNR
NAME	TYPE=BYTE(20), OFFSET=12
ANR	TYPE=BYTE(4), OFFSET=32
GEHALT	TYPE=FULLWORD, OFFSET=36

- **Sichtenbildung durch das Externe Schema**
  - **Anpassung der Datentypen** an die der Wirtssprache (DBS ist „multi-lingual“)
  - **Zugriffsschutz:** Isolation von Attributen, Relationen, ...
  - **Reduktion der Komplexität:** nur die erforderlichen Daten sind für das Anwendungsprogramm sichtbar

Bearbeitung einer DB-Anweisung - dynamischer Ablauf



Bearbeitung einer DB-Anweisung - dynamischer Ablauf (2)

Beispiel-Schema:

Konzeptionelles Schema: PERS (PNR, NAME, BERUF, ADR, ANR, GEHALT)
I5 ... CHAR(50) ...

Externes Schema: PERS’ (PNR, BERUF, GEHALT, ANR)
PIC 9(5), PIC A(25),...

Interne Bearbeitungsschritte:

1. SELECT * FROM PERS’
   WHERE PNR = ‘12345’
2. Vervollständigen der Verarbeitungsinformation aus Konzeptionellem und Internem Schema; Ermittlung der Seiten# (z. B. durch Hashing): P_{789}
3. Zugriff auf DB-Puffer: erfolgreich (weiter mit 7) oder
4. Zugriff auf die DB über DB-Pufferverwaltung/Betriebssystem
5. Durchführen des E/A-Auftrages
6. Ablegen der Seite im DB-Puffer

P_{789}

12345	Maier	Ing.	KL	K55	50000

• • •

7. Übertragen nach Arbeitsbereich

12345	Ing.	50000	K55

8. Status-Information: Return-Code, Cursor-Info
9. Manipulation mit COBOL-Anweisungen

Zusammenfassung

- **DBS-Charakteristika**
  - Zentralisierte Kontrolle über die operationalen Daten (Rolle des DBA)
  - Adäquate Schnittstellen (Datenmodell und DB-Sprache)
  - Zentrale Kontrolle der Datenintegrität und kontrollierter Mehrbenutzerbetrieb
  - Hoher Grad an Datenunabhängigkeit

- **Drei-Schema-Architektur**
  - Externes Schema zur Benutzerorientierung (Sichtenbildung)
  - Konzeptionelles Schema als logische und neutrale DB-Beschreibung
  - Internes Schema als Beschreibung der physischen DB-Aspekte

- **Beschreibungsmodelle für ein DBS**
  - Schichtenmodell ➡ Erklärungsmodell für die statische Abbildungs-hierarchie
  - Dynamisches Verhalten bei der Bearbeitung einer DB-Anfrage

- **Programmierschnittstellen (DB-Sprachen) (siehe Schichtenmodell)**
  - mengenorientierte DB-Schnittstelle ➥ relationale DBS
  - satzorientierte DB-Schnittstelle ➥ hierarchische und netzwerkartige DBS
  - interne Satzschnittstelle (zugriffsmethodenorientierte Programmierschnittstelle) ➥ „DMS“