1 Introduction

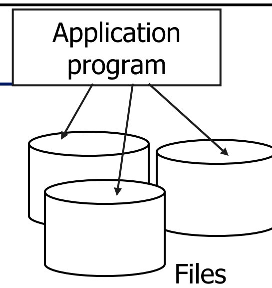
References: Kemper / Eickler chap. 1, Elmasri / Navathe chap 1+2
and "Intro" of most DB books

1 Introduction

1.1 Databases Systems versus File Based Processing

- Example
  - Administration of students, courses, lecturers, rooms… in a university
  - Typical operations:
    - "Find course data of a particular student"
    - "Record the grades of a particular student"
    - "List the average teaching load per lecturer over the last two years"
- Requirements
  - Flexible Querying
  - Multiuser support
  - No loss of data
  - Data consistency

Introduction



- 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 is 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.

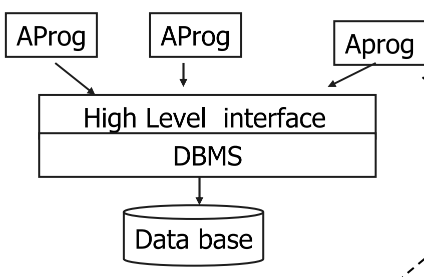
Returns:
the total number of bytes read into the buffer, or -1 if there is no more data because the end of the file has been reached.

Throws:
IOException - if an I/O error occurs.

More than 30 low level operations

Introduction: File system versus DBS

- Why database systems?



Abstract from physical representation and implementation of operations

SELECT title, hours
FROM Registered
WHERE studName='Brewer'

- Well defined interface, high level access
- Database comprises it's own description (!) - the schema
- Concurrent access (on record or lower level)
- More secure access
- Fault tolerant

- But sometimes DBS are an overkill....

1.2 Basic Concepts and Terminology

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)

⇨ Physical Data independence

Basic Abstractions

Data Independence (cont)

- An application program A (or an end user) should see the data used independent from data in the DB not used by A

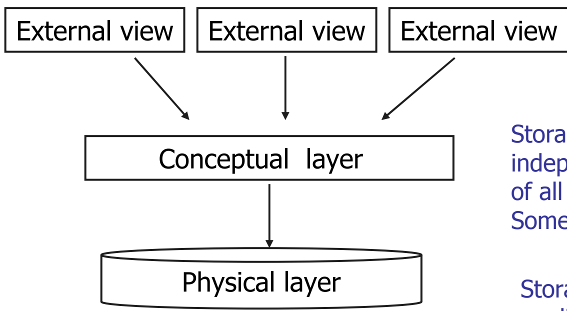
e.g. in a university DB the assignment of students to courses should not be influenced, say, by a new format of zip-codes ("Postleitzahlen") in the student's addresses

or: introduction of fees should not require changes of the exam administration organization

⇨ Logical Data independence

3-Schema-Architecture

- 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

3-Schema-Architecture

- Example: External Model

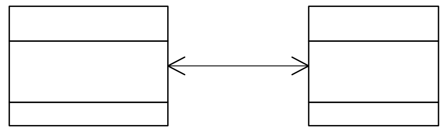
  Imagine a hospital information system managing data about patients, doctors, medication etc. For each patient, data about all diagnostic and the results are recorded.
  – Doctors must be able to read the diagnostics
  – The administration must not read the results, but which kind of diagnostics has been performed (for accounting purposes)
  $\Rightarrow$ Two user groups with different views

- Advantages
  – Access protection (data privacy)
  – Higher degree of independence of application programs from data

DBS concepts & architecture

Important terms!

- Conceptual model
  - Describes high-level concepts in DB design
  - Models subset of real world
  - entity relationship model



- Physical (data) models
  - Logical description of implementation schema

`CREATE INDEX …`

- Data model …

`CREATE TABLE …`

1.2.2 Data models

Important term!

Data Model
language for defining types
- the DB schema -
and for accessing and updating the DB

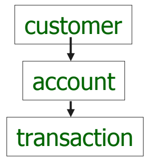
- Most important data model today:
  relations (tables) and SQL (or relational algebra)

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

- Object oriented DM if operations are taken into account, not just data

Legacy data models

- Hierarchical data model: hierarchies of record types e.g.



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

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

- Network datamodel (“CODASYL”) : graph like data structures (see reader)

1.2.3 Database and database schema

DB schema…

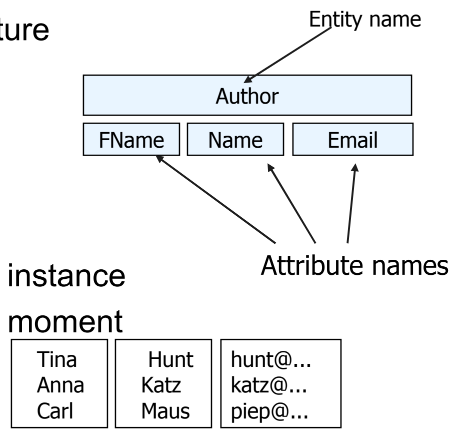
- …is a model of the static aspects of some part of reality
  - e.g: customers, accounts, transactions in a bank
  - students, lectures, profs, course enrolments in a university   ….. etc
- But schema is not called “data model” (!)
- DB schema is the type definition of the database
- 3-schema architecture:
  - physical schema,
  - a conceptual (or logical) schema
  - external schemas (schemata)

Database: set of instances (objects) conforming to schema

Data model vs Database Schema

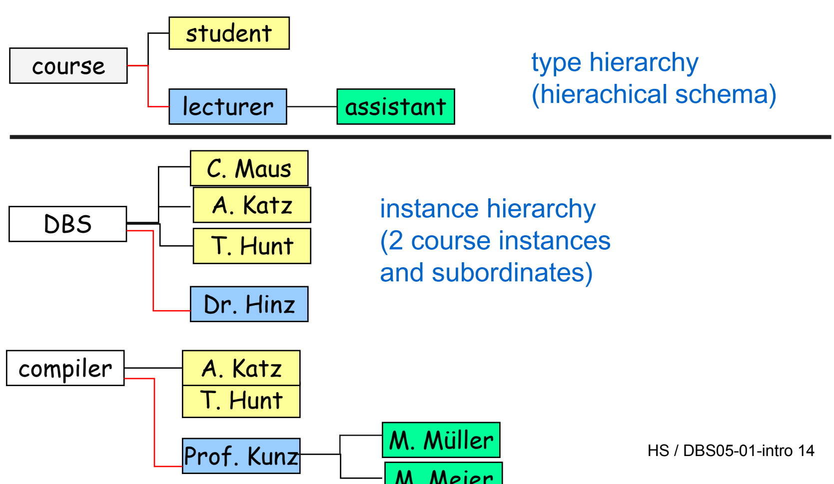
Important terms!

- Database schema
  - Description of DB structure
  - Stored as meta-data
- Database (state)
  - given by the database instance
  - Data in DB at particular moment
  - Extensional
  - Must conform to the schema



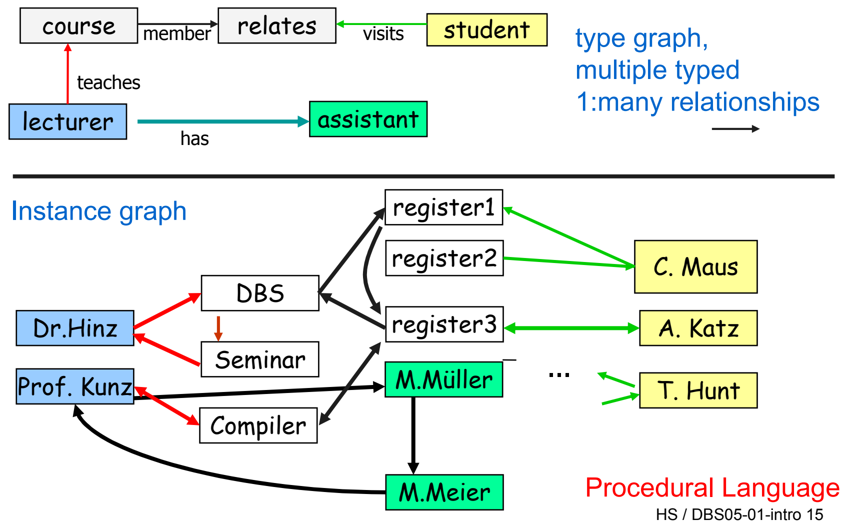
1.2.4 Standard data models

- Hierarchical model (e.g., IMS):
  Data relationships in trees



Data models: Network

- Network model (Codasyl): Data relationships as graphs



Data models – the relational model

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

Author

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

Lecturer

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

schema

instances

Course

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

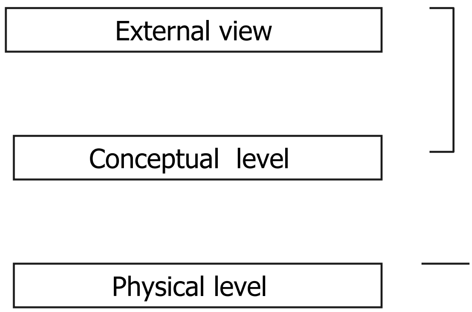
....

All data represented as tables

- 1980: RDBMS everywhere

1.2.5 Data base languages

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

- 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)

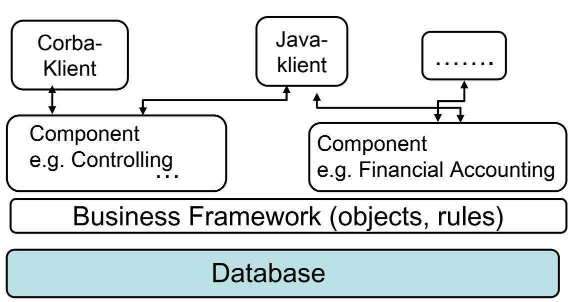
- 1990: Object orientation ⇒ OO data model and OODBMS ⇒ Object-Relational systems
- 1995: Wide scale distribution, WEB
- 1997: Semistructured data, Image DB, ... , XML / DB
- 2000++ Mobility and DBMS
- Automated Object-relational mapping: see objects in your program, don't care about relations

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
  - Informix
  - Sybase
  - DB2 / IBM
  - SQL-Server / Microsoft
  - Adabas (Software AG)
  - MySQL (SAP DB)
  - personal, low cost desktop DBS: MSAccess

Integrated systems

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

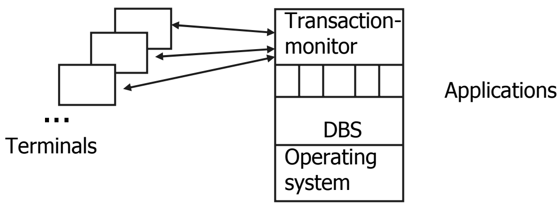


ERP system (Enterprise Ressource Planning)

- New challenges: how to deal with text, pictures, video-streams; mobility, large scale distribution?

Mainframe

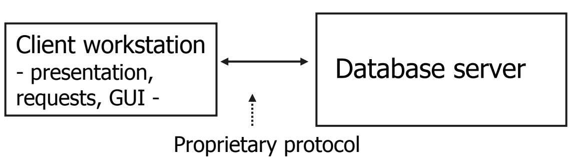
- Mainframe architecture



  - 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

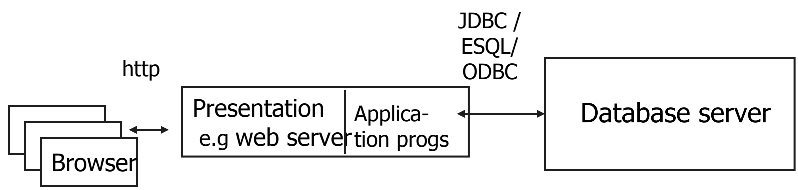
- Two-tier architecture



  - 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)

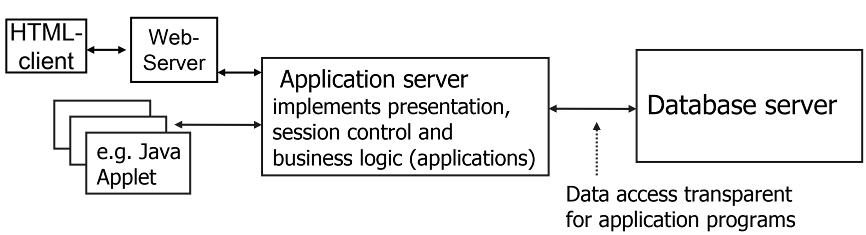
- Application oriented architecture
  - separation of presentation, application logic and DB access



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

Operational requirement:

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

Example:

Suppose you want to 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

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

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

⇨ different technical solutions needed

⇨ Synchronization of thousands of concurrent operations

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

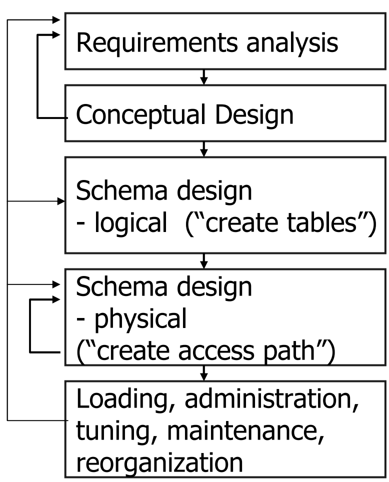
Efficiency

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 up to 30 years

Summary

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