1 Introduction

1.1 Databases vs. files

1.2 Basic concepts and terminology

1.3 Brief history of databases

1.4 Architectures & systems

1.5 Technical Challenges

1.6 DB lifecycle

References: **Kemper / Eickler chap. 1**, Elmasri / Navathe chap 1+2,
and "Intro" of most DB books

Freie Universität Berlin

1.1 Databases Systems versus File Based Processing

Example

Administration of courses, lecturers, rooms…
in a university ... KVV ;)

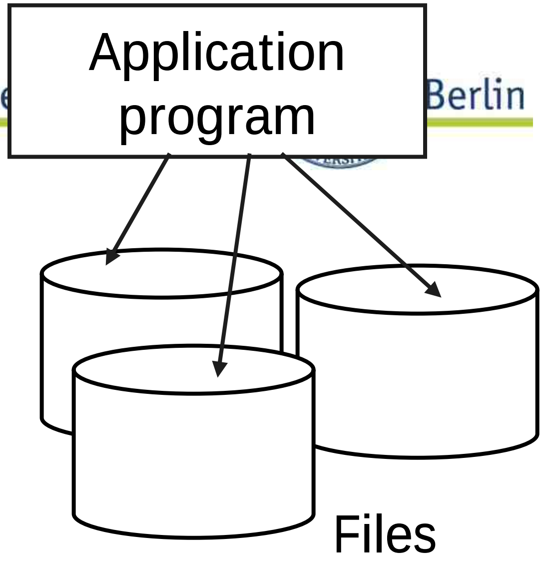
Typical operations:

- "Find all my courses in summer term 2010"
- "Find a room with capacity >20 Friday 8 am"
- "Calculate mean number of courses for the students"

Typically **interactive** and **batch** applications

Why Database systems?

Reading and Writing Random Access Files in Java (taken from Java API)



read

public int **read**(byte[] b, int off, int len) throws IOException

Reads up to len bytes of data from this file into an array of bytes.
This method blocks until at least one byte of input is available. Although RandomAccessFile not a subclass of InputStream, this method behaves in the exactly the same way as the InputStream.read(byte[], int, int) method of InputStream.

Parameters:

b - the buffer into which the data is read.
off - the start offset of the data.
len - the maximum number of bytes read.

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the file has been reached.

Throws:

IOException - if an I/O error occurs.

More than 30 low level operations

Abstraction

What is an appropriate language to manipulate data?

SELECT c.titel, c.hours
FROM Courses c, Lecturers e
WHERE c.lecturer = e.id **AND** e.name = "HS"
AND c.sem = "SoSe2010" .

Result: a table with 2 colums

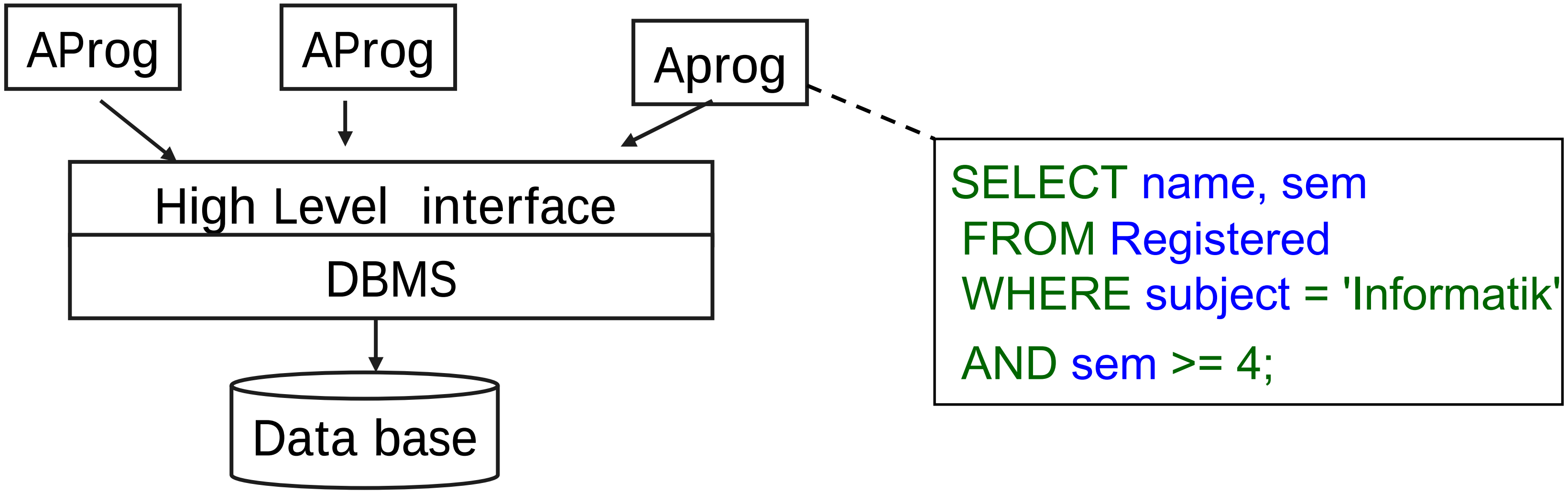
titel	hours
DBS	5
TAS	4
...	

Relational database Systems:

Definition and manipulation of data tables

File system versus DBS

Why database systems?



DBS provide an **abstraction from the physical representation** of data and **from the implementation of operations** (on data)

Files versus Database: differences

Freie Universität Berlin

- Application oriented read/write interface, **high level access**
- Database has it's own **data description** (!) - the **schema**

- More **secure** access
- **Concurrent access** to data
- **Fault tolerance**

Nonfunctional characteristics

1.2 Basic Concepts and Terminology

Freie Universität Berlin

1.2.1 Data independence

Important term!

- Guiding principle: introduce **levels of abstraction**
- Application program should be **independent of physical organization** of data

e.g. hash, B-Tree or sequential access to records should be transparent to the program (ignoring performance impacts)

Def.: Physical Data independence

Application programs are not compromised when storage structure is changed

Basic Abstractions

Data Independence (cont)

Example

Suppose participation in exams has to be introduced for each student in the university database

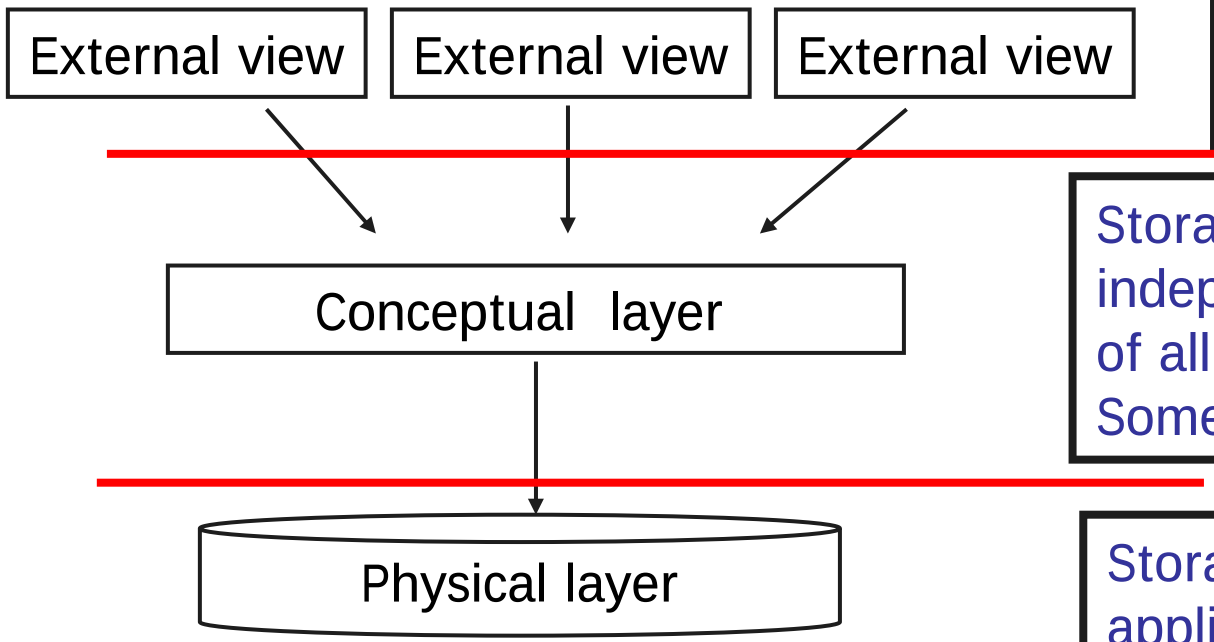
Goal: existing application programs should not need to be changed, except when logically necessary. (e.g. grades for exam presupposes partipation)

Def.: Logical data independence

Application programs are not compromised by changes of the schema (*if possible*)

ANSI/X3/SPARC Architectural Model

"separate physical aspects from logical data structuring from individual user (application) views of the data"



Different applications have different views of storage and data model

Storage and data model independent definition of all data in the database. Sometimes called *logical layer*

Storage model hidden from applications

How to specify a database?

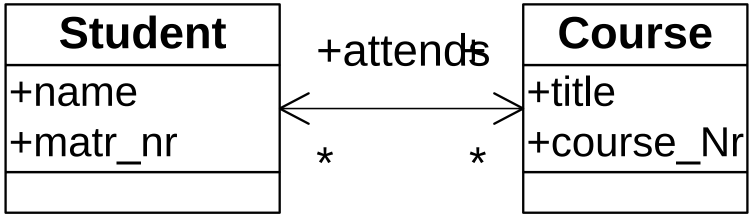
Freie Universität Berlin

Important terms!

Conceptual model

Describes high-level concepts in DB design models subset of real world.

Entity relationship model, (or UML: Universal Modelling Language)

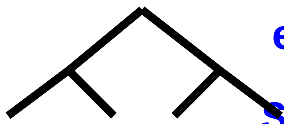


Logical Data Model (DM)

```
CREATE TABLE student
   (name CHAR(..),
    MatrNr NUMBER,....)
```

Physical (data) model

Declarative ("logical") description of implementation schema



e.g.

Search Tree

`CREATE INDEX …`

Def.: Database schema:

Formal description of some part of reality in terms of the data model (e.g. tables)

Schema defined on different levels: logical, physical, external

Schema: - specifies content of database on a **type level**,
- in most cases: schema **separate from data** "schema is first class object"
- may be changed over time, but basically static.
- does not exist for files (hidden in program)

Def.: Database

Set of data objects conforming to a given database schema

Database: **dynamic**, time variant
DB schema: basically **static**.

Important aspect:
Primitives for
- schema specification
- database operations

$\Rightarrow$ **Data Model**

1.2.2 Data models

Freie Universität Berlin

Def.: Data Model

is a <u>language</u>

- for defining the schema
  (Data Definition Language DDL)

- for accessing and updating the DB
  (Data Manipulation Language DML)

Important term!

Most important data model today:
Relations (tables) and SQL (or relational algebra)

FName	Name	title	phone
Bob	Kunz	Prof	33101
Cathy	Hinz	Dr.	33700

← **schema** (the header row)

← **data**: tables (set of rows) (the data rows)

The Relational Model

1970: **Relational model** [E.F. Codd: The Relational Data Model] -> reader

Author

FName	Name	Email
Tina	Hunt	hunt@…
Anna	Katz	katz@…
Carl	Maus	piep@…

Lecturer ← Table name

FName	Name	title	phone
Bob	Kunz	Prof	33101
Cathy	Hinz	Dr.	33700

Course

Title	Lecturer	room
DBS	Hinz	05
Compiler	Katz	03
Seminar	Hinz	05

….

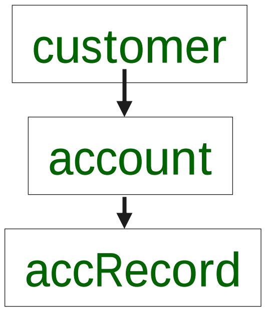
- All data represented as tables,
- Schema ("Metadata") separate from data

since 1980: RDBMS everywhere

Legacy data models (1)

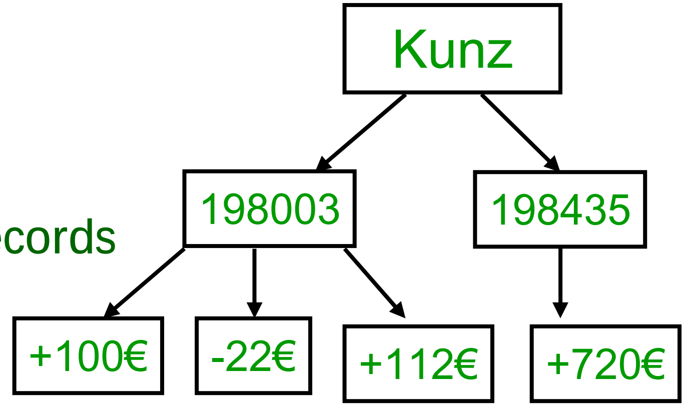
Hierarchical data model: hierarchies of record types

schema



A bank customer has one or more accounts, für each account, there are 0 or more accounting records

data

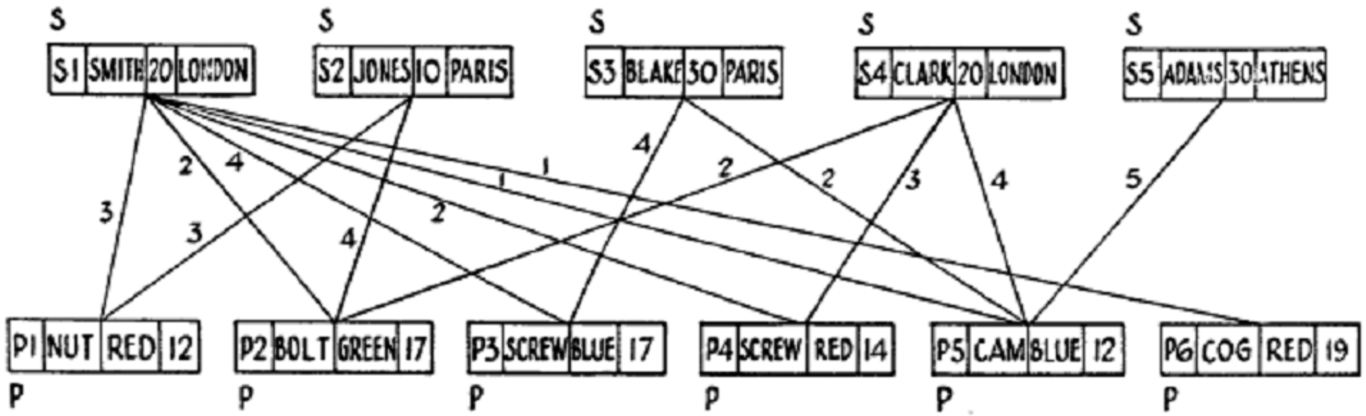


Still in use: IMS (Information Management System), a mainframe oldie.

Legacy data model (2)

Network data model
("CODASYL") : graph like
data structures (see reader

Instances



Schema

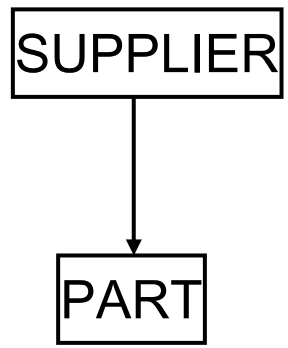


Figure 1.1.3: the suppliers-and-parts data model (network approach)

Example by Codd / Date, ACM SIGFIDET 1974

Other data models (1): XML

Pre-XML representation of data:

```
“PO-1234”,”CUST001”,”X9876”,”2”,”14.98”
```

XML representation of the same data:

```
<?xml version="1.0"?>
<PURCHASE_ORDER>
      <PO_NUM> PO-1234 </PO_NUM>
      <CUST_ID> CUST001 </CUST_ID>
      <ITEM ItemNum ="x9876">

        < QUNTY > 2 </ QUNTY >

        <PRICE> 14.53 </PRICE>

      </ITEM>
</PURCHASE_ORDER>
```

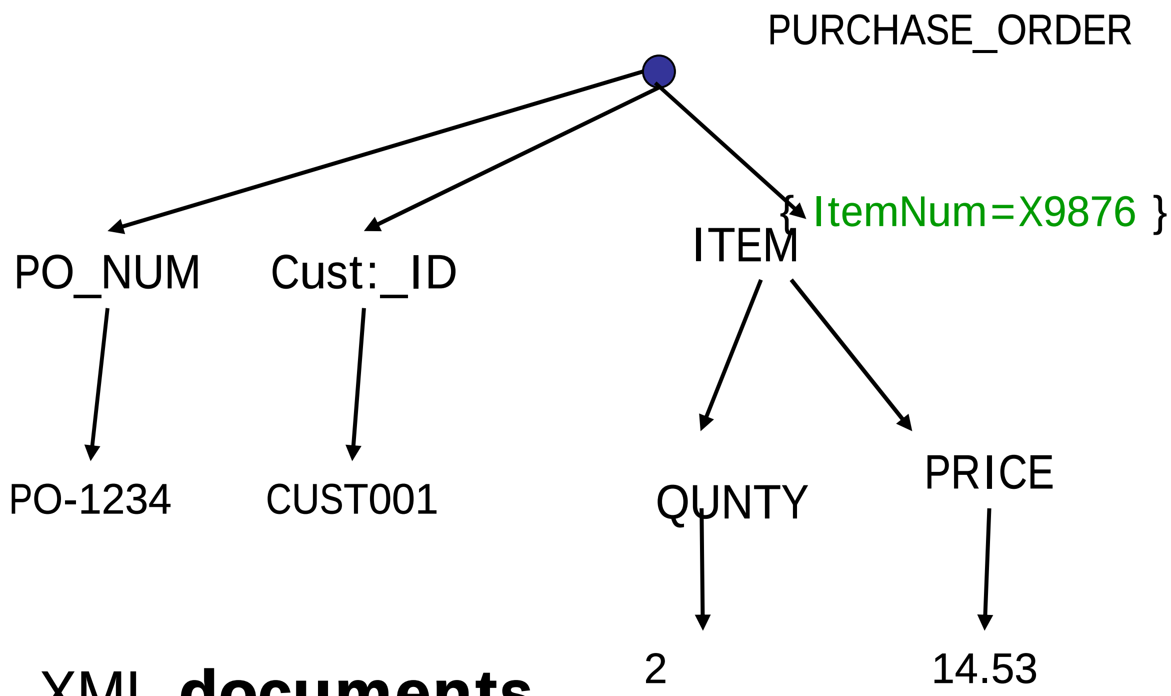
Prologue

Attribute

Elements

XML example

Graphical representation of XML data



For those who do not know what XML is: learn the basics here.

XML **documents**

- **tree** structured
- **data and metadata in the same document** (as opposed to RDBS)

Other Data models (2)

- **RDF** (Ressource description Framework)

- There is a set of **Nodes** (call it N).
- There is a subset of N known as the **PropertyTypes** (call it P).
- There is a set of **3-tuples** called **T**, whose elements are informally known as properties. The first item of each tuple is an element of P, the second item is an element of N and the third item is either an element of N or an atomic value (e.g. a Unicode string).

(Core Data Model of RDF, see http://www.w3.org/TR/WD-rdf-syntax-971002/)

- **Object oriented (data) model**? ... try to define.

Other Data Models (3)

Lightweight database systems

– Key value stores ("schema less DBS")

Example (couchDB):

```
{ "_id":"biking",
 "_rev":"AE19EBC7654",
 "title":"Biking",
"body":"My biggest hobby is mountainbiking.
        The other day...",
 "date":"2009/01/30 18:04:11"
}
```

– basic ideas:
  very simple schema language ("**key:value**"),
  very efficient access by key,
  offload most correctness guarantees to application.

Data model

Caveat:

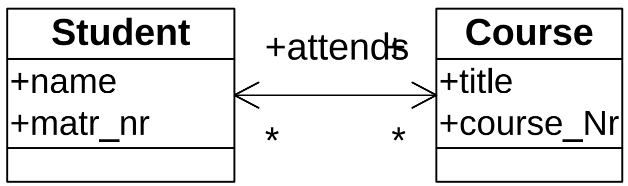
- **Data Model** is a **language**.
- Data Model is **not** the result of **modeling some reality**
- This process is called **data modeling**
- The result is the DB schema

Data model

Relational Data model:

~"Use tables to represent your data"

Schema (conceptual)



Database

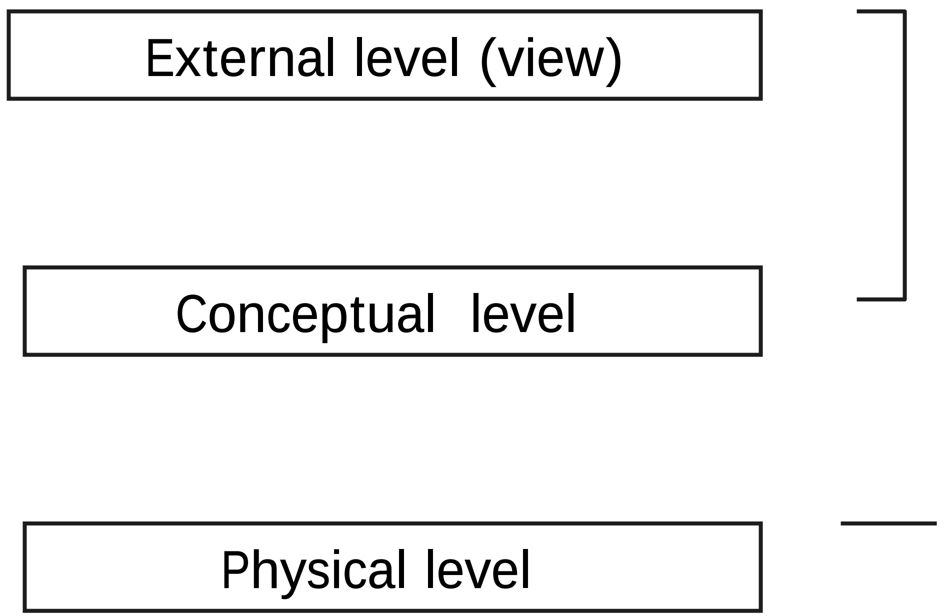
DBS	Hinz	05
Compiler	Katz	03
Seminar	Hinz	05

…

1.2.3 Data base languages

Freie Universität Berlin

Different language levels for relational (tabular) DBS
all covered by **SQL (Structured Query Language)**



Data Definition (DDL) and Manipulation Language (DML)

- Define logical data structures (schema)
- Query database

Data Administration Language

- Define access path
- Adjust tuning and other parameters

SQL and Programming languages

Freie Universität Berlin

Programming Languages

– SQL is an interactive language

– Most applications don't allow users to use SQL directly but have their own GUI (e.g. a forms based web interface)

– How do these applications talk to the DBS?

Embedded SQL

DBS define an Application Programming Interface (API) which is basically a standardized interface for calling the DBS from a program with the SQL-command to be executed and for transferring the result data.

Most popular: **Embedded SQL / C** and **JDBC** (Java)

1.3 History at a glance

- Business Data Processing as the driving force for DBS development
- ~ 1965 File system approach to data management leads to chaos.
- **What are the right abstractions? ⇒ data model**
- 1970: Tables!
  (*Codd's seminal paper*)
- 1973: Research prototypes for Relational DBS, *Transactions*
- 1980: RDBMS everywhere,
  Distributed DBS

History (cont)

- 1990: **Object orientation** ⇒ OO data model and OODBMS ⇒ Object-Relational systems

- 1995: Wide scale distribution, **WEB**

- 1997: Semistructured data, Image DB, ... , XML / DB

- 2000++ Mobility and DBMS
- 2005++ Unstructures Data – e.g. text. Querying text???

- Automated **Object-relational mapping**: only objects in the program, don't care about relations

1.4 Architectures and Systems

Legacy systems

Information Mangement Systems (**IMS**), hierarchical systems by IBM

Universal Data Store (**UDS**) , network system by Siemens

The dominating Relational DBMS

Oracle

Postgres

MySQL

SQL-Server / Microsoft

Sybase

DB2 / IBM, Informix

Adabas (Software AG)

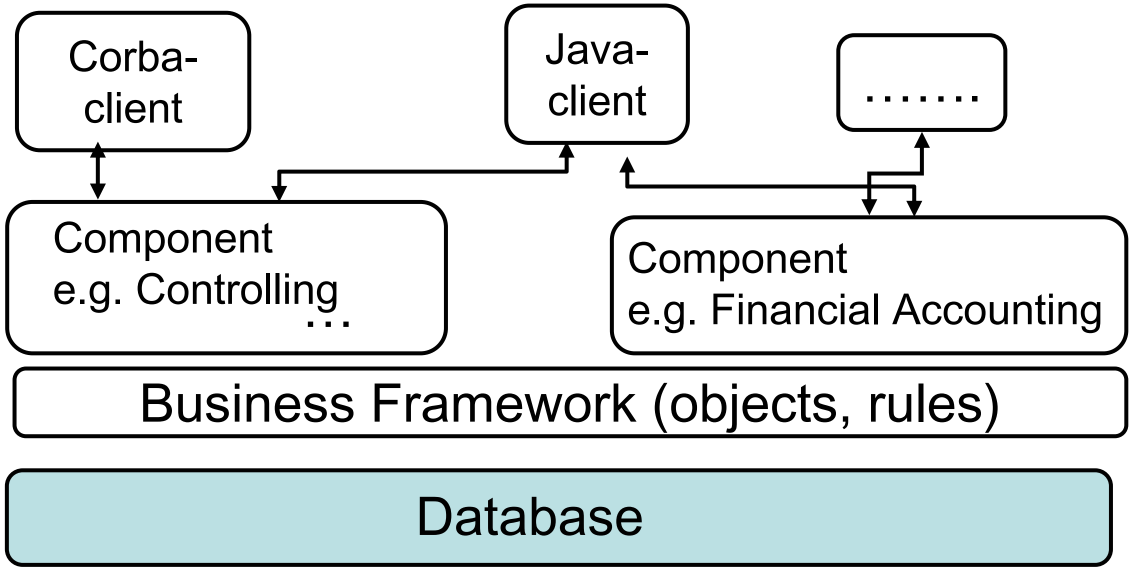
personal, low cost desktop DBS: **MSAccess**

Java "persistence" related DBS: **Derby**, ...

Integrated systems

Freie Universität Berlin

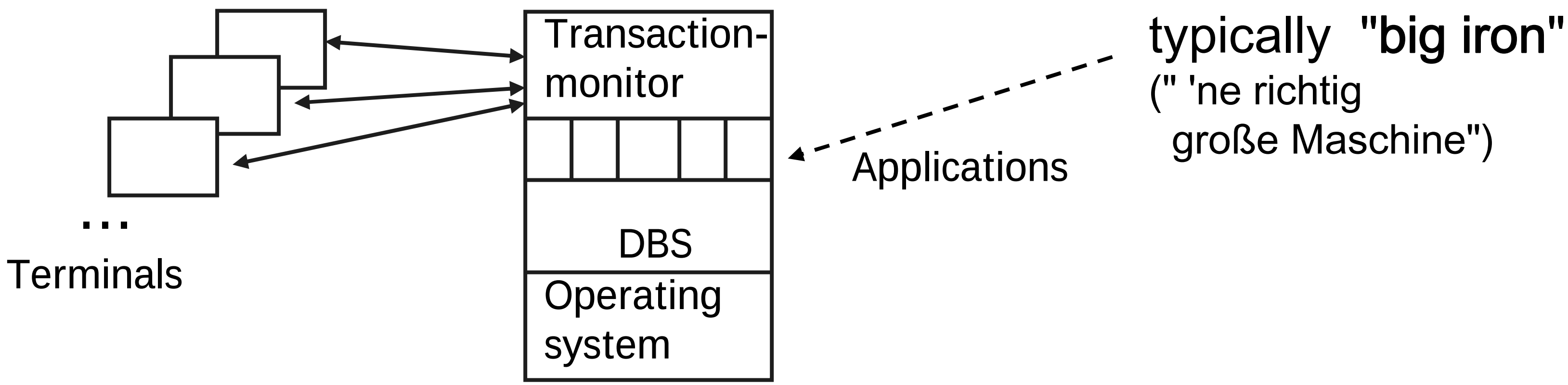
More and more **integration with application** software, e.g. SAP R3 uses Oracle (mostly) behind the curtains



ERP system
(Enterprise Ressource Planning)

Mainframe

- **Mainframe architecture**



– **Transaction monitor** queues requests, schedules application programs (usually simple application logic)

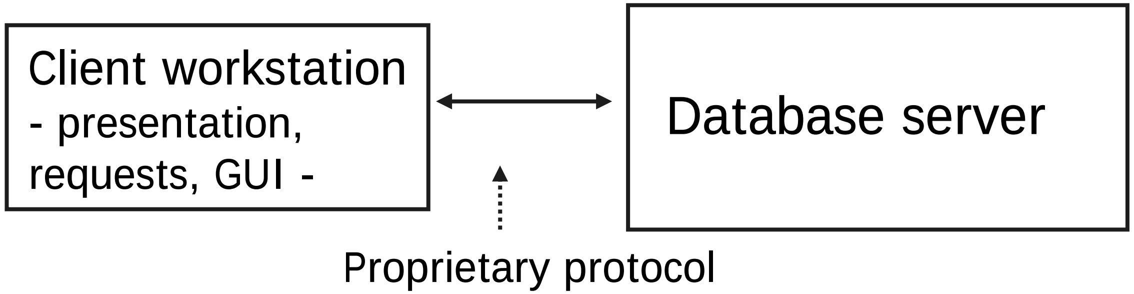
Still in use today, e.g. flight reservation systems

very efficient, but expensive hardware

2-tier Architecture

Freie Universität Berlin

Two-tier architecture



typically used with 4GL ("Fourth Generation Languages")
i.e. languages for easy development of simple form-based application and reports.

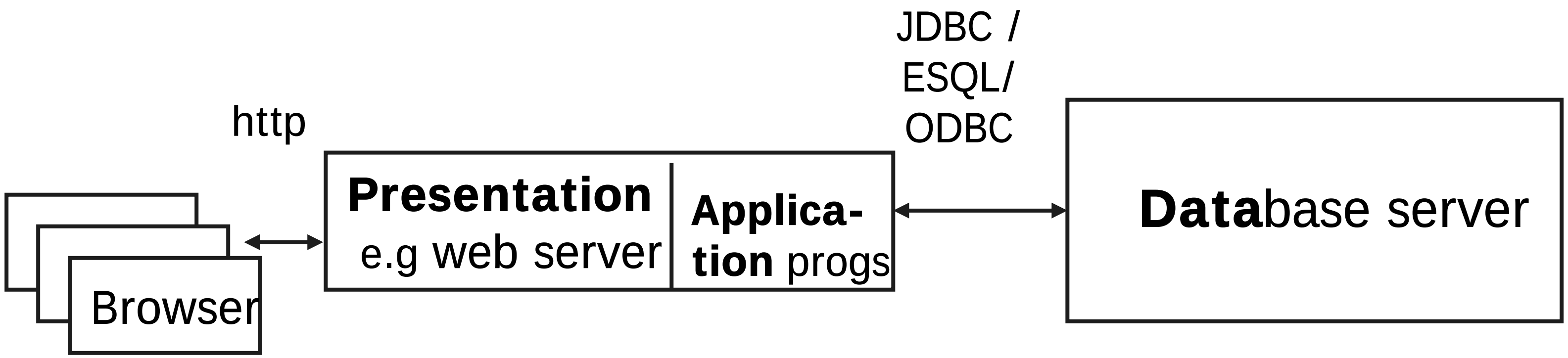
Transaction support through database system

Used in medium size applications

Three-tier Architecture (1)

Application oriented architecture

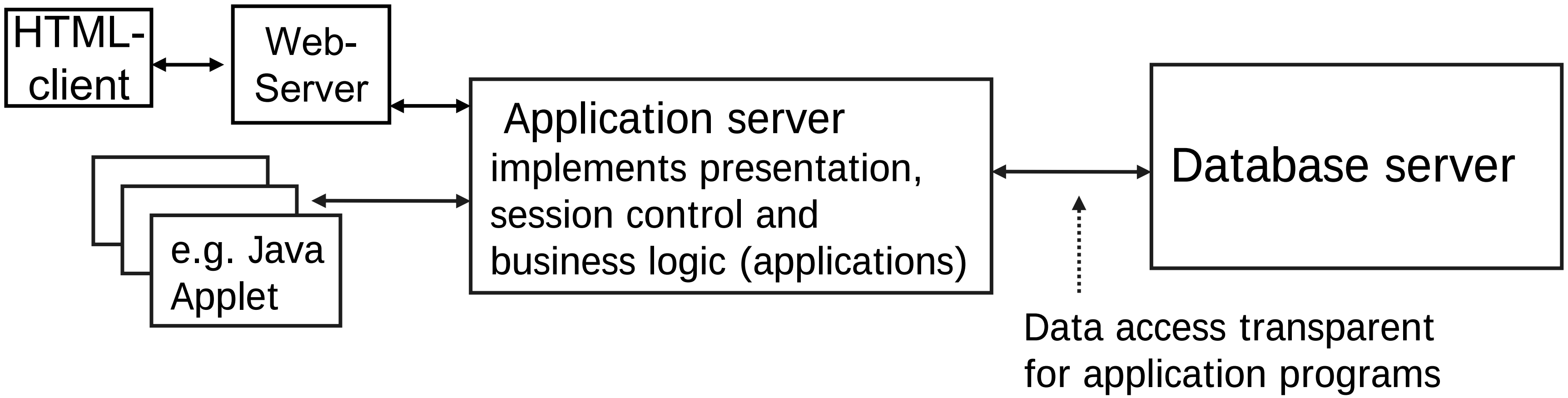
separation of presentation, application logic and DB access



e.g. CGI or Servlet application running under control of a web server

Three-tier Architecture (2)

Middle tier: framework for implementing business logic and business objects



Particularly useful with automatic object-relational mapping between database (relational) and programming language (object oriented)

1.5 Technical challenges

Freie Universität Berlin

Operational requirement:

The DBS should never do anything which destroys the *consistency of database and modeled reality* (called **integrity**)

Example:
Transfer 100 $ from one account a1 to another one a2. Several steps are required: reading the value of a1, decrease the amount (100 $), write a1, increase the value of a2 by the amount.

Main technical issue:

Execution of operations must **guarantee correctness properties**

Technical challenges

Freie Universität Berlin

Operational requirement:

No interference of operations of different users

Example: Auction system. Two independent bidders A, B
read highest bid h, B's bid : h+a , A's bid h+b
B's bid is lost even if h+a < h+b
A and B are the programs executing the bids for human users

Technical Challenges

Synchronisation of independent DB-users:

How to **avoid conflicting read / write access** ?

⇨ concurrent programming

But DB have many resources: each record is a resource – there may be millions (*) of them

⇨ **Synchronization of thousands of concurrent operations** ?

(*) Wal-Mart: 200 Mio transaction / week = 300 TA/sec – 24/7

source: The Economist Feb 27,2010

Technical challenges

Fail-safe operation

Example: System crash when writing a block with account data on disk. DB must not be corrupted

System failure should **not corrupt database** state

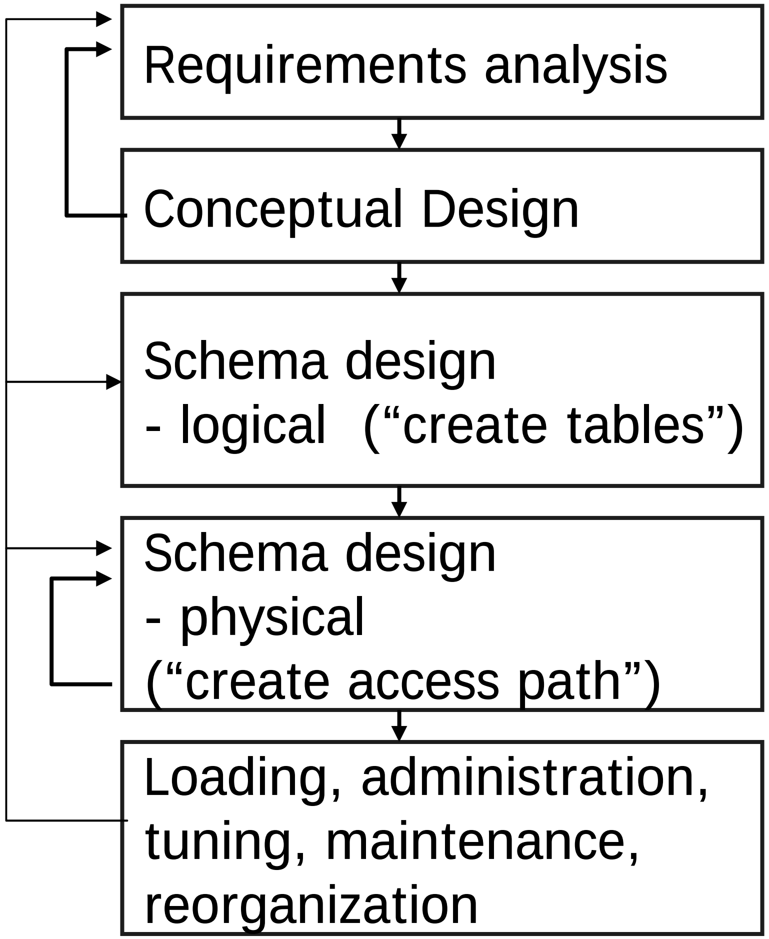
Efficiency

Hundreds of clients active on the same DB,
Hundreds or thousands operations / sec,
Response time requirement in interactive environment: < 3 sec

Data security

Access by unauthorized users might be a disaster

1.6 Lifecycle



System analyst
DB designer
Application programmer

Application programmer
DB administrator

DB administrator

Compare: **Lifecycle of HW ~3 years**
Software ~ 5 years,
Data 30 years !?

Summary

- Database $\neq$ Database System
- Database: **data** and **metadata** (schema)
- Data model: high level **data definition** and **data manipulation language**
- Relational Data Model (RDM) / SQL
- Two- /Three-tier-architecture
- Technical requirements
  - Concurrency
  - Fault-tolerance
  - Integrity
  - Efficiency
- Life cycle