Institut für Informationssysteme
Technische Universität Braunschweig

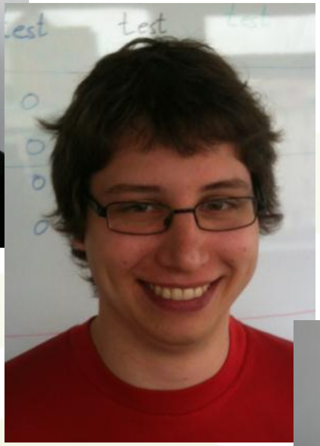
Relational Database Systems 1

Wolf-Tilo Balke
Benjamin Köhncke

Institut für Informationssysteme
Technische Universität Braunschweig
www.ifis.cs.tu-bs.de

0. Organizational Issues

- Who is who ?
  - Wolf Tilo Balke
    - (Lecture, exams)
  - Simon Barthel
    - (Detours, tutorial)
  - Philipp Wille
    - SQL Lab
  - Regine Dalkıran
    - (Office)



- In case of questions, feel free to ask us.

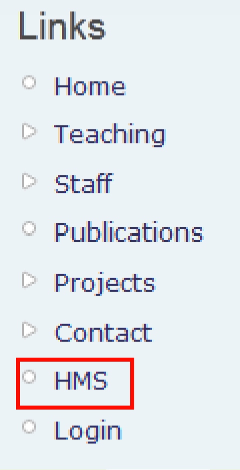
0. Organizational Issues

- Lecture
  - October 18, 2012 to January 31, 2013
  - 15.00–17.30 (including a break)
  - “Integrated” lecture:
    Theory, exercises, and detours
  - 5 credits

- Homework:
  - Weekly assignments
  - … can be downloaded from our website
  - … should be completed in groups of two students

0. Organizational Issues

- Tutorial groups:
  - Led by our teaching assistants
  - Homework discussion
- Homework Management System
  - Please sign up to our HMS System
    - To be found on the ifis homepage
  - Achieved points will be entered in there
- In order to pass this **module** you need:
  - … to achive **50%** of homework points (1 CP)
  - … to pass the exam (4 CP)



0. Why Should You be Here?

- Database system are an **integral part** of most businesses, workflows and software products

- There is an abundance of **jobs** for people with good **database skills**
  - Help yourself to put you into a good position within the job market
  - Prepare for a sunny and wealthy future!

0. Why Should You be Here?

salary.com

0. Why Should You be Here?

Job descriptions also exactly describe this course…

Job Description

Senior Database Analyst
Reviews, evaluates, designs, implements and maintains company database[s]. Identifies data sources, constructs data decomposition diagrams, provides data flow diagrams and documents the process. Writes codes for database access, modifications, and constructions including stored procedures. May require a bachelor's degree in a related area and 4-6 years of experience in the field or in a related area. Familiar with a variety of the field's concepts, practices, and procedures. Relies on experience and judgment to plan and accomplish goals. Performs a variety of complicated tasks. May lead and direct the work of others. Typically reports to a project leader or manager. A wide degree of creativity and latitude is expected.

Alternate Job Titles: Database Analyst III | Level III Database Analyst | Senior Database Analyst

Job Description

Database Administrator Manager
Manages the administration of an organization's database. Analyzes the organization's database needs and develops a long-term strategy for data storage. Established policies and procedures related to data security and integrity; monitors and limits database access as needed. Oversees the design, maintenance and implementation of the systems that manage an internal database. Requires a bachelor's degree with at least 7 years of experience in the field. Familiar with a variety of the field's concepts, practices, and procedures. Relies on extensive experience and judgment to plan and accomplish goals. Performs a variety of tasks. Leads and directs the work of others. A wide degree of creativity and latitude is expected. Typically reports to top management.

Alternate Job Titles: Database Administration Manager | Database Administrator Manager

monster

http://jobsearch.monster.com/

0. Why Should You be Here?

- "**Larry Ellison** is the **highest-paid CEO** of a public company, according to a survey of executive compensation going back 10 years compiled by the Wall Street Journal.
  - With compensation totaling $1.84 billion in the 10-year period ending in May, Ellison, **Oracle Corp**.'s founder and CEO, outdistanced runner-up Barry Diller, CEO of IAC/InterActiveCorp. and Expedia Inc., at $1.14 billion. Apple Inc. CEO Steve Jobs came in fourth with a paltry $749 million."
  - (San Francisco Chronicle, July 28, 2010)

0. Instructional objectives

After successfully completing this course students should be able to

- explain the fundamentel terms of
  - relational database systems,
  - theoretical and practical query languages,
  - conceptual and logical design of databases including normalization,
  - application programming as well as
  - further concepts like constraints, views, indexes, transactions and object databases.

0. Instructional objectives

They should furthermore be able to

- design and implement a database for any specified domain using ER-Diagramms or UML-Diagrams, the Relational Model and SQL-DDL,
- normalize a given relational database schema,
- enhance the database with views, indexes, constraints, triggers and access rights,
- formulate data retrieval queries in SQL and the Relational Algebra and Calculus,
- write programms to access databases using JDBC.

0. Courses at ifis

Everybody should know this!

- Basic course in databases
  - Relational Databases I (Bachelor)
    - What can we do with an DBMS?
    - Conceptual modeling, data retrieval, relational model, SQL, building applications, basic data models
  - SQL Lab (Bachelor)
    - Advanced features of SQL and database programming
    - **Hands-on experience**
  - Relational Databases II (Master)
    - How can we implement a DBMS?
    - Storage models, query optimization, transactions, concurrency control, recovery, data security

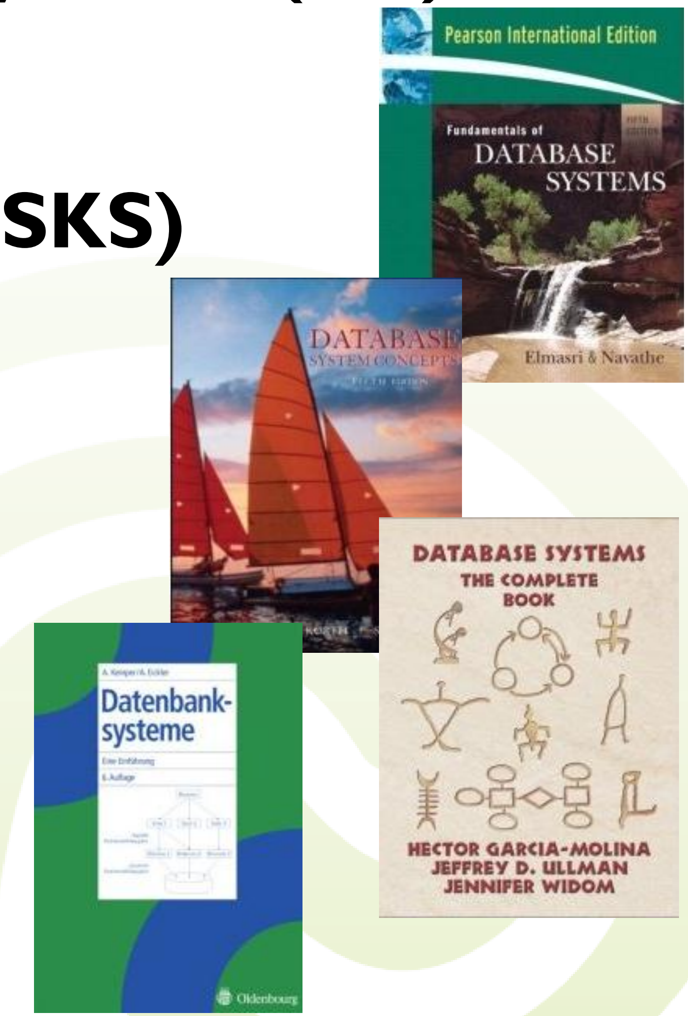
0. Courses at ifis

- Advanced courses in databases (Master)
  - Information Retrieval and Web Search Engines
  - Multimedia Databases
  - Distributed Data Management
  - Knowledge-Based Systems and Deductive Databases
  - Data Warehousing and Data Mining Techniques
  - XML Databases
  - Spatial Databases and Geographic Information Systems
  - Digital Libraries

If you still don't have enough!

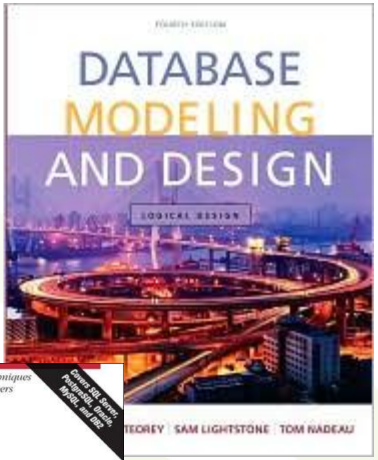
0. Recommended Literature

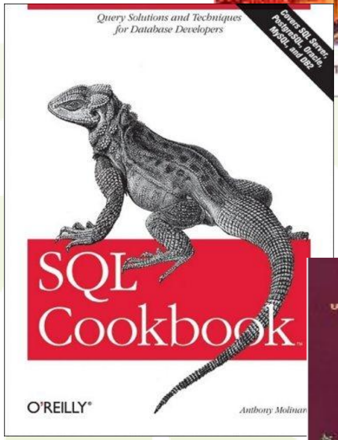
- **Fundamentals of Database Systems (EN)**
  - Elmasri and Navathe
  - Addison-Wesley
- **Database System Concepts (SKS)**
  - Silberschatz, Korth, and Sudarshan
  - McGraw Hill
- **Database Systems (GUW)**
  - Garcia-Molina, Ullman, and Widom
  - Prentice Hall
- **Datenbanksysteme (KE)**
  - Kemper, and Eickler
  - Oldenbourg

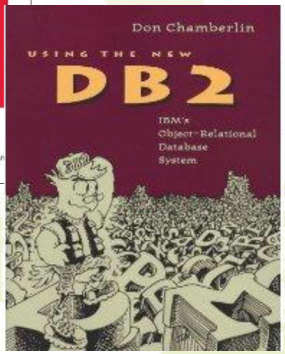


0. Recommended Literature

- **Database Modeling and Design: Logical Design**
  – Teorey, Lightstone, and Nadeau
  – Morgan Kaufmann
- **SQL Cookbook**
  – Molinaro
  – O'Reilly
- **Using the New DB2**
  – Chamberlin
  – AP Professional





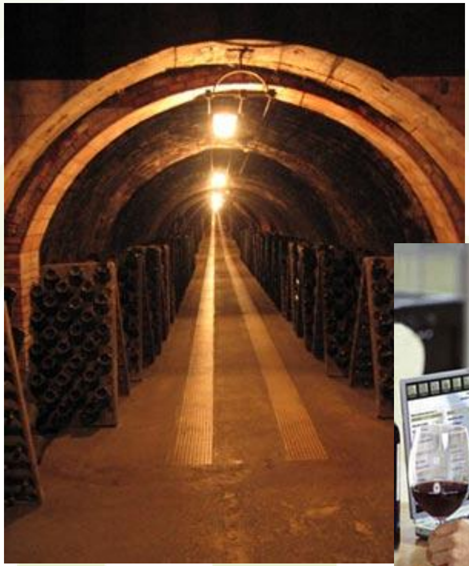


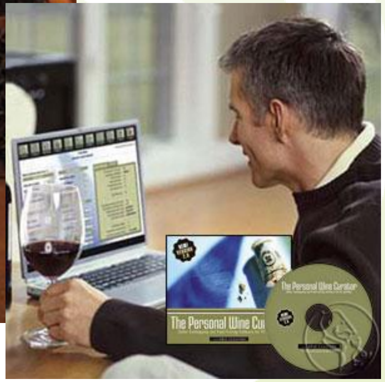
1.1 What's a Database?

- **Managing** large amounts of **data** is an integral part of most nowadays business and governmental activities
  - Collecting taxes
  - Bank account management
  - Bookkeeping
  - Airline reservations
  - Human resource management
  - …

1.1 What's a Database?

- **Databases** are needed to manage that **vast amount of data**
- A database (**DB**) is a collection of **related data**
  - Represents some aspects of the **real world**
    - **Universe of discourse**
  - Data is logically **coherent**
  - Is provided for an intended group of **users** and **applications**





1.1 What's a Database?

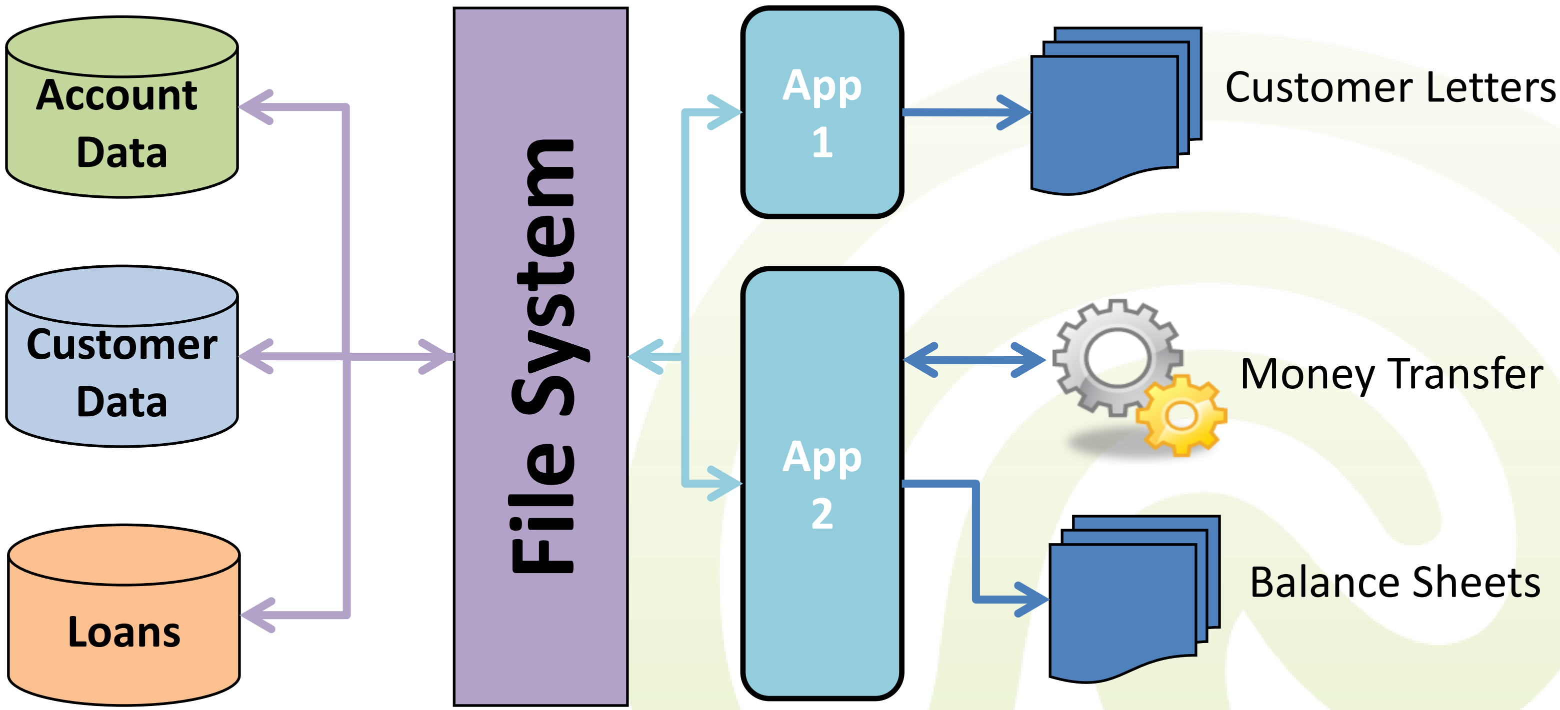
- As for today, the **database industry** is one of the most successful branches of computer science
  - Constantly growing since the 1960s
  - More than **$8 billion revenue** per year
  - DB systems found in nearly any application
  - Ranging from large commercial transaction-processing systems to small open-source systems for your Web site

1.1 What's a Database?

- A database management system (**DBMS**) is a collection of programs to maintain a database, that is, for
  - Definition of data and structure
  - Physical construction
  - Manipulation
  - Sharing/Protecting
  - Persistence/Recovery

1.1 File Systems

- A file system is not a database!
- File management systems are **physical** interfaces



1.1 File Systems

- **Advantages**
  - Fast and easy access
- **Disadvantages**
  - Uncontrolled redundancy
  - manual maintenance of consistency
  - Limited data sharing and access rights
  - Poor enforcement of standards
  - Excessive data and access paths maintenance

1.1 Databases

- Databases are **logical** interfaces
  - Retrieval of data using **data semantics**
  - Controlled redundancy
  - Data consistency & integrity constraints
  - Effective and secure data sharing
  - Backup and recovery
- However…
  - More complex
  - More expensive data access

1.1 Databases

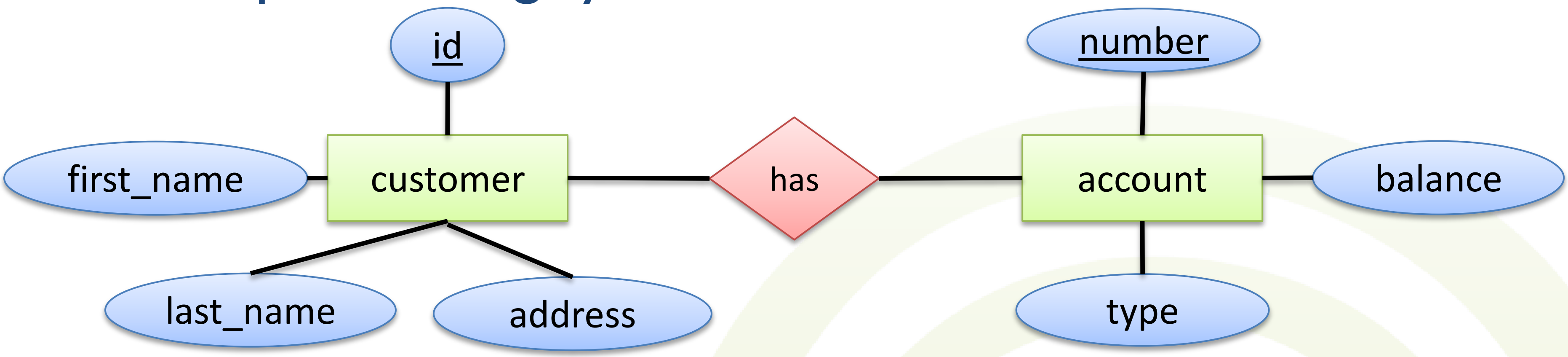
- **DBMS** replaced previously dominant file-based systems in **banking** due to special requirements
  - **Simultaneous** and quick access is necessary
  - Failures and loss of data **cannot** be tolerated
  - Data always has to remain in a **consistent** state
  - Frequent queries and modifications

1.1 Characteristics of DBs

- Databases **control redundancy**
  - Same data used by different applications or tasks is **stored only once**
  - Access via a **single interface** provided by DBMS
  - Redundancy only purposefully used to speed up data access (e.g. materialized views)
- Problems of **uncontrolled redundancy**
  - Difficulties in consistently updating data
  - Waste of storage space

1.1 Characteristics of DBs

- Databases are **well-structured,** e.g. ER model:
  - Simple banking system



- Relational Databases provide
  - **Catalog** (data dictionary) contains all **meta data**
  - Defines the **structure** of the data in the database

EN 1.3

1.1 Characteristics of DBs

- Databases support **declarative querying**
  - Just specify what you want, not how and from where to get it
  - Queries are separated and abstracted from the actual physical organization and storage of data
- Get the first name of all customers with last name "Smith"
  - File system: Trouble with physical organization of data
    - Load file "c:\datasets\customerData.csv"
    - Build a regular expression and iterate over lines:
      If 2nd word in line equals "Smith," then return 3rd word
    - Stop when end-of-file marker is reached
  - Database system: simply query
    - SELECT first_name FROM data WHERE last_name='Smith'

1.1 Characteristics of DBs

- Databases aim at **efficient** manipulation of data
  - Physical tuning allows for good data allocation
  - Indexes speed up search and access
  - Query plans are optimized to improve performance
- Example: Simple Index

Index File
(checking accounts)

number
4543032
7809849
8942214

Data File

number	type	balance
1278945	saving	€ 312.10
2437954	saving	€ 1324.82
4543032	checking	€ -43.03
5539783	saving	€ 12.54
7809849	checking	€ 7643.89
8942214	checking	€ -345.17
9134354	saving	€ 2.22
9543252	saving	€ 524.89

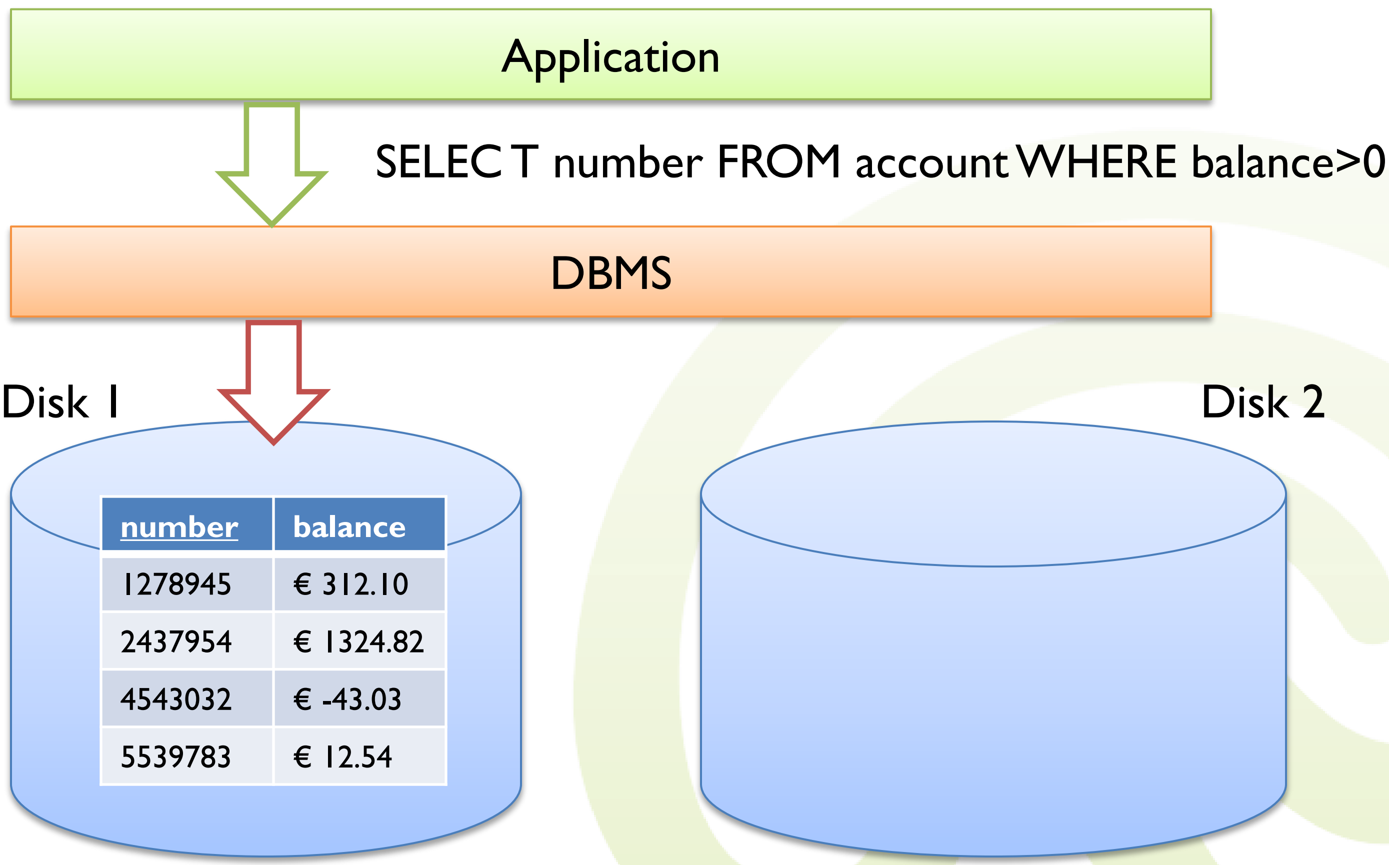
1.1 Characteristics of DBs

- **Isolation** between applications and data
  - Database employs **data abstraction** by providing **data models**
  - Applications work only on the **conceptual representation** of data
    - Data is strictly **typed** (Integer, Timestamp, Varchar, …)
    - Details on where data is actually stored and how it is accessed are **hidden** by the DBMS
    - Applications can access and manipulate data by invoking **abstract operations** (e.g. SQL select statements)
  - DBMS-controlled parts of the file system are **protected** against external manipulations (tablespaces)

EN 1.3

1.1 Characteristics of DBs

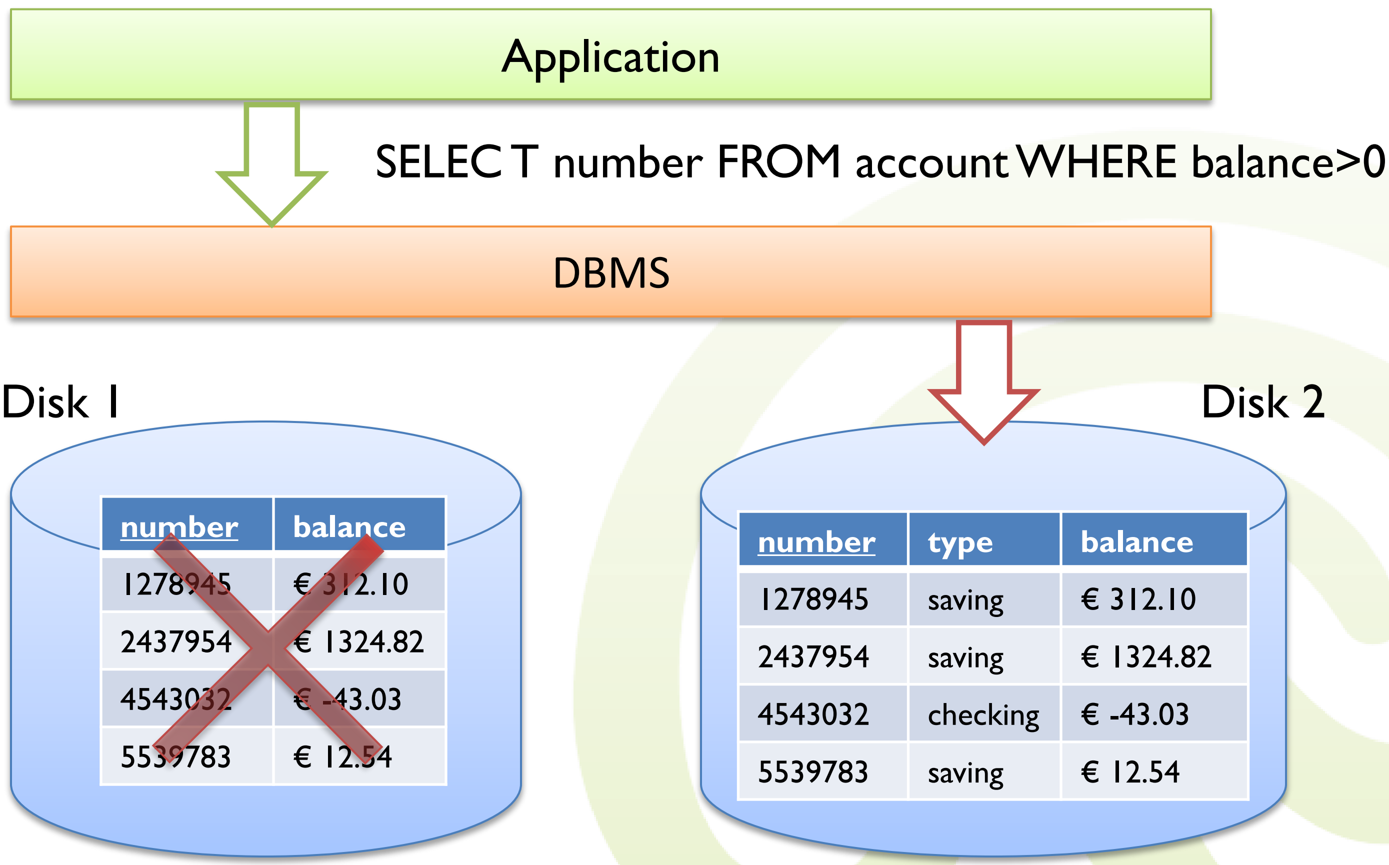
- **Example:** Schema can be changed and tablespace moved without being noticed by app



number	balance
1278945	€ 312.10
2437954	€ 1324.82
4543032	€ -43.03
5539783	€ 12.54

1.1 Characteristics of DBs

- **Example:** Schema can be changed and tablespace moved without being noticed by app



Disk 1:

number	balance
1278945	€ 312.10
2437954	€ 1324.82
4543032	€ -43.03
5539783	€ 12.54

Disk 2:

number	type	balance
1278945	saving	€ 312.10
2437954	saving	€ 1324.82
4543032	checking	€ -43.03
5539783	saving	€ 12.54

1.1 Characteristics of DBs

- Supports multiple **views** of the data
  - Views provide a different perspective of the DB
    - A user's conceptual understanding or task-based excerpt of the data (e.g. aggregations)
    - Security considerations and access control (e.g. projections)
  - For applications, a view does not differ from a table
  - Views may contain **subsets** of a DB and/or contain **virtual data**
    - Virtual data is **derived** from the DB (mostly by simple SQL statements, e.g. joins over several tables)
    - Can either be computed at query time or **materialized** upfront

EN 1.3

1.1 Characteristics of DBs

- Example views: **Projection**
  - Saving account clerk vs. checking account clerk

Original Table

number	type	balance
1278945	saving	€ 312.10
2437954	saving	€ 1324.82
4543032	checking	€ -43.03
5539783	saving	€ 12.54
7809849	checking	€ 7643.89
8942214	checking	€ -345.17
9134354	saving	€ 2.22
9543252	saving	€ 524.89

Saving View

number	balance
1278945	€ 312.10
2437954	€ 1324.82
5539783	€ 12.54
9134354	€ 2.22
9543252	€ 524.89

Checking View

number	balance
4543032	€ -43.03
7809849	€ 7643.89
8942214	€ -345.17

1.1 Characteristics of DBs

- **Sharing** of data and support for **atomic multi-user transactions**
  - Transactions are a **series of database operations** executed as **one logical operation**
  - **Concurrency control** is necessary for maintaining consistency
    - Multiple users and applications may access the DB at the same time
  - **Transactions** need to be **atomic** and **isolated** from each other

EN 1.3

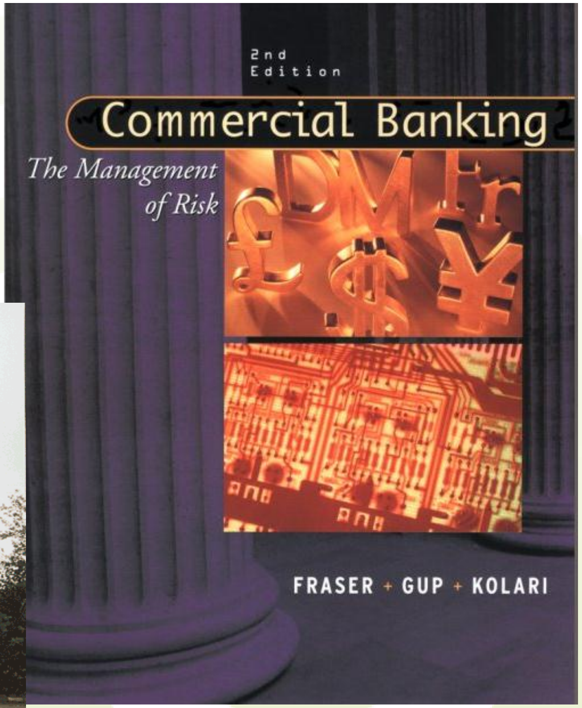
1.1 Characteristics of DBs

- Example: Atomic transactions
  - **Program:**
    Transfer *x* Euros from Account 1 to Account 2
    1. Debit amount *x* from Account 1
    2. Credit amount *x* to Account 2

1.1 Characteristics of DBs

- Example: Atomic transactions
  - **Program:**
    Transfer *x* Euros from Account 1 to Account 2
    1. Debit amount *x* from Account 1
    2. Credit amount *x* to Account 2
  - **But what happens if the system fails after performing the first step?**

- **Persistence** of data and disaster **recovery**
  - Data needs to be persistent and accessible at all times
  - **Quick** recovery from system crashes **without data loss**
  - Recovery from natural disasters (fire, earthquake, …)



EN 1.3

1.1 Database Users

- Usually **several groups of persons** are involved in the daily usage of a large DBMS (many job opportunities for smart DB people…)
- Persons directly involved on DB level
  - **Database administrators**
    - Responsible for tuning and maintaining the DBMS
    - Management of storage space, security, hardware, software, etc.
  - **Database designers**
    - Identifies the data that needs to be stored and chooses appropriate data structures and representations
    - Integrates the needs of all users into the design

1.1 Database Users

- **Application developers**
  - Identify the requirements of the end-users
  - Develop the software that is used by (naïve) end-users to interact with the DB
  - Cooperate closely with DB designers

- Persons working behind the scenes
  - **DBMS designers and implementers**
    - Implement the DBMS software
  - **Tool developers**
    - Develop generic tools that extend the DBMS' functionalities
  - **Operators and maintenance personnel**
    - Responsible for actually running and maintaining the DBMS hardware

1.1 Database Users

- **End users**
  - All people who use the DB to do their job
- End users split into
  - **Naïve end users**
    - Make up most DB users
    - Usually **repeat** similar tasks over and over
    - Are supported by predesigned interfaces for their tasks
    - Examples: bank tellers, reservation clerks, …

1.1 Database Users

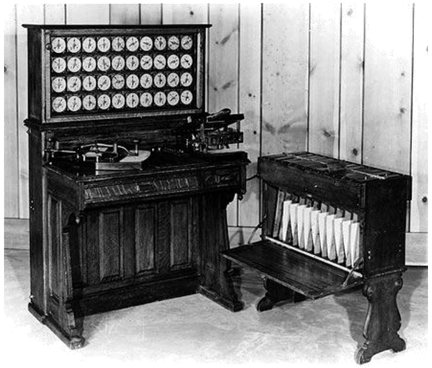
- **Sophisticated end users**
  - Require **complex** non-standard operations and views from the DB
  - Are familiar with the facilities of the DBMS
  - Can solve their problems themselves, but require complex tools
  - Examples: engineers, scientists, business analysts, …

- **Casual end users**
  - Use DB only from time to time, but need to perform different tasks
  - Are familiar with query languages
  - Examples: People in middle or senior management

1.3 History of DBs

Detour

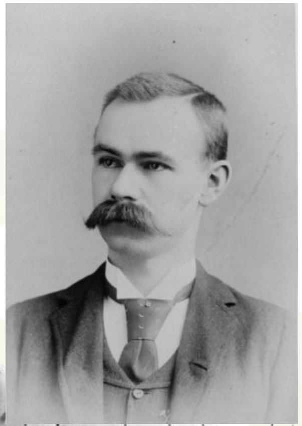
- Databases have an exceptional history of development
  - Many synergies between **academic, governmental** and **industrial** research
  - Much to be learned from it
  - Most popular concepts used today have been invented decades ago

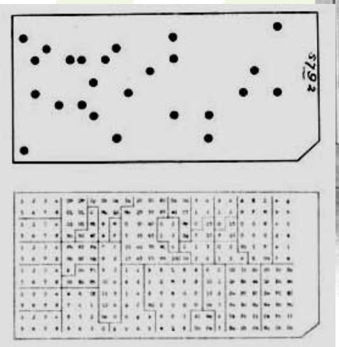


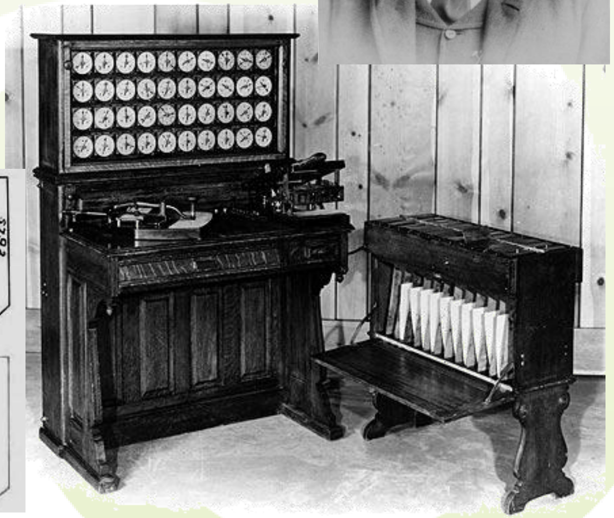
1.3 History of DBs

Detour

- The beginnings
  - 1880: U.S. Bureau of Census instructs **Herman Hollerith** to develop a machine for storing census data
  - Result: **Punch card** tabulator machine
    - The evaluation of 1880's census took 8 years
    - 1890's has been finished after only one year
  - Leads to the foundation of **IBM**
    - International Business Machines
  - Data processing machines soon established in accounting



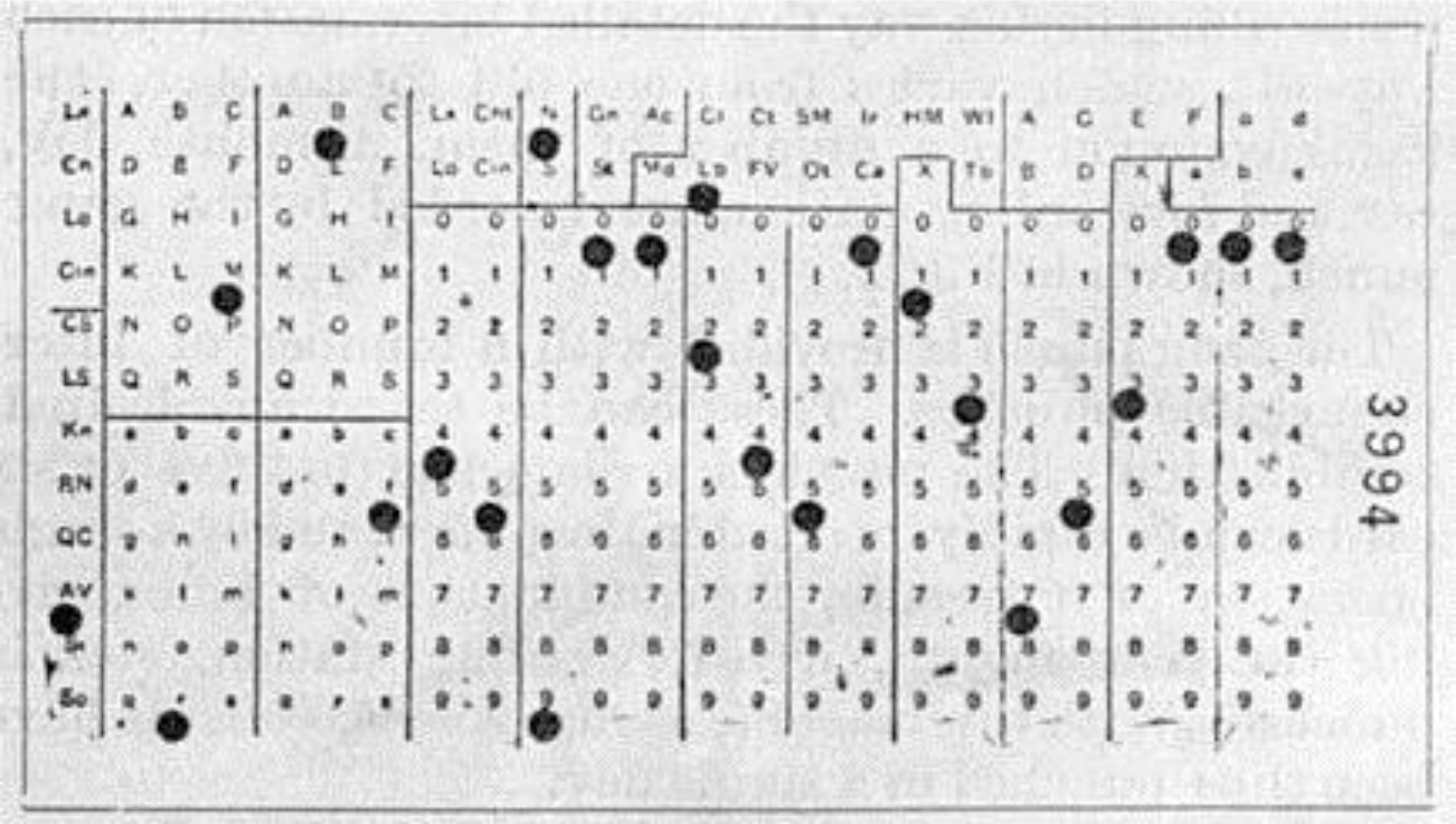




1.3 History of DBs

Detour

- One of Hollerith's punch cards:



1.3 History of DBs

Detour

- Tabulator machines
  - Operations or "programs" directed by a plug board
  - Up to 150 cards per minute
  - Results were printed or punched for input to other processing steps

1.3 History of DBs

Detour

- In **1951** IBM develops the electric **UNIVAC I**
  - first commercial computer produced in the U.S.
    - Programmable (turing complete)
    - Input (programs and data) with tape or punched cards

- In **1959,** U.S. dominated the (still highly active) punch card machine market
  - Within the U.S., the Pentagon alone used more than 200 data processing computers, costing $70 million per year

1.3 History of DBs

Detour

- In **1964,** the term "data base" appeared for the first time in military computing using **time sharing systems**
  - Data could be shared among users
  - But data was still bound to one specific application
    - Similar data needed by multiple applications had to be duplicated
    - Consistency problems when updating data
  - Data structure highly-dependent on the hardware and (low-level) programming language used
    - Inspired by punch cards and optimized for magnetic tapes
    - Usually, no **relationships** between different records have been stored, just plain data

1.3 History of DBs

Detour

- To turn stored data into a proper **database**, the following goals had to be achieved (McGee, 1981):
  - **Data consolidation**
    - Data must be stored in a central place, accessible to all applications
    - Knowledge about relationships between records must be represented
  - **Data independence**
    - Data must be independent of the specific quirks of the particular low level programming language used
    - Provide high-level interfaces to physical data storage
  - **Data protection**
    - Data must be protected against loss and abuse

1.3 History of DBs

Detour

- **Data consolidation** motivated the development of data models
  - Hierarchical data model
  - Network data model
  - **Relational data model**
  - Object-oriented data model
  - Semantic data model
- **Data independence** inspired the development of query models and high-level languages
  - **Relational Algebra, SQL**
- **Data protection** led to development of transactions, backup schemes, and security protocols

1.3 History of DBs

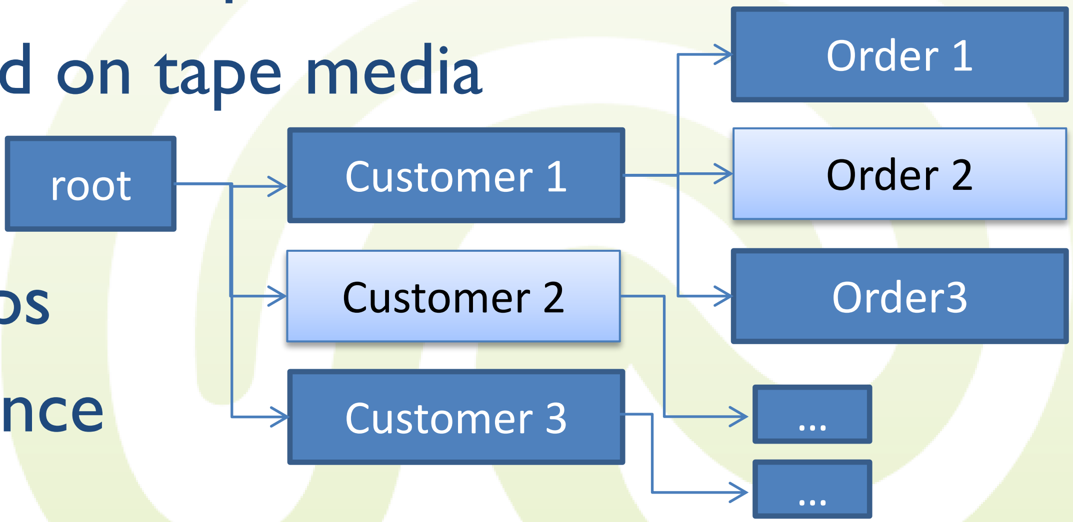
Detour

- **Hierarchical data model**
  - First appearance in **IBM's IMS** database system, designed for the Apollo Program in **1966**
    - Still, as of 2006, 95% of all Fortune 1000 companies use IBM IMS in their data backbone…
  - Benefits from **advances** in **hardware** design
    - Random access main memory and tape media available

1.3 History of DBs

Detour

- **Hierarchical data model**
  - Each type of record has some defined structured data
  - Hierarchical **one-to-many** relationships
- **Advantages**
  - 1:n relationships can be expressed
  - Can easily be stored on tape media
- **Disadvantages**
  - No n:m relationships
  - No data independence



1.3 History of DBs

Detour

- **Network data model**
  - In the mid-1960th, direct access storage devices (DASD) gained momentum
    - Primarily hard disks
    - More complex storage schemes possible
  - Hierarchical model failed,
    e.g. for bill-of-material-processing (BOMP)
    - Many-to-many relationships needed
    - Development of the IBM DBOMP system (1960)
  - Result: Network model
    - Two types of files: Master files, chain files
    - Chain file entries could chain master file entry to one another

1.3 History of DBs

Detour

- **Network data model**
  - The model was standardized by Charles W. Bachman for the **CODASYL** Consortium in 1969
    - CODASYL = Conference of Data Systems Languages
    - Thus, also called the CODASYL model
  - Allowed for more natural modeling of **associations**
  - Advantage
    - **Many-to-many-relationships**
  - Disadvantages
    - No declarative queries
    - Queries must state the data access path

1.3 History of DBs

Detour

- **The relational data model**
- Published by **Edgar F. "Ted" Codd** in 1970, after several years of work
  - *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, 1970
  - Employee of IBM Research
    - IBM **ignored** his idea for a long time as not being "practical" while pushing it's hierarchical IMS database system
    - Other researchers in the field also **rejected** his theories
    - Finally, he received the Turing Award in 1981

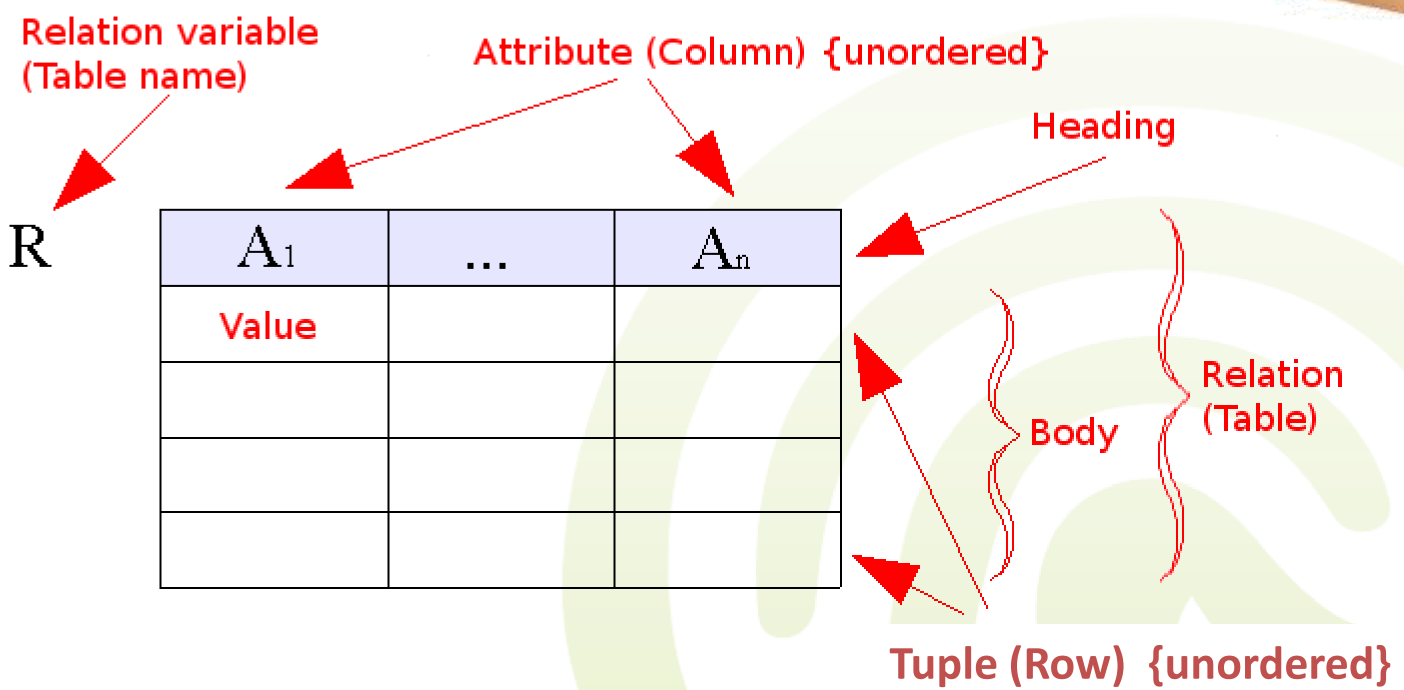
1.3 History of DBs

Detour

- **Idea underlying the relational model:**
  - Database is seen as a collection of **predicates** over a finite set of **predicate variables**
    - Example:
      - is_supervisor_of(*x*, *y*)
      - is_supervisor_of('W.-T. Balke', 'S. Barthel') (TRUE)
      - is_supervisor_of('P. Wille', 'S. Barthel') (FALSE)
    - The set of all true assignments is called a **relation**
    - Relations are stored in **tables**
  - Contents of the DB = a collection of **relations**
  - Queries are also **predicates**
    - Queries and data are very similar
    - Allows for **declarative querying**

1.3 History of DBs

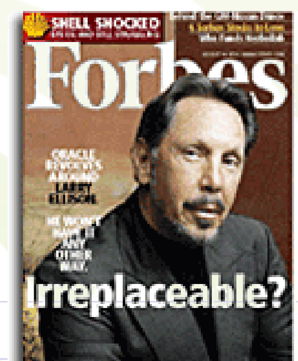
- It's really like a collection of index cards
  - More details during the next weeks…

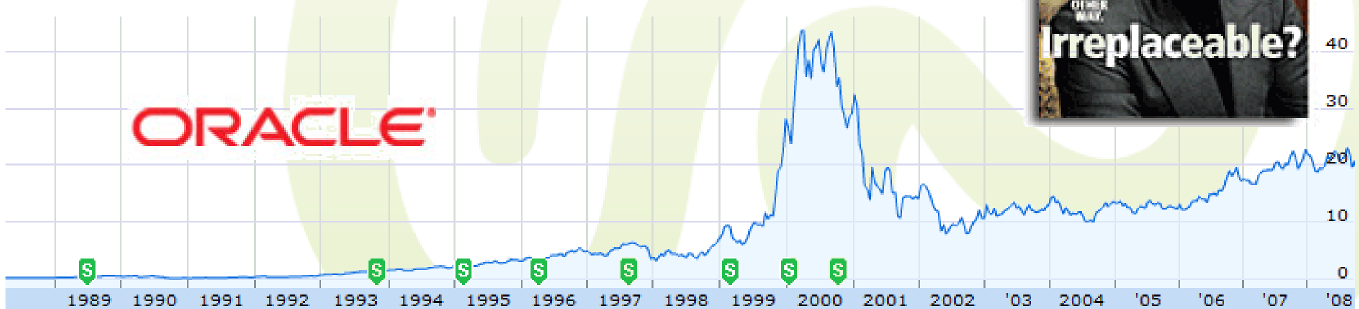


1.3 History of DBs

Detour

- Beginning 1977, **Lawrence J. Ellison** picked up the idea and created **Oracle DB** (currently in version 11g)
  - And became insanely rich – long time in the Top 10 of the richest people
  - In 2007 Oracle ranked third on the list of largest software companies in the world, after Microsoft and IBM





1.3 History of DBs

Detour

- Oracle also sells a suite of **business applications**
  – Oracle eBusiness Suite
  – Includes software to perform financial- and manufacturing-related operations, customer relationship management, enterprise resource planning, and human resource management
- Basically gained from high-value acquisitions
  – JD Edwards, PeopleSoft, Siebel Systems, BEA, …

1.3 History of DBs

Detour

- During the 1970s, IBM had also decided to develop a relational database system
  - **System R** with the first implementation of the **SQL** declarative query language (SEQUEL)
  - At first, mostly a research prototype, later became the base for **IBM DB2**

- Today, the relational model is the **de-facto standard** of most modern databases

1.3 History of DBs

Detour

Year	Event
1880	Hollerith census machine
1951	Univac I electrical data machine
1959	First CODASYL Conference
1960	Flight reservation system SABRE
1966	IMS hierarchical database
1969	Network model
1971	CODASYL Recommendation for DDL and 3-Layer-Architecture
1975	System R introduces SEQUEL query language
1976	System R introduces transaction concepts
1976	Peter Chen proposes entity relationship modeling
1980	Oracle, Informix and others start selling DBMS with SQL support

1.3 History of DBs

Detour

Year	Event
1983	Work on ACID transactions published by Theo Haerder and Andreas Reuter
1986	SQL standardized as SQL-1 ANSI/SQL
1987	SQL internationally standardized as ISO 9075
1989	SQL 2 standard supports referential integrity
1991	SQL 2 supports domains and key definitions
1993	Object-oriented data model
1995	Preliminary SQL 3 supporting sub-tables, recursion, procedures, and triggers
1996	First object-oriented databases
1999	First part of the SQL 3 standard finalized
2003	SQL 2003 finalized with support for object-relational extensions
…	**To be continued …**

1.3 History of DBs

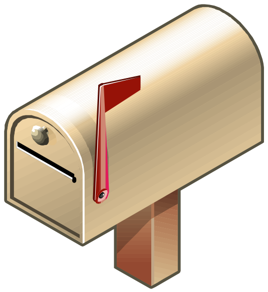
Detour

- **Beyond the relational model…**
  - Data models based on formal logic
    - Deductive databases and expert systems
  - Object-oriented data models
    - Main Idea: Object-oriented design (garage metaphor)
    - Very easy integration in OO programming languages
    - Today, mostly integrated into the relational model
  - Semi-structured data models
    - Most important: XML
    - Allows a large degree of structural freedom
  - For details, take the master's courses on these topics …

Homework

- **Weekly** homework assignments
  - **50%** of the maximum homework score is required to receive the "Studienleistung"
  - can be **downloaded** from our website:
    - http://www.ifis.cs.tu-bs.de
- Homework should be completed within groups of two students **(no larger groups, please!)**
- To be handed in **before the next lecture**
  - Drop your homework into the mailbox at our institute (Informatikzentrum, 2nd floor)
  - Or just give it to us right before the next lecture
  - No email submissions!

Homework



- Mark each sheet of paper with
  - your **names** and **matriculation numbers**
  - your **tutorial group number**
- If you have multiple pages, **staple** them together
- Of course, you can **discuss** the homework assignment with other people, but **do not copy it**
- Homework is graded and corrected/commented by our student assistants and returned to you in your tutorial group
  - For any questions regarding the grading, contact the respective student assistant directly

Tutorial groups

- The tutorial groups start in two weeks
  - But: **Registration is required!**
  - You can sign up in the lists at our whiteboard at the institute
  - Registration possible until October 25 (next Thursday)

Tutorial group: Registration

Registration starts directly after this lecture and will look like this:

Gruppe 1

Dienstags 13:15 – 14:45 Uhr in Raum IZ 160

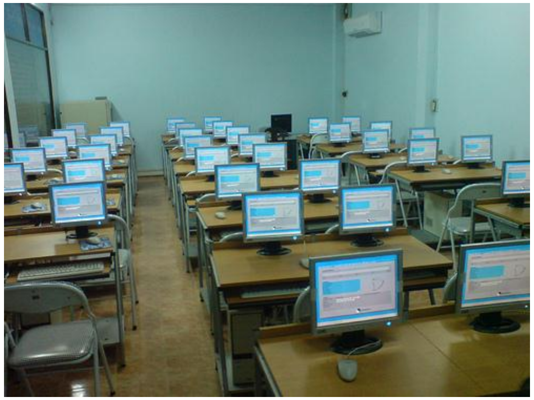
Gruppen-Nummer	Matrikel-Nummer	Email	Nachname	Vorname	RDB 1	SQL
1	12345	ben@ifis.de	Köhncke	Benjamin	x	x
	67891	tilo@ifis.de	Balke	Wolf-Tilo	x	x
2						
3						
4						
5						

Group Number and Time Slot

One Group

Which Courses?

SQL Lab



- In addition to this course, we offer a practical lab course
  - **"SQL Lab"**
  - Students in "Bachelor Informatik" and "Bachelor Wirtschaftinformatik" are recommended to participate
    - Awards 4 credits
    - Others may also voluntarily participate, but it is up to their course of study to accept the credits or not

SQL Lab

- Lab course extends the written home works with additional **computer-based tasks**
  - Extended **data modeling** using model tools
    - ER models
    - UML models
  - **Integrating** data models
  - **Creating / Modifying / Querying** databases
    - Developing of easy up to complex SQL queries
    - Modifying data with SQL
  - Connection between databases and applications
    - Using JDBC in Java to interact with databases

SQL Lab

- The lab course starts in three weeks
  - But: **Registration is required!**
  - You can check for the lab when you sign up for your tutorial group
  - Registration possible **until October 25** (next Thursday)
- Larger assignments have to be completed (every two or three weeks) and will be graded
- **Fixed(!) pairs of two** students each
  - You may choose a preferred partner
  - If you don't, you get a random partner

SQL Lab: Grading

- Each assignment will be **graded** as follows:
  - "Good" solution: +1
  - "OK": 0
  - "Bad" or insufficient: –1

- To pass the lab, the sum of all grades must be **positive** at the end of the semester and each assignment must be delivered

Summary

- Databases
  - are **logical interfaces**
  - **control redundancy**
  - are **well-structured**
  - support **declarative querying**
  - aim at **efficient manipulation** of data
  - support **multiple views** of the data
  - support **atomic multi-user transactions**
  - support **persistence** and **recovery** of data