

Prof. Dr.-Ing. Dr. h. c. Theo Härder
AG Datenbanken und Informationssysteme
Zi. 36/330, Tel.: 0631-205-4030
E-Mail: haerder@informatik.uni-kl.de
<http://wwwdvs.informatik.uni-kl.de/>

Datenbankanwendung

Wintersemester 2003/2004

TU Kaiserslautern
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern

Vorlesung:

Ort: 48–208

**Zeit: Mo., 10.00–11.30 Uhr
und**

Ort: 48–210

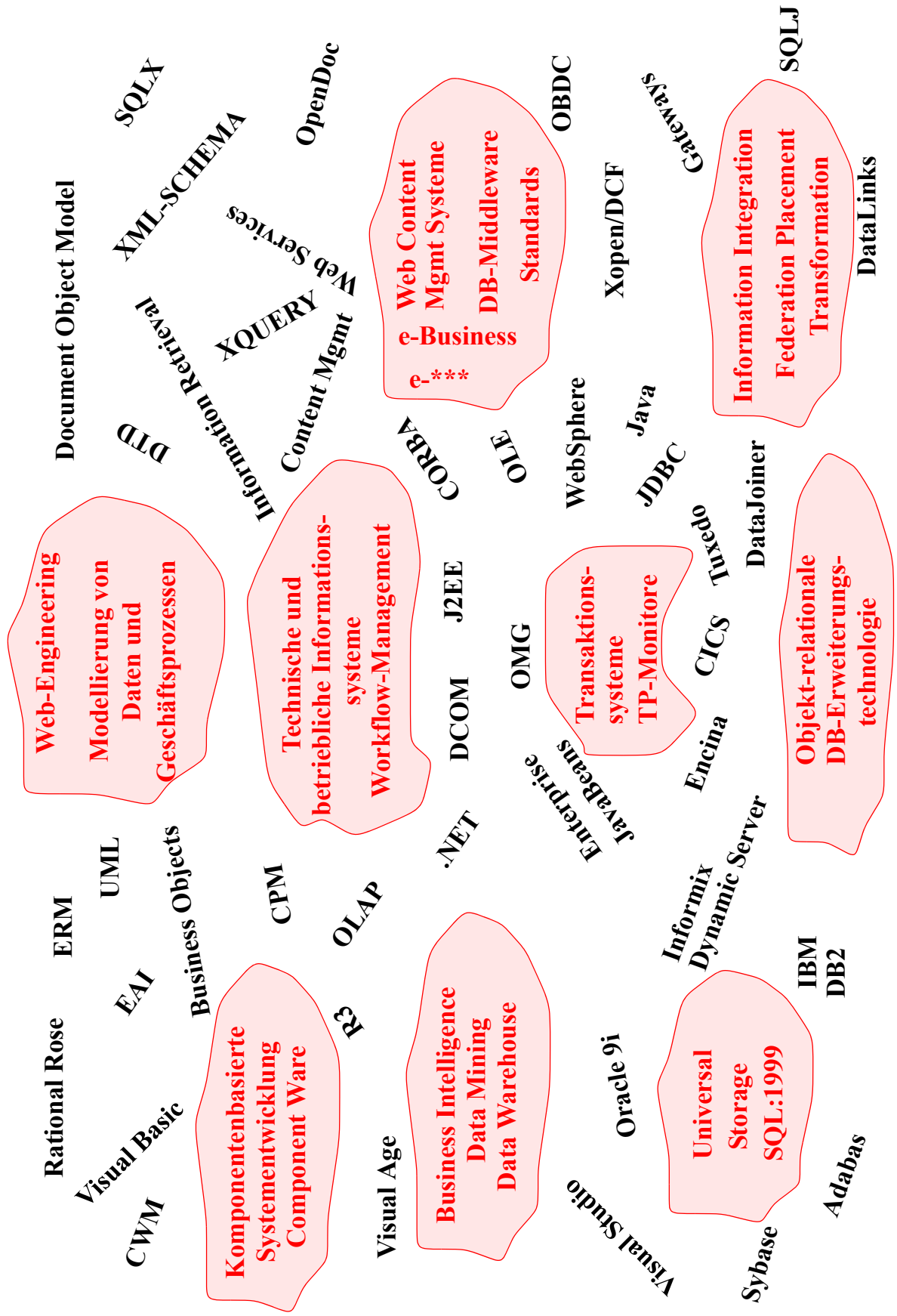
Zeit: Mi., 11.45–13.15 Uhr

Übung:

Ort: 13–222

Zeit: Do., 15.30–17.00 Uhr

Weltbild des Lehrgebietes Datenverwaltungssysteme



Ziele

- **Vermittlung von Grundlagen- und Methodenwissen¹ zur Anwendung von Datenbanksystemen; Erwerb von Fähigkeiten und Fertigkeiten für DB-Administrator und DB-Anwendungsentwickler**
 - Entwurf, Aufbau und Wartung von Datenbanken sowie Programmierung und Übersetzung von DB-Programmen, insbesondere auf der Basis von
 - Relationenmodell und SQL
 - objektorientierten und objekt-relationalen Datenmodellen mit Bezug auf die Standards ODMG und SQL:1999
 - Sicherung der Abläufe von DB-Programmen
 - Transaktionsverwaltung, Synchronisation, Fehlerbehandlung
 - Semantische Integrität, aktive DB-Mechanismen
 - Datenschutz und Zugriffskontrolle
 - Beschreibung und Analyse von Klassen wichtiger DB-Anwendungen wie Transaktionssysteme, Data-Warehouse- und Data-Mining-Systeme
- **Voraussetzungen für Übernahme von Tätigkeiten:**
 - Entwicklung von datenbankgestützten Anwendungen
 - Nutzung von Datenbanken unter Verwendung von (interaktiven) Datenbanksprachen
 - Systemverantwortlicher für Datenbanksysteme, insbesondere Unternehmens-, Datenbank-, Anwendungs- und Datensicherungsadministrator

1. Grundlagenwissen ist hochgradig allgemeingültig und nicht von bestimmten Methoden abhängig. Die Halbwertszeit ist sehr hoch. Methodenwissen muß ständig an die aktuelle Entwicklung angepaßt werden. In der Informatik haben sich die entscheidenden Methoden alle 8-10 Jahre erheblich geändert. Werkzeugwissen ist methodenabhängig. Werkzeuge haben in der Informatik oft nur eine Lebensdauer von 2-3 Jahren.

ÜBERSICHT (vorl.)

0. Übersicht und Motivation

- Datenstrukturen und Datenbanken
- Überblick über die wichtigsten Datenmodelle

1. Anforderungen und Beschreibungsmodelle

- Anforderungen an DBS
- Aufbau von DBS
- Beschreibungsmodelle
(Fünf-Schichten-Modell, Drei-Ebenen-Beschreibungsarchitektur)

2. Logischer DB-Entwurf

- Konzeptioneller DB-Entwurf
- Normalformenlehre (1NF, 2NF, 3NF, 4NF)
- Synthese von Relationen

3. Tabellen und Sichten

- Datendefinition von SQL-Objekten
- Schemaevolution
- Indexstrukturen
- Sichtenkonzept

4. Anwendungsprogrammier-Schnittstelle

- Kopplung mit einer Wirtssprache
- Übersetzung und Optimierung von DB-Anweisungen
- Eingebettetes / Dynamisches SQL, PSM
- CLI, JDBC und SQLJ

ÜBERSICHT (2)

5. Transaktionsverwaltung

- Transaktionskonzept
- Ablauf von Transaktionen
- Commit-Protokolle

6. Serialisierbarkeit

- Anomalien beim Mehrbenutzerbetrieb
- Theorie der Serialisierbarkeit
 - Final-State-Serialisierbarkeit, Sichtenserialisierbarkeit
 - Konfliktserialisierbarkeit
- Klassen von Historien

7. Synchronisation

- Sperrverfahren
(hierarchische Verfahren, Deadlocks)
- Konsistenzebenen
- Optimierungen
(Optimistische Verfahren, Prädikatssperren, Mehrversions- und Zeitstempelverfahren, Objektsperren, spezielle Protokolle)
- Leistungsbewertung und Lastkontrolle

8. Logging und Recovery

- Fehlermodell und Recovery-Arten
- Logging-Strategien
- Recovery-Konzepte – Abhängigkeiten
- Sicherungspunkte
- Transaktions-, Crash- und Medien-Recovery

ÜBERSICHT (3)

9. Integritätskontrolle und aktives Verhalten

- Semantische Integritätskontrolle
- Regelverarbeitung in DBS, Trigger-Konzept von SQL
- Definition und Ausführung von ECA-Regeln

10. Datenschutz und Zugriffskontrolle

- Technische Probleme des Datenschutzes
- Konzepte der Zugriffskontrolle, Zugriffskontrolle in SQL
- Sicherheitsprobleme in statistischen Datenbanken

11. Objektorientierung und Datenbanken

- Beschränkungen klassischer Datenmodelle
- Grundkonzepte der Objektorientierung
- SQL:1999 – Neue Funktionalität
 - ORDBS: Anforderungen, Architekturvorschläge
 - Erhöhung der Anfragemächtigkeit, Rekursion

12. Große Objekte

- Anforderungen und Verarbeitung mit SQL
- Lokator-Konzept
- Speicherungsstrukturen, . . .

13. Anwendungsklassen

- Transaktionssysteme
- Data Warehouse
- Data Mining
- . . .

LITERATURLISTE

- Elmasri, R., Navathe, S. B.:* Grundlagen von Datenbanksystemen, 3. Auflage, Pearson Studium, 2002
- Garcia-Molina, H., Ullman, J.D., Widom, J.:* Database Systems – The Complete Book, Prentice Hall, Upper Saddle River, NJ, 2002
- Härder, T., Rahm, E.:* Datenbanksysteme – Konzepte und Techniken der Implementierung, Springer-Verlag, Berlin, 2001
- Heuer, A., Saake, G.:* Datenbanken – Konzepte und Sprachen, 2. Auflage, Int. Thompson Publ. Comp., 2000
- Kemper, A., Eickler, A.:* Datenbanksysteme – Eine Einführung, 4. Auflage, Oldenbourg-Verlag, 2001
- Lewis, P. M., Bernstein, A., Kifer, M.:* Databases and Transaction Processing – An Application-Oriented Approach, Addison-Wesley, 2002
- Ramakrishnan, R.:* Database Management Systems, McGraw-Hill, Boston, 1998
- Weikum, G., Vossen, G.:* Transactional Information Systems, Morgan Kaufmann Publishers, San Francisco, CA, 2002

ZEITSCHRIFTEN:

- ACM TODS* Transactions on Database Systems, ACM Publikation (vierteljährlich)
- Information Systems* Pergamon Press (6-mal jährlich)
- The VLDB Journal* VLDB Foundation (vierteljährlich)
- Informatik – Forschung und Entwicklung* Springer Verlag (vierteljährlich)
- ACM Computing Surveys* ACM-Publikation (vierteljährlich)

TAGUNGSBÄNDE:

- SIGMOD* Tagungsbände, jährliche Konferenz der ACM Special Interest Group on Management of Data
- VLDB* Tagungsbände, jährliche Konferenz „Very Large Data Bases“
- ICDE* Tagungsbände, jährliche Konferenz „Int. Conf. on Data Engineering“
- BTW* Tagungsbände der alle 2 Jahre stattfindenden Tagungen „Datenbanksysteme für Business, Technologie und Web“ der GI, und weitere Tagungen innerhalb des GI-FB „DBIS“ und viele weitere Konferenzreihen

Datenstrukturen?

- **Bisher bekannte Datenstrukturen**

- Felder (Reihungen, Arrays,)
- Verbunde (Tupel, Sätze, Records,)
- Listen, Graphen, Bäume

➔ bisher im **Hauptspeicher**, d. h. Bestand nur für die Dauer einer Programmausführung („transiente“ Daten)

- **Hier nun neue Aspekte:**

Nutzung von Hintergrundspeicher und neuen Operationen

- **Persistenz:**

Werte bleiben über Programmende, Sitzungsende, Rechnereinschaltung, hinaus erhalten

- andere Arten des Zugriffs: Lese- und Schreiboperationen, in vorgegebenen Einheiten von Blöcken und Sätzen

➔ **Strukturen und zugehörige Algorithmen (Suchen, Sortieren), die im Hauptspeicher optimal sind, sind es auf Sekundärspeicher nicht unbedingt!**

- gezielter, wertabhängiger Zugriff auch auf sehr große Datenmengen²
- umfangreiche Attributwerte (z. B. Bilder, Texte)
- Homogenisierung und Verarbeitung der im Web verfügbaren Daten³

-
2. 1 Gigabyte (GByte) = 1,000 Megabytes = 10^9 Bytes
1 Terabyte (TByte) = 1,000 Gigabytes
1 Petabyte (PByte) = 1,000 Terabytes
1 Exabyte (EByte) = 1,000 Petabytes
1 Zettabyte (ZByte) = 1,000 Exabytes
1 Yottabyte (YByte) = 1,000 Zettabytes

3. Schätzungen:
How much data is out there?
300 PByte online, 8 EByte offline, 200 EByte analog data (audio, video, etc.)

Datenmodelle?

- **Mengen von Konstruktoren**

zur (abstrakten) Erzeugung von Datenstrukturen
mit darauf definierten Operatoren

z. B. Tabellen (Relationen): Mengen von gleichartig strukturierten Tupeln

```
CREATE TABLE STUDENT
    (MATRIKELNUMMER      INTEGER,
     NACHNAME            VARCHAR (40) ,
     FBNUMMER            INTEGER,
     GEBURTSDATUM       CHAR (8) ,
     . . . . . )

CREATE TABLE FACHBEREICH
    (FBNUMMER            INTEGER,
     FBNAME              VARCHAR (20) ,
     DEKAN               PROF ,
     . . . . . )
```

nicht nur ein einzelner Satz, sondern Menge

- **Wichtige Rolle von Beziehungen**

in einigen Datenmodellen auch sehr spezielle Arten von **Beziehungen**
zwischen Sätzen und/oder Tupeln: Hierarchien, Zusammensetzungen,
funktionale Zuordnungen u.v.a.m.

- **Modellbegriff**

vorgegebene Menge von (sprachlichen) Ausdrucksmitteln,
mit denen Diskursbereich beschrieben (erfaßt) werden muß

➔ auch Programmiersprachen und Betriebssysteme haben ein
„Datenmodell“

Datenbanken?

- **Große Datenmengen**
 - auch, aber **nicht immer** entscheidend
- **Übereinstimmung von Modell und Miniwelt⁴**
 - Genauigkeit der Abbildung und Erhaltung ihrer Integrität (Bedeutungstreue)
 - zeitgerechter und durch Integritätsbedingungen abgesicherter Änderungsdienst
- **Datenunabhängigkeit (der Anwendungen)**
 - Benutzung der Daten, ohne Details der systemtechnischen Realisierung zu kennen (abstraktes „Datenmodell“, z. B. Tabellen)
 - einfache Handhabung der Daten, mächtige Auswertungsoperationen
- **Offenheit der Daten für neue Anwendungen (Anwendungsneutralität der Speicherung)**
 - symmetrische Organisationsformen (keine Bevorzugung einer Verarbeitungs- und Auswertungsrichtung)
 - explizite Darstellung der Annahmen/Zusicherungen (nicht in den Anwendungsprogrammen verstecken)
 - Redundanzfreiheit aus der Sicht der Anwendung: keine wiederholte Speicherung in unterschiedlicher Form für verschiedene Anwendungen
 - Konsistenzüberwachung durch das Datenbanksystem

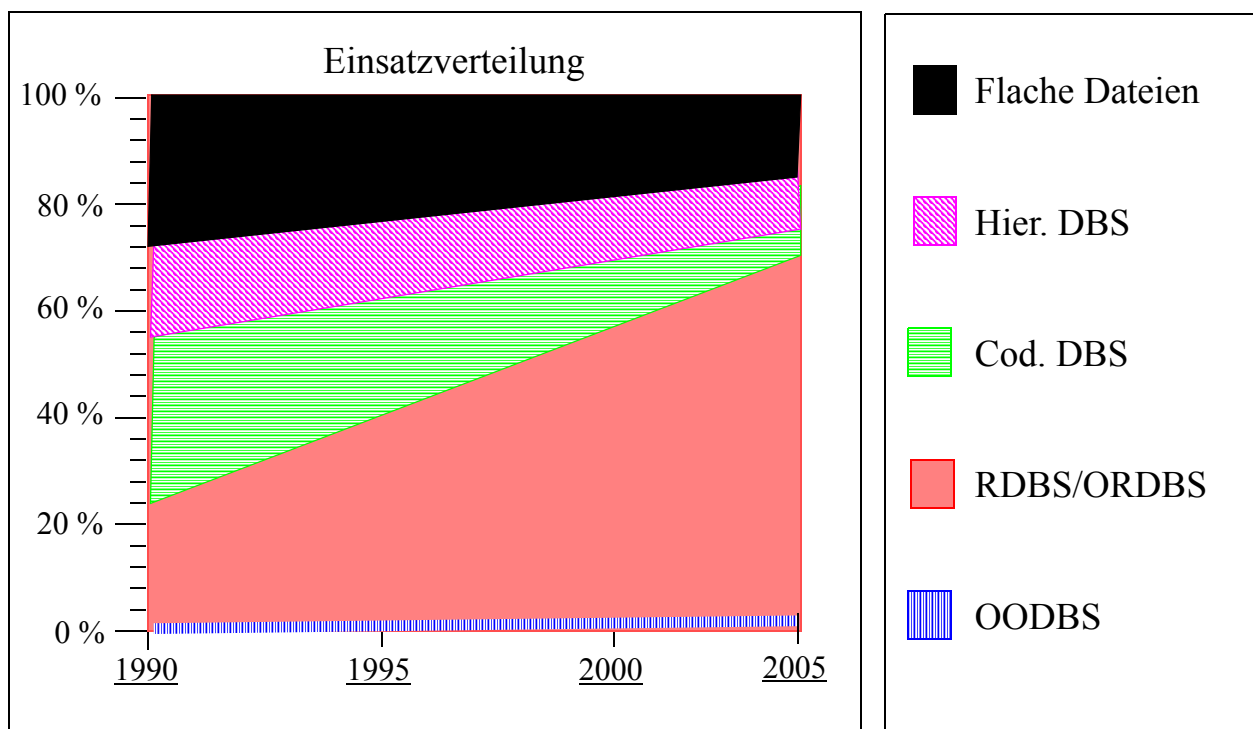
4. Ein Datenbanksystem verwaltet Daten einer realen oder gedanklichen Anwendungswelt. Diese Daten gehen aus Informationen hervor, die stets aus den Sachverhalten und Vorgängen dieser Anwendungswelt durch gedankliche Abstraktionen (Abbilder, Modelle) gewonnen werden. Sie beziehen sich nur auf solche Aspekte des betrachteten Weltausschnitts, die für den Zweck der Anwendung relevant sind. Ein solcher Weltausschnitt wird auch als *Miniwelt* (Diskurswelt) bezeichnet.

Datenbanken? (2)

- **Transaktionskonzept mit Garantie von ACID-Eigenschaften**
 - Atomarität (*atomicity*)
 - Konsistenz (*consistency*)
 - Isolation (*isolated execution*)
 - Dauerhaftigkeit (*durability*)
- **Ausfallsicherheit**
 - Aufzeichnung redundanter Daten im Normalbetrieb
 - Replikation von Datenstrukturen
 - Vorkehrungen für den Katastrophenfall
 - automatische Reparatur der Datenbestände nach Programm-, System- und Gerätefehlern
 - Rückgängigmachen unvollständiger Transaktionen, so daß sie wiederholt werden können
 - Wiederherstellen der Ergebnisse vollständiger Transaktionen, so daß sie nicht wiederholt werden müssen
- **Mehrbenutzerbetrieb**
 - gleichzeitiger (zeitlich eng verzahnter) Zugriff verschiedener Anwendungen und Benutzer auf gemeinsame Daten
 - Synchronisation, d. h. Vermeidung von Fehlern in der wechselseitigen Beeinflussung
 - Kooperation über gemeinsame Daten mit hohem Aktualitätsgrad

Verteilung von DBS und Dateien

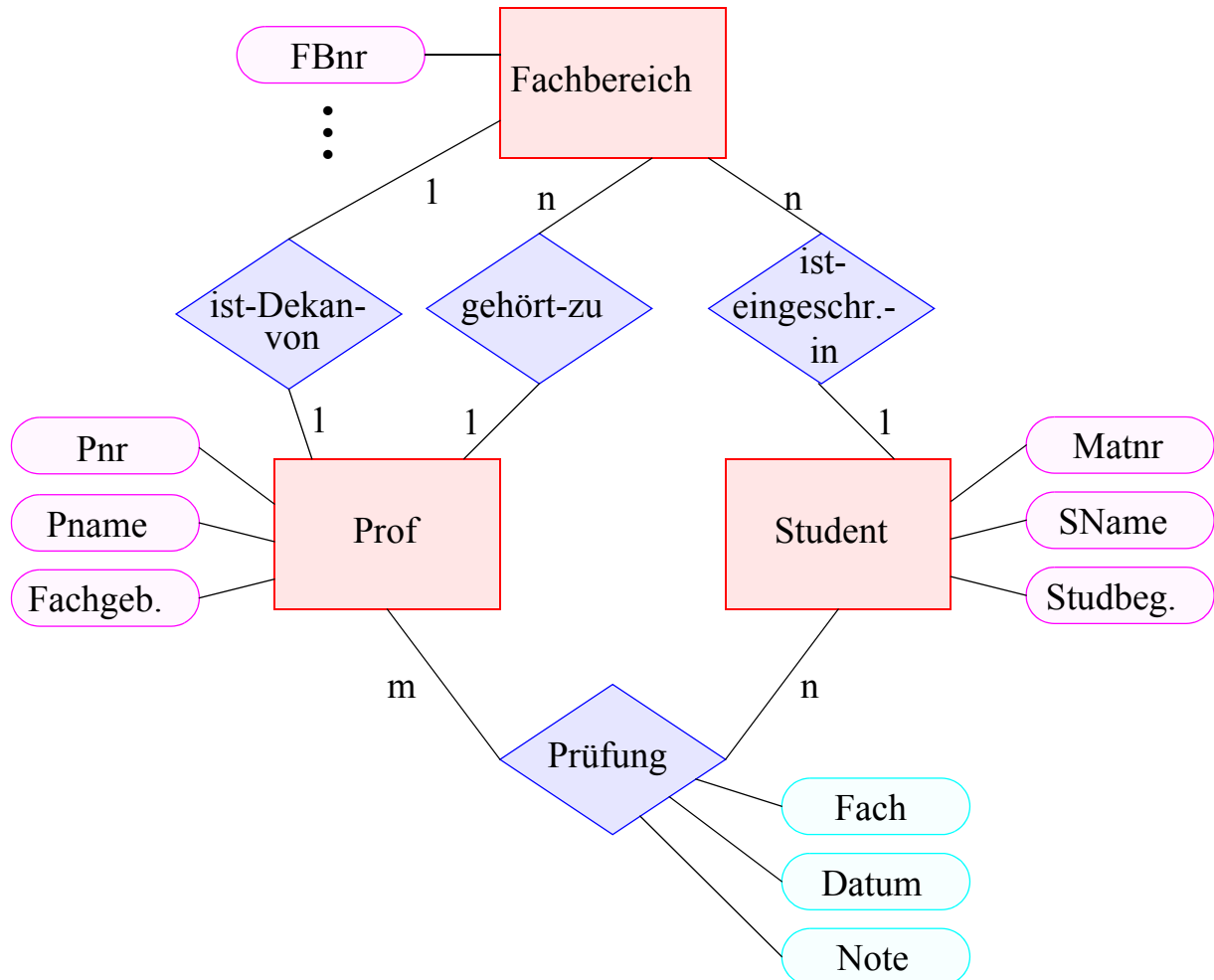
- **Es gibt verschiedenartige Datenmodelle und die sie realisierenden DBS**
 - relational und objekt-relational (RDBS/ORDBS)
 - hierarchisch (DBS nach dem Hierarchiemodell)
 - netzwerkartig (DBS nach dem Codasyl-Standard)
 - objektorientiert (OODBS)



- **Künftige DBS**
 - Aufstellung berücksichtigt nur strukturierte Daten. 85% der weltweit verfügbaren Daten aber sind semi- oder unstrukturiert (Internet, wiss. Aufzeichnungen und Experimente usw.)
 - SQL-XML-DBS, XML-SQL-DBS native XML-DBS
- **Voraussetzung für die Vorlesung: Beherrschung von**
 - Informationsmodellen (erweitertes ER-Modell)
 - Relationenmodell und Relationenalgebra
 - SQL-92 als Standardsprache

Überblick über die wichtigsten Datenmodelle

- Entity/Relationship-Diagramm der Beispiel-Miniwelt



- Spezifikation benutzerdefinierter Beziehungen (rein struktureller Natur)
- Klassifikation der Beziehungstypen (1:1, 1:n, n:m)

➔ **relativ semantikarme Darstellung von Weltausschnitten**

- deshalb Verfeinerungen/Erweiterungen durch
 - **Kardinalitätsrestriktionen** ([1,1]:[1,10], [0:1]:[0:*])
 - **Abstraktionskonzepte**
(Klassifikation/Instantiierung, Generalisierung/Spezialisierung, Element-/Mengen-Assoziation, Element-/Komponenten-Aggregation)

Relationenmodell – Beispiel

DB-Schema

FB		
<u>FBNR</u>	FBNAME	DEKAN

PROF			
<u>PNR</u>	PNAME	FBNR	FACHGEB

STUDENT			
<u>MATNR</u>	SNAME	FBNR	STUDBEG

PRÜFUNG				
<u>PNR</u>	<u>MATNR</u>	FACH	DATUM	NOTE

Ausprägungen

FB	<u>FBNR</u>	FBNAME	DEKAN
	FB 9	WIRTSCHAFTSWISS	4711
	FB 5	INFORMATIK	2223

PROF	<u>PNR</u>	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	<u>MATNR</u>	SNAME	FBNR	STUDBEG
	123 766	COY	FB 9	1.10.00
	225 332	MÜLLER	FB 5	15.04.97
	654 711	ABEL	FB 5	15.10.99
	226 302	SCHULZE	FB 9	1.10.00
	196 481	MAIER	FB 5	23.10.00
	130 680	SCHMID	FB 9	1.04.02

PRÜFUNG	<u>PNR</u>	<u>MATNR</u>	FACH	PDATUM	NOTE
	5678	123 766	BWL	22.10.03	4
	4711	123 766	OR	16.01.02	3
	1234	654 711	DV	17.04.03	2
	1234	123 766	DV	17.04.03	4
	6780	654 711	SP	19.09.03	2
	1234	196 481	DV	15.10.03	1
	6780	196 481	BS	23.10.03	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 2000 begonnen haben.

```
SELECT *
FROM STUDENT
WHERE FBNR = 'FB5' AND STUDBEG < '1.1.00'
```

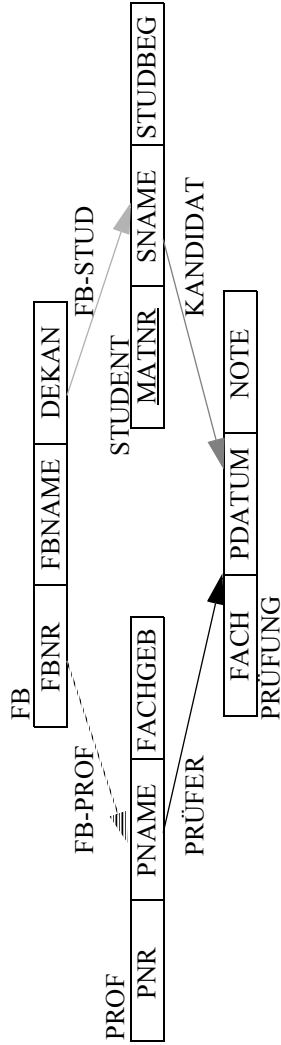
Q2: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
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.

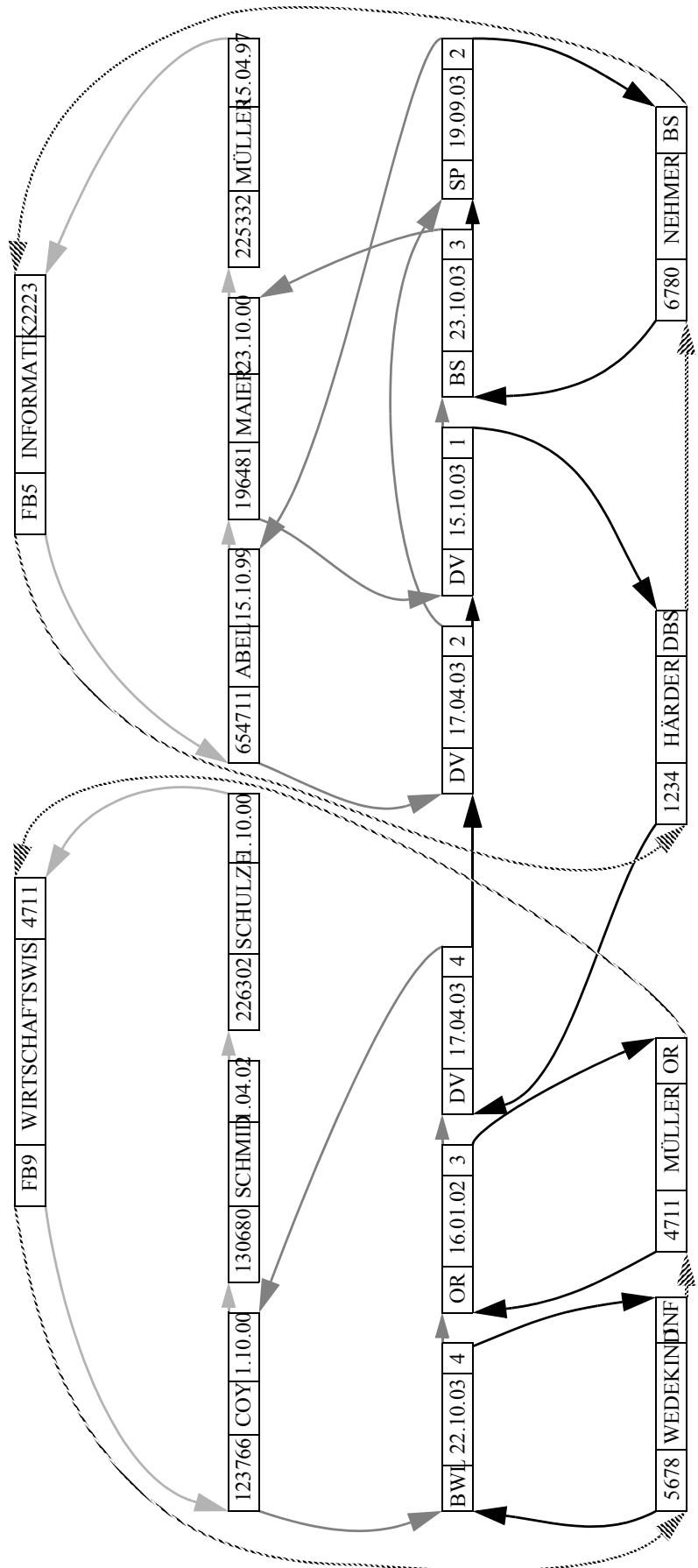
```
SELECT S.FBNR, AVG (P.NOTE)
FROM PRÜFUNG P, STUDENT S
WHERE P.FACH = 'DV' AND P.MATNR = S.MATNR
GROUP BY S.FBNR
HAVING (SELECT COUNT(*)
        FROM STUDENT T
        WHERE T.FBNR = S.FBNR) > 1000
```

Netzwerkmodell – Beispiel



DB-Schema

Ausprägungen



Netzwerkmodell – Beispiel (2)

- **Prozedurale DB-Sprachen**

- Programmierer als Navigator
- satzweiser Zugriff über vorhandene Zugriffspfade

- **Anfragebeispiele**

Q4: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 2000 begonnen haben.

```
MOVE 'FB5' TO FBNR OF FB;  
FIND FB USING FBNR;
```

```
NEXT_STUDENT:  FETCH NEXT STUDENT WITHIN FB-STUD SET;  
                IF  END_OF_SET THEN EXIT;  
                IF  STUDBEG < '1.1.00' THEN  
                    PRINT STUDENT RECORD;  
                GOTO NEXT_STUDENT;
```

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

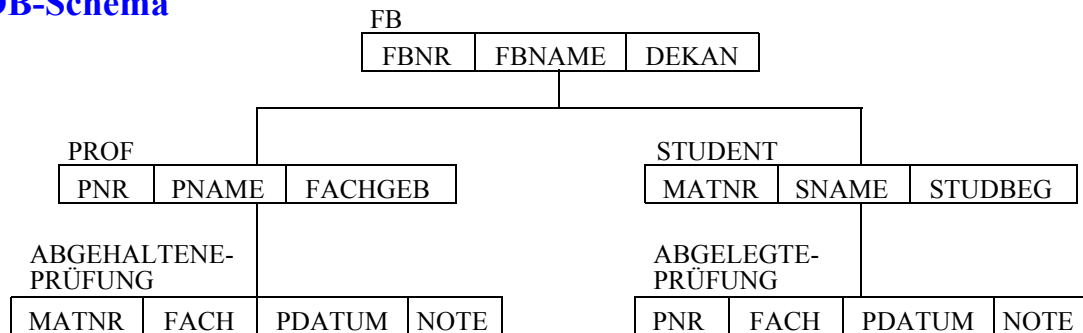
```
MOVE 'FB5' TO FBNR OF FB;  
FIND FB USING FBNR;
```

```
NEXT_STUDENT:  FETCH NEXT STUDENT WITHIN FB-STUD SET;  
                IF  END_OF_SET THEN EXIT;
```

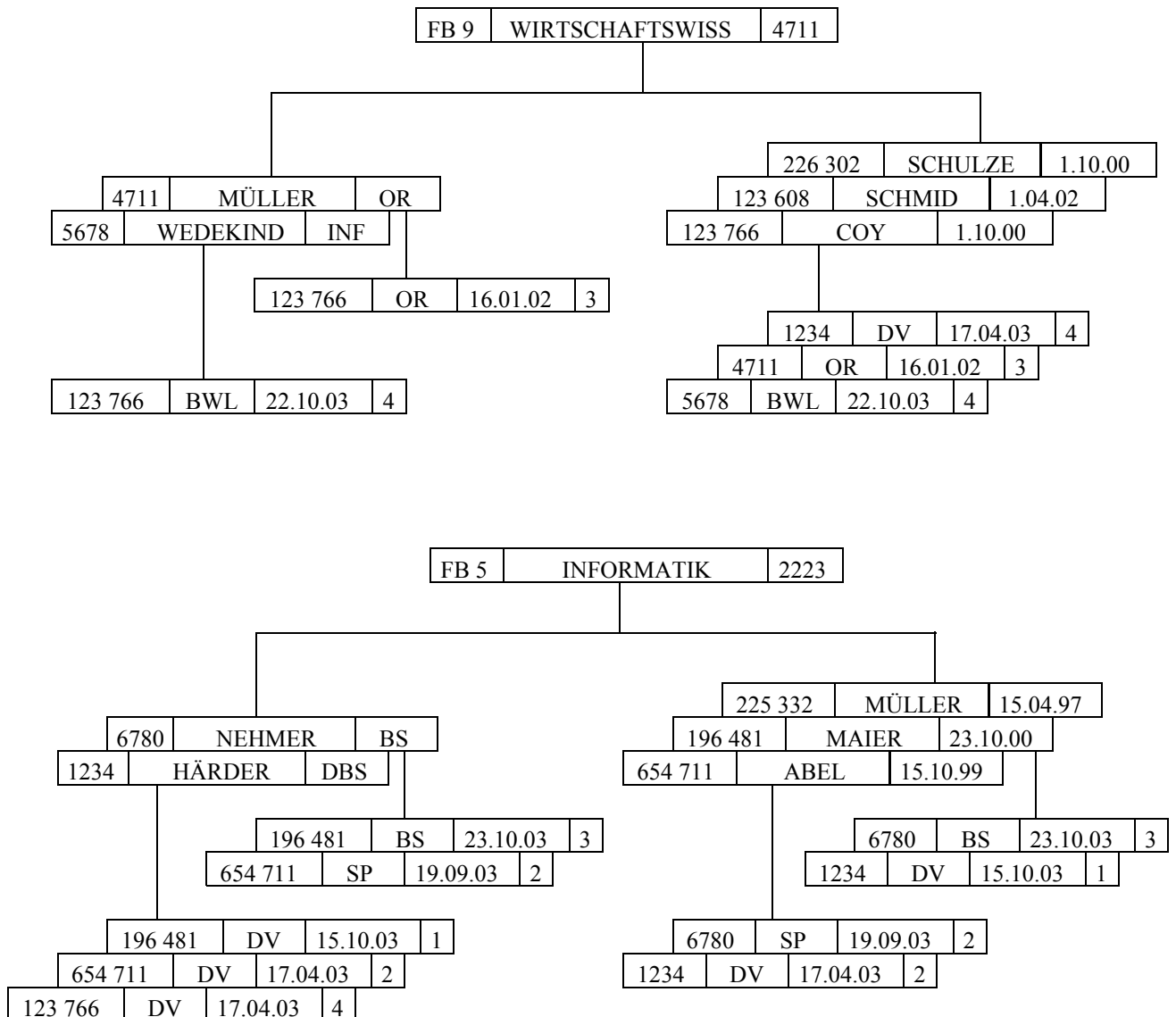
```
NEXT_PRÜFUNG:  FETCH NEXT PRÜFUNG WITHIN KANDIDAT SET;  
                IF  END_OF_SET THEN GOTO NEXT_STUDENT;  
                IF  FACH = 'DV' AND NOTE ≤ '2'  
                DO;  
                    PRINT STUDENT RECORD;  
                GOTO NEXT_STUDENT;  
                END;  
                GOTO NEXT_PRÜFUNG;
```

Hierarchisches Datenmodell – Beispiel

DB-Schema



Ausprägungen



Hierarchisches Datenmodell – Beispiel (2)

- **Prozedurale DB-Sprachen**

- navigierende Verarbeitung
- satzweiser Zugriff in Baumstrukturen
- durch den Zwang zu Hierarchien
 - unnatürliche Organisation bei komplexen Beziehungen (n:m)
 - Auswertungsrichtung ist vorgegeben

- **Anfragebeispiele**

Q6: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 2000 begonnen haben.

```
GU FB (FBNR = 'FB5');
```

```
NEXT_STUDENT: GNP STUDENT (STUDBEG < '1.1.00');  
IF END_OF_PARENT THEN EXIT;  
PRINT STUDENT RECORD;  
GOTO NEXT_STUDENT;
```

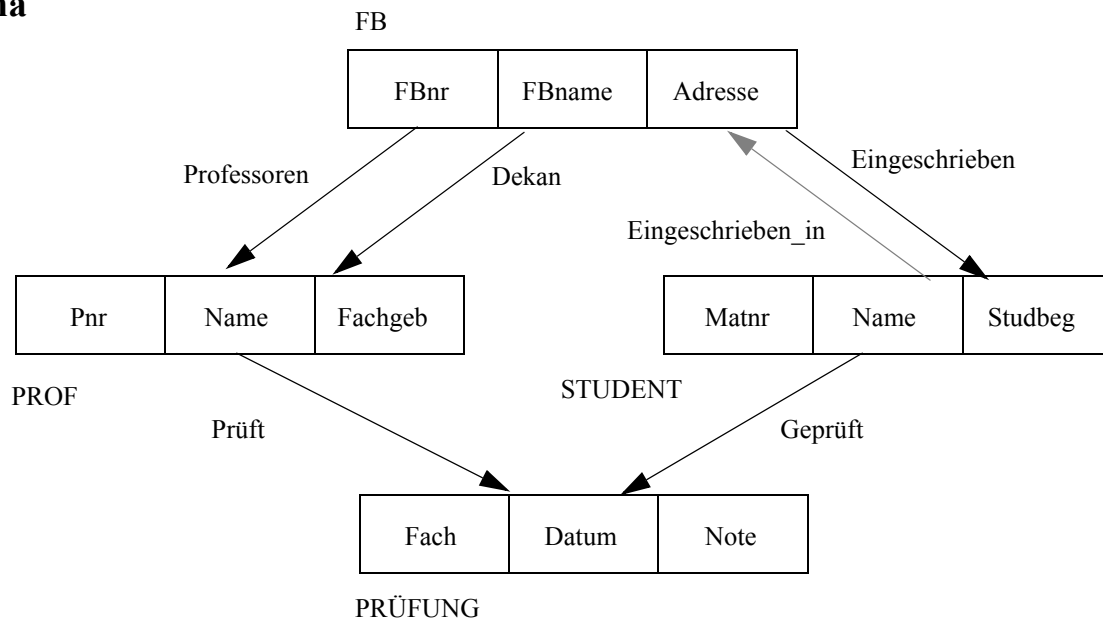
Q7: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
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;
```

Objektorientierte Datenmodelle

Schema



- **Typdefinitionen (fiktives Modell)**

```
CREATE TYPE ADRESSEN_T
```

```
(Strasse char(30),  
Hausnummer integer,  
PLZ integer,  
Ort char(30));
```

```
CREATE TYPE DATE_T
```

```
(Jahr integer,  
Monat integer,  
Tag integer);
```

```
CREATE TYPE PERSON_T
```

```
(Name char(30)  
Adresse ADRESSEN_T  
... (* weitere allg. Personendaten *)  
... );
```

```
CREATE FUNCTION ... ; (* Definition von zugehörigen Methoden *)
```

Objektorientierte Datenmodelle (2)

- Typdefinitionen

CREATE TYPE PRÜFUNG_T

(Fach char(30),
Pdatum DATE_T,
Note integer)

CREATE FUNCTION Prüfungsbericht (Prüfung REF(PRÜFUNG_T))
RETURNS integer ... ;

CREATE TYPE FB_T

(FBnr char(30),
FBname char(30),
Adresse ADRESSEN_T,
Dekan REF(PROF_T),
Eingeschrieben SET(REF(STUDENT_T))
Professoren SET(REF(PROF_T)));

CREATE FUNCTION Durchschnittsnote.pro.Student (FB_T) ...;

CREATE FUNCTION Durchschnittsnote.pro.Fach (FB_T) ...;

CREATE TYPE PROF_T UNDER PERSON_T

(Pnr integer,
Fachgeb char(30),
Prüft SET(REF(PRÜFUNG_T)));

CREATE FUNCTION

Prüfungsergebnis (PROF_T, STUDENT_T, PRÜFUNG_T), ... ;

CREATE TYPE STUDENT_T UNDER PERSON_T

(Matnr integer,
Studbeg DATE_T,
Eingeschrieben_in REF(FB_T)
Geprüft SET(REF(PRÜFUNG_T)));

CREATE FUNCTION

Prüfungsanmeldung (PROF_T, STUDENT_T, PRÜFUNG_T) ...;

Prüfungsverlegung (PROF_T, STUDENT_T, PRÜFUNG_T) ...;

Objektorientierte Datenmodelle (3)

- **Tabellendefinitionen**

```
CREATE TABLE FB OF FB_T  
(PRIMARY KEY FBnr  
Dekan WITH OPTIONS SCOPE Prof,  
Professoren WITH OPTIONS SCOPE Prof,  
Eingeschrieben WITH OPTIONS SCOPE Student);
```

```
CREATE TABLE Prof OF PROF_T  
(PRIMARY KEY Pnr  
Prüft WITH OPTIONS SCOPE Prüfung);
```

```
CREATE TABLE Student OF STUDENT_T  
(PRIMARY KEY Matrnr  
Eingeschrieben_in WITH OPTIONS SCOPE FB,  
Geprüft WITH OPTIONS SCOPE Prüfung);
```

```
CREATE TABLE Prüfung OF PRÜFUNG_T  
(PRIMARY KEY Fach, Pdatum, Note);
```

Objektorientierte Datenmodelle – Ausprägungen

FB	FBnr	FBname	Dekan	Adresse	Professoren	Eingeschrieben
@1	FB 9	WIRTS.-WISS	@5	@201	@4, @5, ...	@7, @10, ...
@2	FB 5	INFORMATIK	@111	@202	@3, @6, ...	@8, @9, ...

PROF	Pnr	Name	Fachgeb	Prüft
@3	1234	HÄRDER	DATENBANKSYSTEME	@15, @16, ...
@4	5678	WEDEKIND	INFORMATIONSSYSTEME	@13, ...
@5	4711	MÜLLER	OPERATIONS RESEARCH	@14
@6	6780	NEHMER	BETRIEBSSYSTEME	@17, @19

STUDENT	Matnr	Sname	Studbeg	Eingeschr._in	Geprüft
@7	123 766	COY	1.10.00	@1	@13, @16, ...
@8	225 332	MÜLLER	15.4.97	@2	-
@9	654 711	ABEL	15.10.99	@2	@15, @17
@10	226 302	SCHULZE	1.10.00	@1	-
@11	194 481	MAIER	23.10.00	@2	@18, @19
@12	130 680	SCHMID	1.4.02	@1	-

PRUEFUNG	Fach	Pdatum	Note
@13	BWL	22.10.03	4
@14	OR	16.1.02	3
@15	DV	17.4.03	2
@16	DV	17.4.03	4
@17	SP	19.9.03	2
@18	DV	15.10.03	1
@19	BS	23.10.03	3

Objektorientierte Datenmodelle – Anfragebeispiele

- **Prozedurale und deskriptive DB-Sprachen**

- Zugriff über OIDs und Referenzen
- Einsatz von Zugriffsroutinen (Methoden, Funktionen, Prozeduren)
- Hinzufügen von deklarativen und mengenorientierten Zugriffsmöglichkeiten

- **Anfragebeispiele**

Q8: Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 2000 begonnen haben.

```
SELECT *
FROM STUDENT S
WHERE S.Eingeschrieben_in->FBnr = 'FB5'
      AND S.Studbeg < '1.1.00';
```

Q9: Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
SELECT *
FROM STUDENT S
WHERE S.Eingeschrieben_in->FBnr = 'FB5' AND EXISTS
      (SELECT *
       FROM TABLE (S.Geprüft) P
       WHERE P.Fach = 'DV' AND P.Note <= 2);
```

Q12: Finde die Durchschnittsnoten der DV-Prüfungen für alle Fachbereiche mit mehr als 1000 Studenten.

```
SELECT F.FBnr, Durchschnittsnote.pro.Fach (F, 'DV')
FROM FB F,
WHERE 1000 < (SELECT COUNT(*)
              FROM TABLE (F.Eingeschrieben));
```


Bewertung – Relationenmodell

- **Informationen des Benutzers**

- ausschließlich durch den Inhalt der Daten
- keine physischen Verbindungen
- keine bedeutungsvolle Ordnung

- **Deskriptive Sprachen**

- hohes Auswahlvermögen und Mengenorientierung
- leichte Erlernbarkeit auch für den DV-Laien

- **Vorteile**

- strenge theoretische Grundlage
- einfache Informationsdarstellung durch Tabellen
- keine Bindung an Zugriffspfade oder Speichertechnologie
- keine Aussage über die Realisierung
- hoher Grad an Datenunabhängigkeit
- symmetrisches Datenmodell; d.h. es gibt keine bevorzugte Zugriffs- oder Auswertungsrichtung
- Parallelisierung der Auswertung möglich
- Verteilung der Daten über Prädikate

- **Nachteile**

- zu starke Beschränkung der Modellierungsmöglichkeiten
- Schwerfällige und unnatürliche Modellierung bei komplexeren Objekten
- Einsatz von nicht-prozeduralen Sprachen „soll“ Ineffizienz implizieren!?
- aber: Optimierung der Anforderungen liegt in der Systemverantwortung

Bewertung – Hierarchische Datenmodelle und Netzwerkmodelle

- **Informationen des Benutzers**

- Darstellung durch „Verbindungen“ von Datensätzen
- Berücksichtigung ihrer Anordnung in Speicherungsstrukturen

- **Prozedurale Sprachen**

- Programmierer als Navigator
- satzweiser Zugriff über vorhandene Zugriffspfade

- **Vorteile**

- Entwurf effizienter Programme durch den Anwendungsprogrammierer
- höheres Leistungsvermögen bei zugeschnittenen Zugriffspfadstrukturen

- **Nachteile**

- starke Bindung an Zugriffspfade
- geringerer Grad an Datenunabhängigkeit
- beschränkt auf die Benutzerklassen „Anwendungsprogrammierer und “Systempersonal“
- bei hierarchischen Datenmodellen
 - unnatürliche Organisation bei komplexen Beziehungen (n:m)
 - Auswertungsrichtung ist vorgegeben

Bewertung – Objektorientierte Modelle

- **Informationen des Benutzers**

- Darstellung durch „Verbindungen“ von Datensätzen
- sowie durch den Inhalt der Daten

- **Prozedurale und deskriptive Sprachen**

- Programmierer als Navigator
- hohes Auswahlvermögen
- satzweiser Zugriff (über vorhandene Zugriffspfade)

- **Vorteile (auch bei objekt-relationalen Datenmodellen)**

- reichhaltiger Vorrat an Modellierungskonzepten
- Generalisierung/Spezialisierung: Typ- und Tabellenhierarchien – Vererbung
- Einsatz benutzerdefinierter Datentypen und Funktionen
- Aktive Mechanismen: Trigger, ECA-Regeln
- Entwurf effizienter Programme durch den Anwendungsprogrammierer
- Leistungsvermögen aufgrund der Nutzung zugeschnittener Zugriffspfadstrukturen

- **Nachteile**

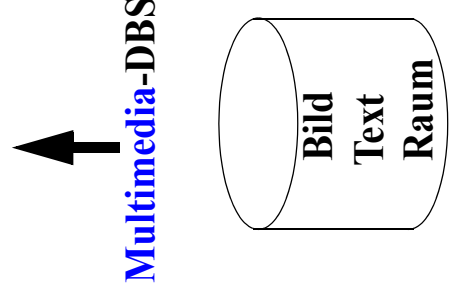
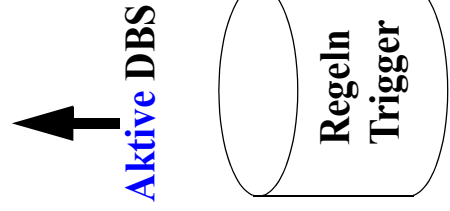
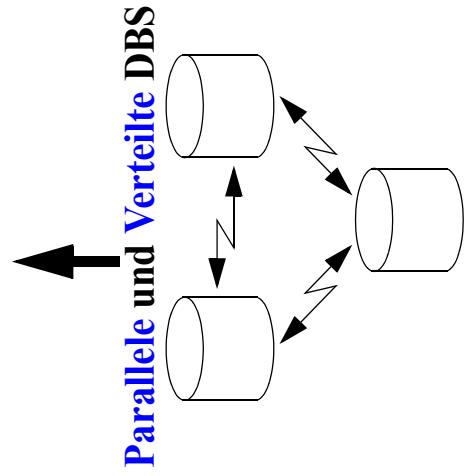
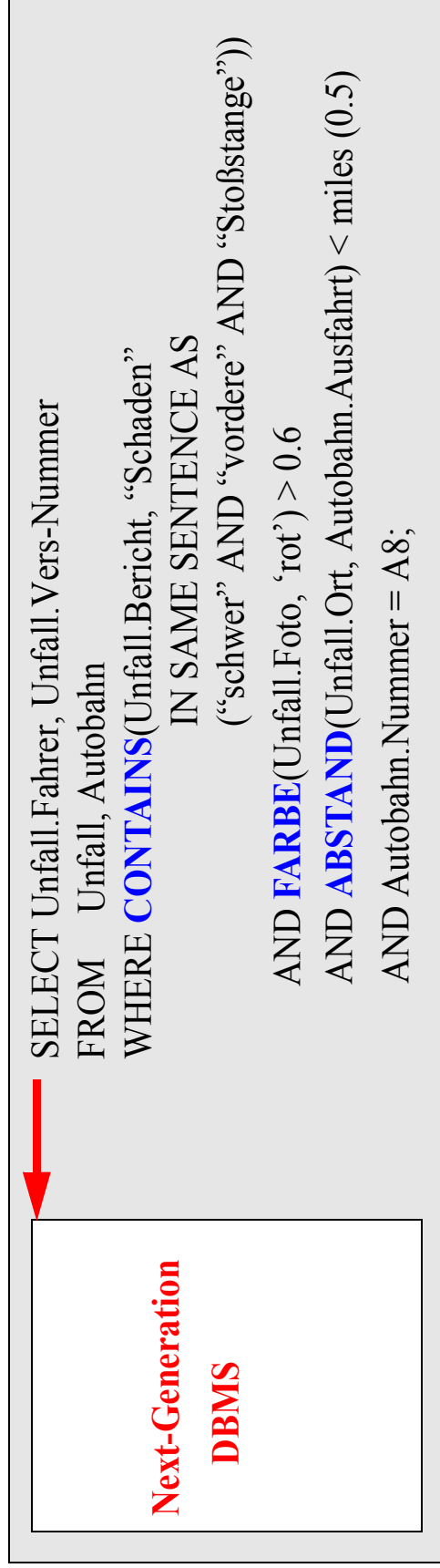
- kein einheitliches Datenmodell
- keine theoretische Grundlage
- Informationsdarstellung durch ‘Tabellen’ und ‘Verbindungen’
- Bindung an Zugriffspfade – Symmetrie des Datenmodells?
- Grad an Datenunabhängigkeit?

➔ **Objekt-relationale Datenmodelle wollen die Vorteile beider zugrundeliegender Modelle vereinigen**

WWW-basiertes Verarbeitungsmodell ... Transaktionsverarbeitung ... Client/Server-Verarbeitungsmodell ... Objektorientiertes Verarbeitungsmodell

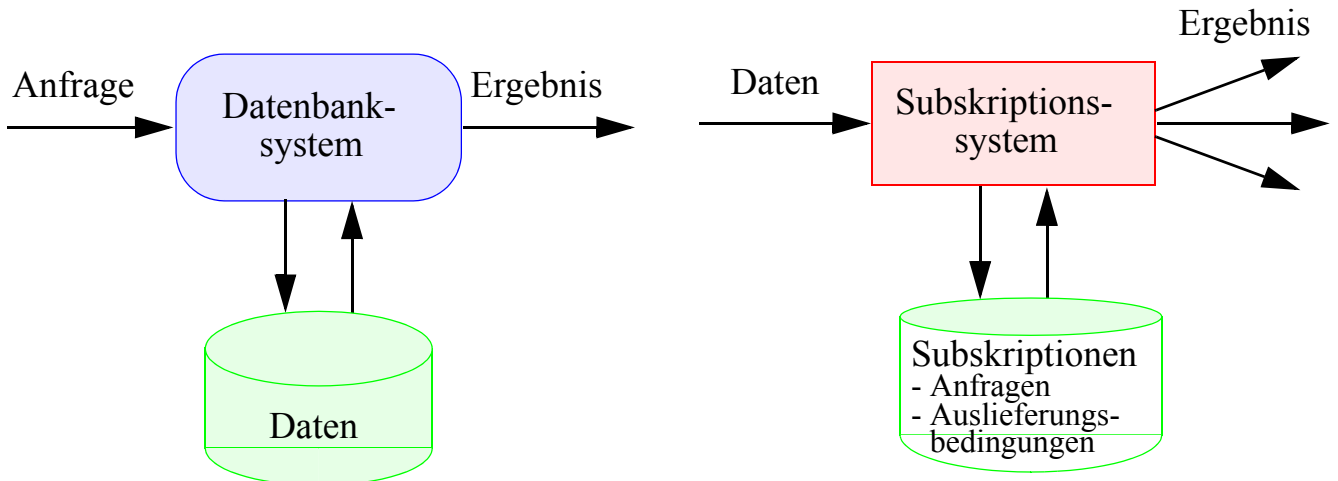
Middleware und Componentware

The Big Picture



Ein weiteres Paradigma – „Alles fließt (panta rhei)⁵“

• Vertauschte Rollen



- statt Auswertung von gespeicherten Daten Filterung, Verknüpfung und Transformation von Datenströmen
- zentrale Bedeutung für die individuelle Informationsversorgung, insbesondere bei einer immer weiter fortschreitenden Verwendung vieler kleiner und damit mobiler Endgeräte

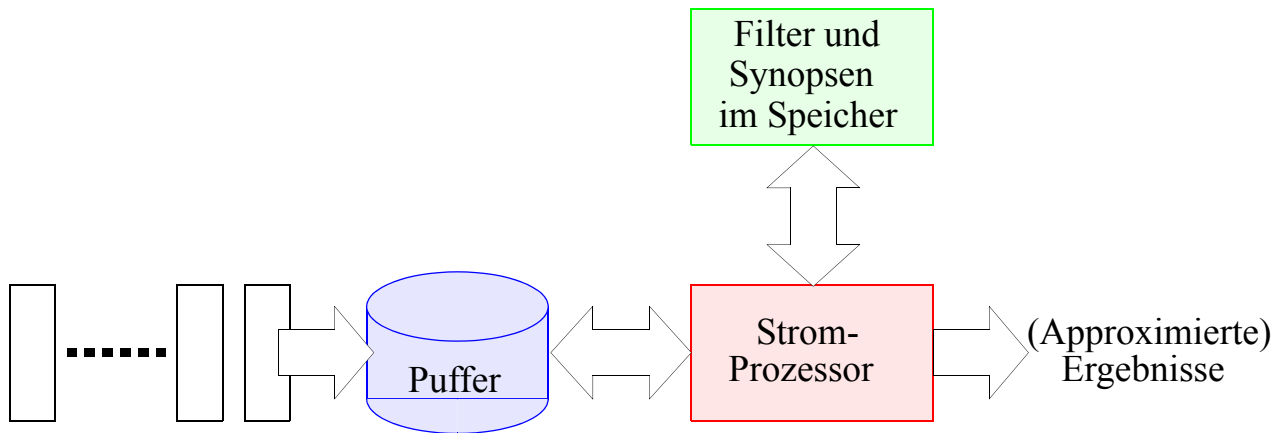
• Wichtige Unterschiede

Eigenschaft	Datenbankbasierte Informationssysteme	Subskriptionssysteme
Verarbeitungs-charakteristik	zustandsorientiert (globaler Zustand)	konsumorientiert (evtl. lokaler temporärer Zustand)
Anfragesemantik	isolierte Anfrage	stehende Anfragen ('standing query')
Zugriffscharakteristik	systemzentriert ('row-set-model')	dokumentenzentriert ('document-model')
Auswertungssemantik	komplexe Analysen	informativ, Auslöser detaillierter Analysen
Schemaaspekt	Existenz eines globalen Schemas	Zugriff auf lokale Schemata der partizipierenden Datenquellen

5. fälschlicherweise Heraklit zugeschriebene Formel für sein Weltbild

Datenströme – Auswertungsmodell

- Daten strömen und sind nicht in einer DB gespeichert⁶

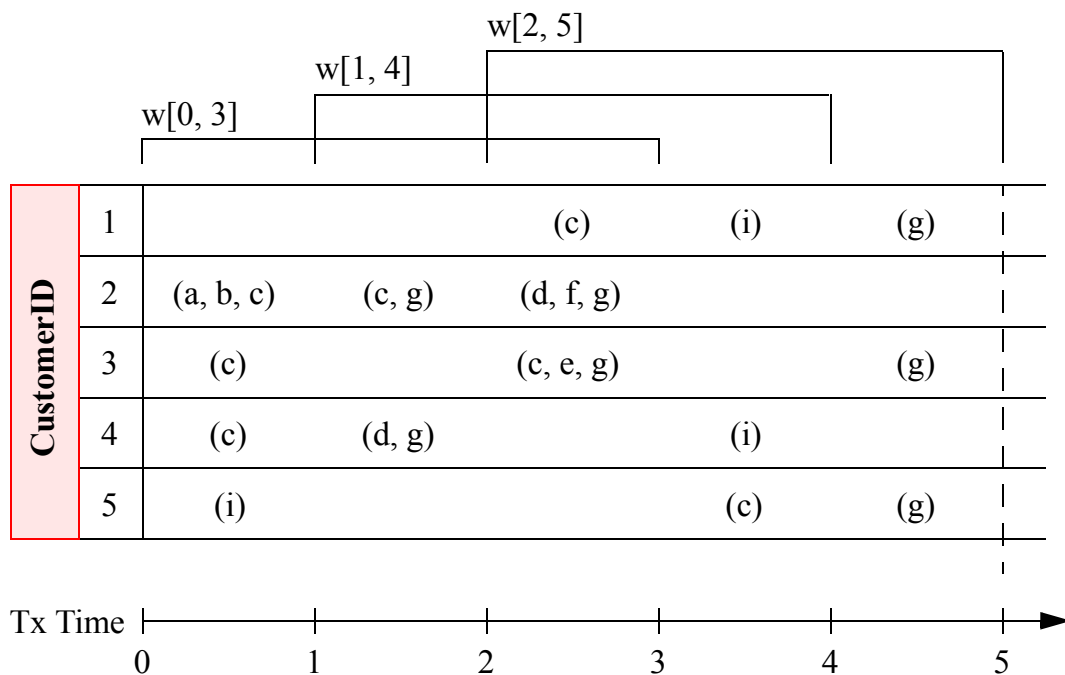


Datenströme

- Analyse von flachen Datenströmen (z. B. sensorgeneriert)
- strombasierte Analyse von XML-Dokumenten
- Wie funktionieren Selektionen und Joins?

- Beispiel für einen Online-Transaktionsfluß

- Warenkorb-AW: Kunden kaufen Waren, die als Ströme zu analysieren sind
- Bestimmung der Häufigkeit von Mustern (frequent pattern identification) ist offensichtlich sehr schwierig!



6. Aktuelles Forschungsthema im Rahmen der dynamischen Informationsfusion

A Dozen Long-Term Systems Research Problems (J. Gray)⁷

1. Scalability:

Devise a software and hardware architecture that scales up by a factor for 10^6 . That is, an application's storage and processing capacity can automatically grow by a factor of a million, doing jobs faster (10^6 x speedup) or doing 10^6 larger jobs in the same time (10^6 x scaleup), just by adding more resources.

2. The Turing Test:

Build a computer system that wins the imitation game at least 30% of the time.

3. Speech to text:

Hear as well as a native speaker.

4. Text to speech:

Speak as well as a native speaker.

5. See as well as a person:

Recognize objects and motion.

6. Personal Memex:

Record everything a person and hears, and quickly retrieve any item on request.

7. World Memex:

Build a system that given a text corpus, can answer questions about the text and summarize the text as precisely and quickly as a human expert in that field. Do the same for music, art and cinema.

7. J. Gray is the recipient of the 1998 A. M. Turing Award. These problems, many related to database systems, are extracted from the text of the talk J. Gray gave in receipt of that award.

A Dozen Long-Term Systems Research Problems (J. Gray) (2)

8. TelePresence:

Simulate being some other place retrospectively as an observer (TeleObserver): hear and see as well as actually being there, and as well as a participant, and simulate being some other place as a participant (TelePresent): interacting with others and with the environment as though you are actually there.

9. Trouble-Free Systems:

Build a system used by millions of people each day and yet administered and managed by a single part-time person.

10. Secure System:

Assure that the system of problem 9 only services authorized users, service cannot be denied by unauthorized users, and information cannot be stolen (and prove it).

11. AlwaysUp:

Assure that the system is unavailable for less than one second per hundred years – 8 9's of availability (and prove it).

12. Automatic Programmer:

Devise a specification language or user interface that:

- (a) makes it easy for people to express designs (1,000x easier),
- (b) computers can compile, and
- (c) can describe all applications (is complete).

The system should reason about application, asking questions about exception cases and incomplete specification. But it should not be onerous to use.