ifis
Institut für Informationssysteme
Technische Universität Braunschweig

Relational Database Systems 1

Wolf-Tilo Balke
Niklas Kiehne, Denis Nagel,
Enrique Pinto Dominguez
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de

0. People involved

- Who is who ?
  - Wolf-Tilo Balke
    - lecture, exams
  - Niklas Kiehne
    - lecture, detours
  - Denis Nagel, Enrique Pinto Dominguez
    - homework / tutorials
  - Regine Dalkiran
    - office

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

0. Organizational Issues

- Lecture
  - October 23, 2025 to February 5, 2026
  - 15:00 – 17:15 (including a break)
  - *integrated* lecture (theory and detours)
  - 5 credits
  - Written exam on **13.03.2026**


- Homework
  - weekly assignments
    - … can be downloaded from StudIP
    - … must (!) be completed in groups of **three** students

0. Organizational Issues

- **Tutorial groups**
  - led by our HiWis
  - homework discussion
- In order to pass this **module** (5CP) you need to
  1) … achieve **50%** of homework points (Studienleistung, ungraded)
  2) … pass the exam (Prüfungsleistung, graded)

0. Homework and Tutorials

- Homework assignments (each Thursday)
  - **downloadable** from **StudIP after each lecture**
  - Tutorial groups will start **03.11**
- Homework is graded by our HiWis
  - **Send it by e-mail** to your HiWi by the following week's Thursday **no later than 15:00**
    - HiWi mail addresses are in StudIP
- Tutorial groups meet in room IZ 251

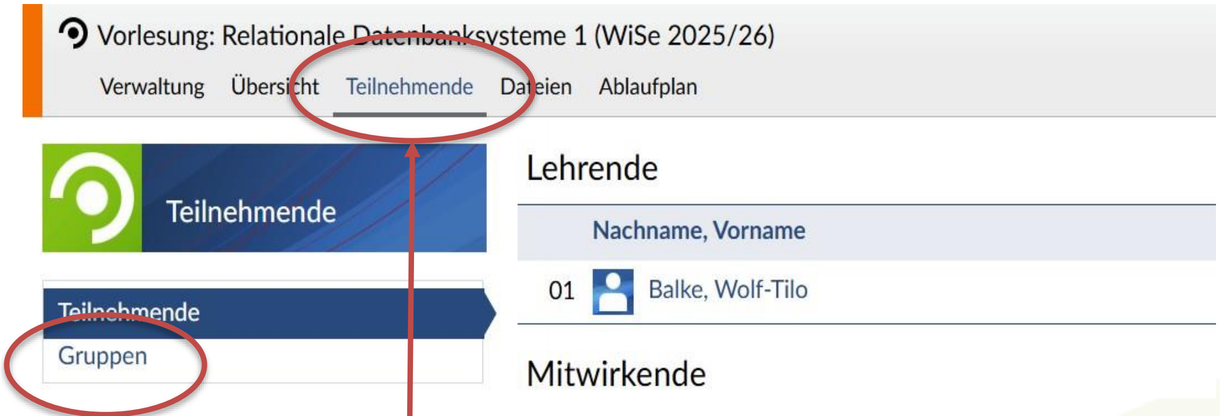
0. Homework and Tutorials

- **Groups of <u>3 students!</u>**
  - Homework to be sent in **before the next lecture**
    - **3 pm sharp!** (~1 week of time)
  - Please send us **PDF files only**!
  - Mark each page with
    - **both** your **names** and
    - **matriculation numbers**
  - Adhere to our file naming guidelines
- Any **deviations** will lead to **0pts** for the sheet

0. Homework and Tutorials

- **So, who is my HiWi?**
  - We will open **Stud.IP groups** where you can assign yourself to a tutorial group, iff you do not yet have the Studienleistung
    - please make sure that **you and your partners** are **assigned to the same group** in Stud.IP!
    - groups are open starting **23.10 18:00 till 30.10 18:00**
  - The HiWi will:
    - send you the corrected homework before his tutorial starts
    - discuss the respective homework in his tutorial session

0. Tutorial Groups

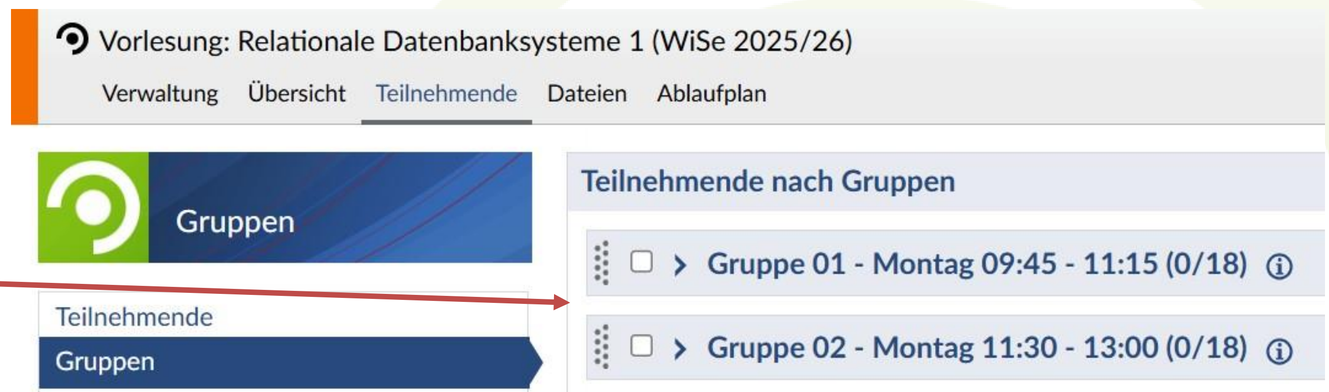


Open: 23.10 18:00
till 30.10 18:00

1.

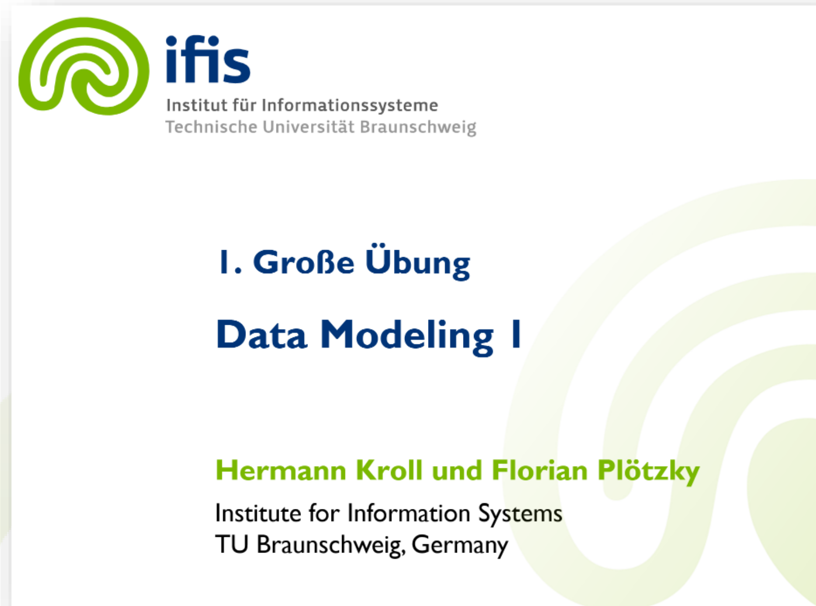
2.

3.



0. Classroom Exercise

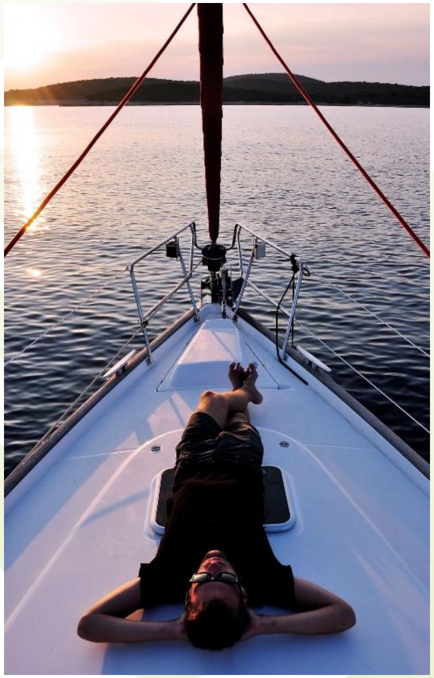
- **Each Thursday** we will upload a classroom exercise consisting of:
  - **an exercise sheet**
  - **a solution**
  - **a recording to explain the solution**



- Practice for **yourself**! Try to solve the sheet and then watch the video. Additional service to practice for the exam and to solve the homework!

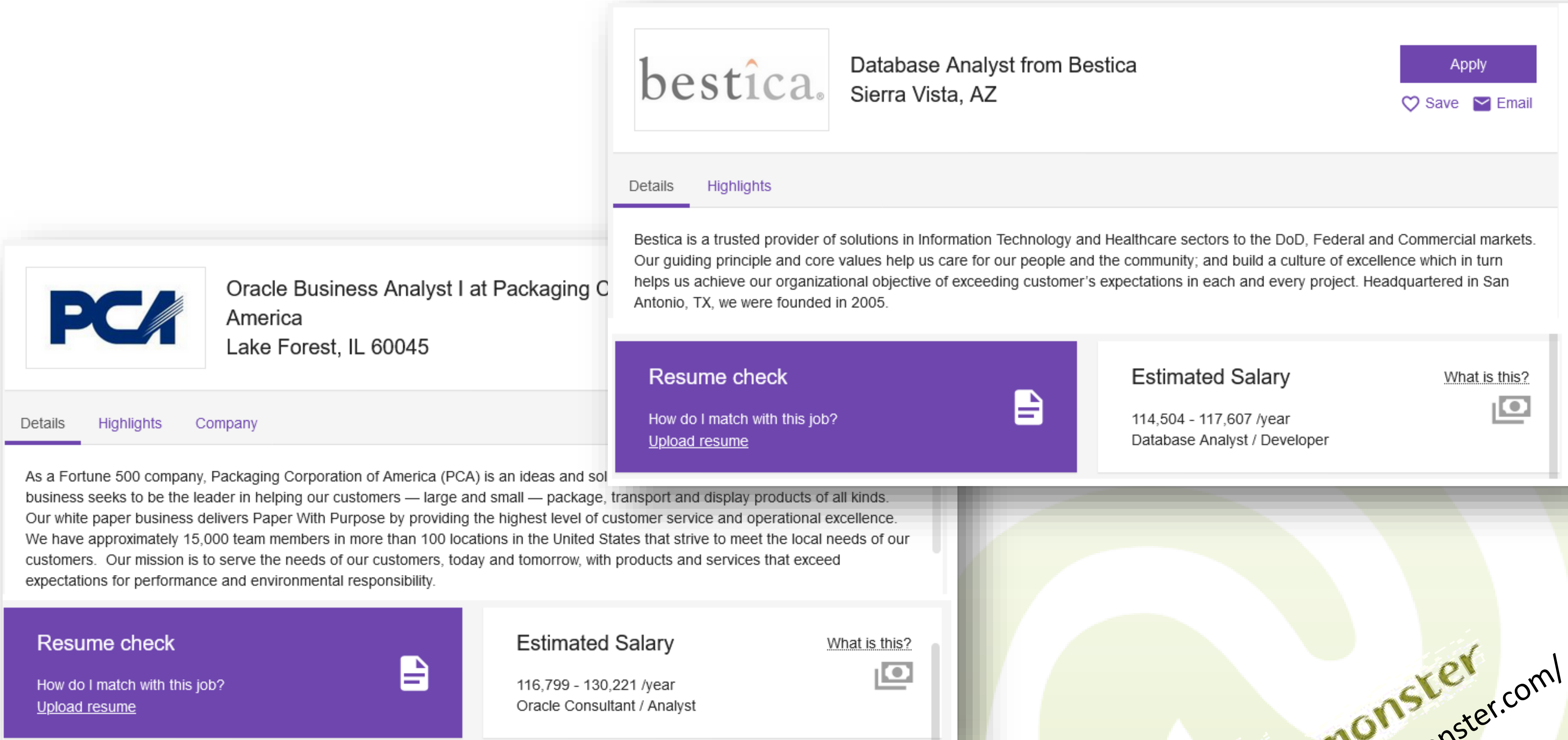
0. Why should you be here?

- Its mandatory in your course of study….
- 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?

Job descriptions also exactly describe this course…



monster

http://jobsearch.monster.com/

0. Course Objectives

- After successfully completing this course you should be able to explain the fundamental terms of…
  - **databases** in general
  - the **relational model**
  - **theoretical** and **practical** aspects of **query languages**
  - conceptual and logical **design** of databases including **normalization**
  - application programming
  - further concepts like constraints, views, indexes, transactions and object databases

0. Course Objectives

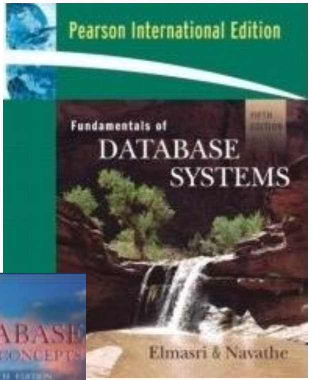
- You should furthermore be able to
  - design and implement a database for any specified domain using **ER-Diagrams** or **UML-Diagrams**, the Relational Model and SQL-DDL
  - **normalize** a given relational database schema
  - **enhance** the database with views, indexes, constraints, and triggers
  - formulate data retrieval **queries** in **SQL**, Relational **Algebra**, and Relational **Calculi**
  - write programs **accessing databases** using JDBC

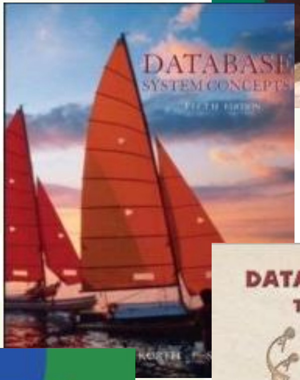
0. Contents of this Course

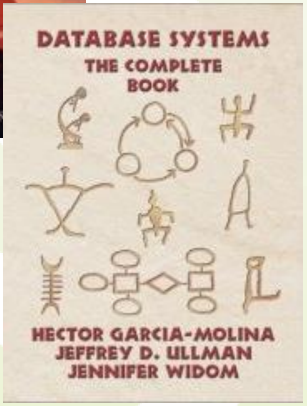
Lecture	Topic
1	Introduction
2	Data Modeling 1
3	Data Modeling 2
4	View Integration
5	Relational Model
6	Relational Algebra
7	Relational Calculus
8	SQL 1
9	SQL 2
10	Normalization
11	Application Programming 1
12	Application Programming 2
13	Object Persistence
14	NoSQL

0. Recommended Literature

- <u>**Fundamentals of Database Systems**</u> **(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

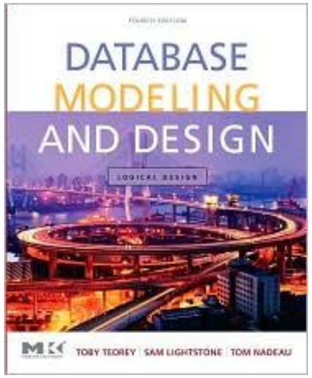




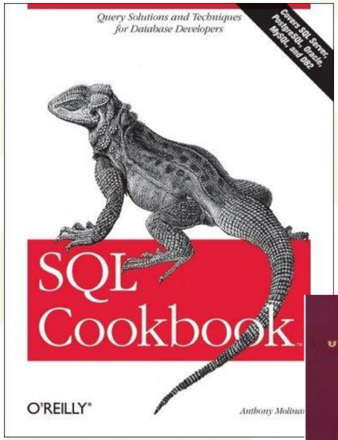


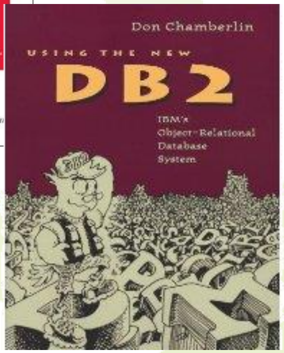
0. Recommended Literature

- **Database Modeling and Design: Logical Design**
  - Teorey, Lightstone, and Nadeau
  - Morgan Kaufmann
  - http://www.sciencedirect.com/science/book/9780123820204



- **SQL Cookbook**
  - Molinaro
  - O'Reilly
- **PostgreSQL Documentation**
  - https://www.postgresql.org/docs/
- **W3Schools SQL**
  - http://www.w3schools.com/sql/





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
  - Relational Databases II (Bachelor or Master)
    - How can we implement a DBMS?
    - Storage models, query optimization, transactions, concurrency control, recovery, data security

0. Courses at ifis

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

If you still don't have enough!

I Introduction

- **What is a Database?**
- Characteristics of a Database
- History of Databases

All images, with the exception of some screenshots, book covers and logos are from unsplash.com, wikipedia.org or wikimedia.org unless stated otherwise.

1.1 What is 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 is 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 is a Database?

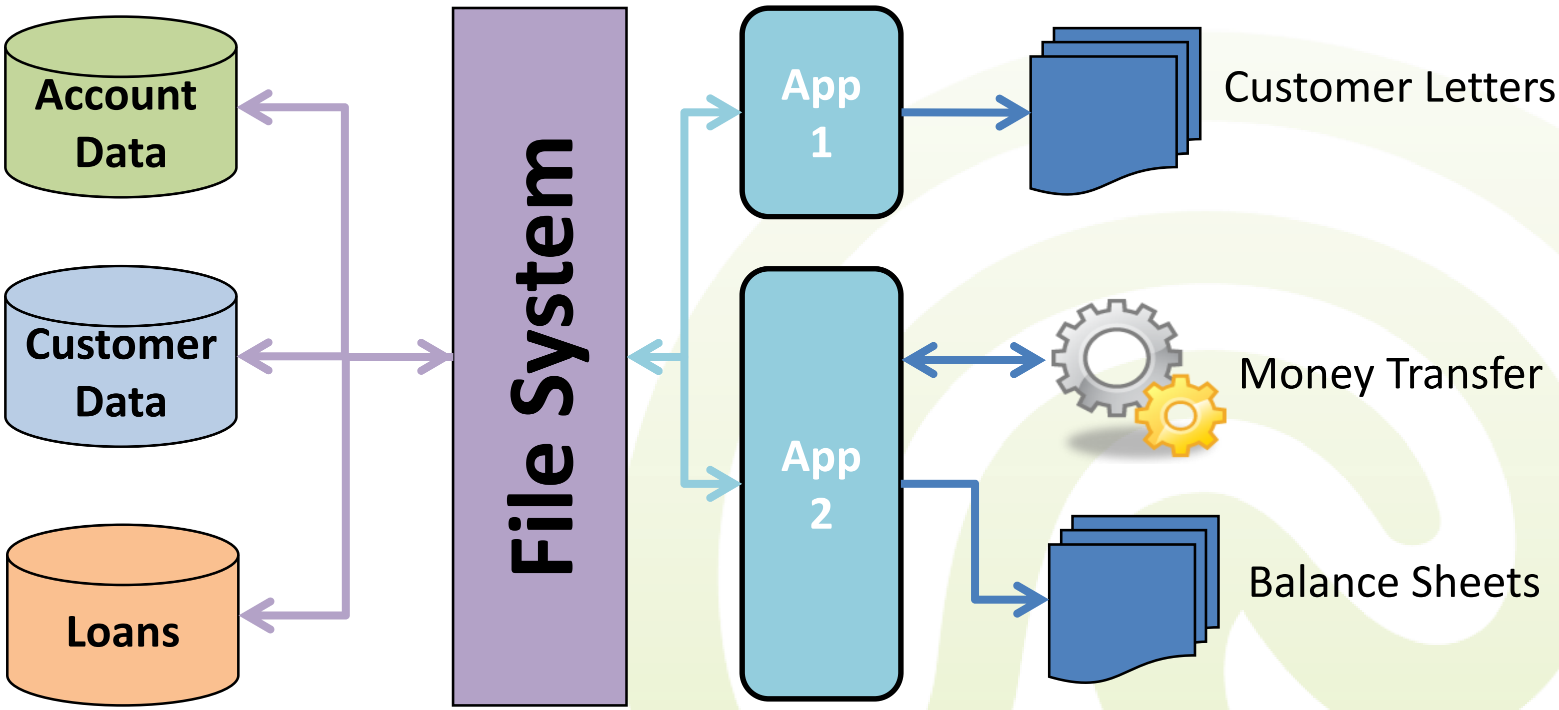
- As for today, the **database industry** is one of the most successful branches of computer science
  - constantly growing since the 1960s
  - more than **$8 billion revenue** per year
  - DB systems found in nearly any application
  - ranging from large commercial transaction-processing systems to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

1.1 What is a Database?

- Databases are maintained by using a collection of programs called a database management system (**DBMS**), that deals with
  - 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

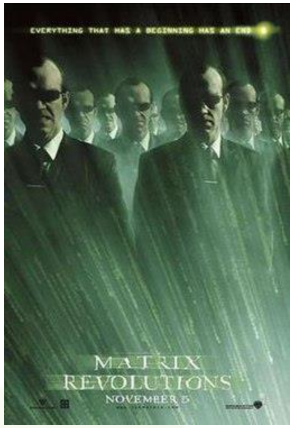
- Databases are **logical** interfaces
  – retrieval of data using **data semantics**
  – controlled redundancy
  – data consistency & integrity constraints
  – effective and secure data sharing
  – backup and recovery
- However…
  – more complex
  – more expensive data access

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

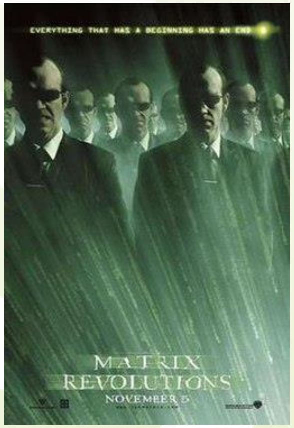
1 Introduction

- What is a Database?
- **Characteristics of a Database**
- History of Databases



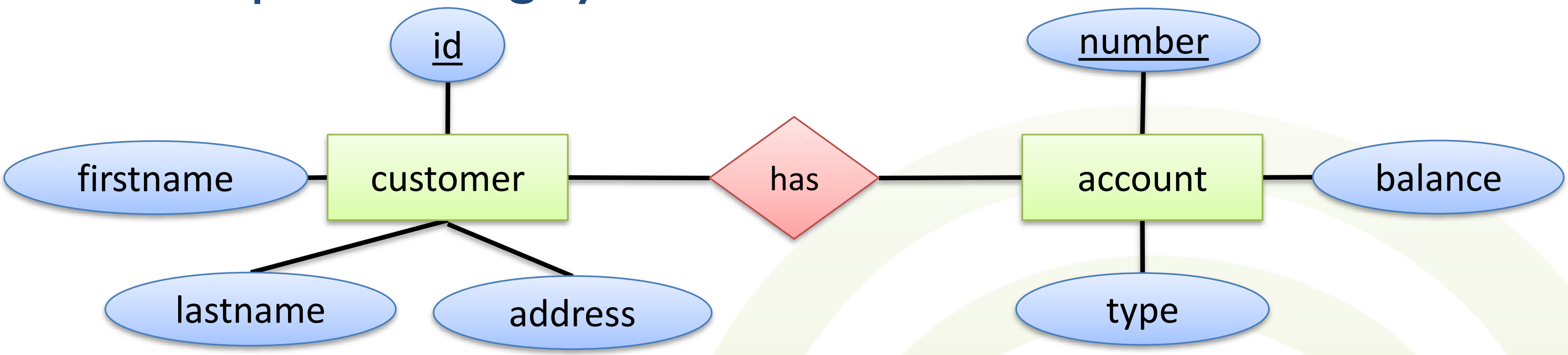
1.2 Characteristics of DBs

- 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**
  - updating data may result in inconsistent data



1.2 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

1.2 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 firstname of all customers with lastname *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 1st word.
    - Stop when end-of-file marker is reached.
  - database system: simply query
    - SELECT firstname FROM data WHERE lastname='Smith'

1.2 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

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

Index File
(checking accounts)

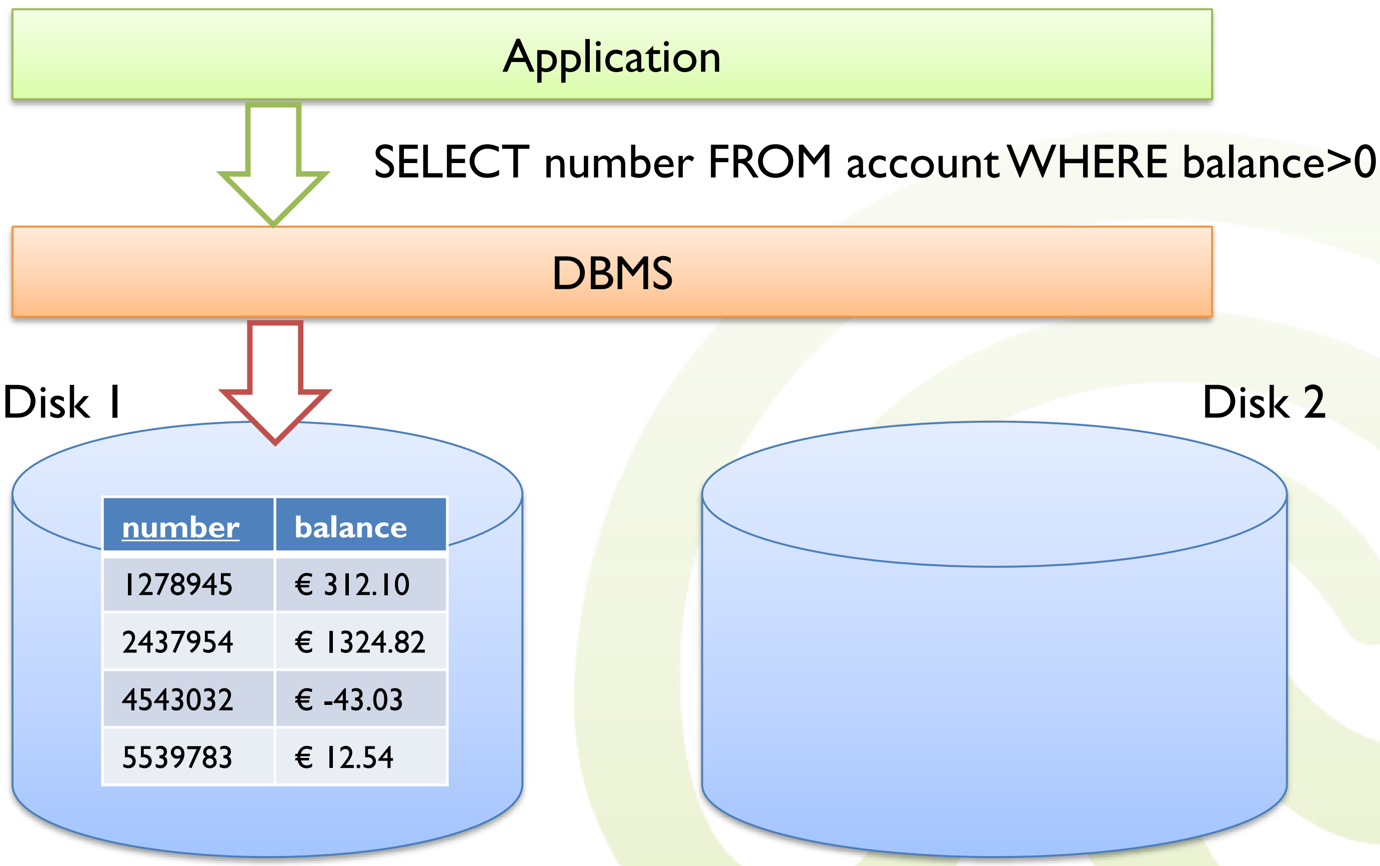
number
4543032
7809849
8942214

1.2 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, Float, 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 statements)
  - DBMS-controlled parts of the file system are **protected** against external manipulations (tablespaces)

1.2 Characteristics of DBs

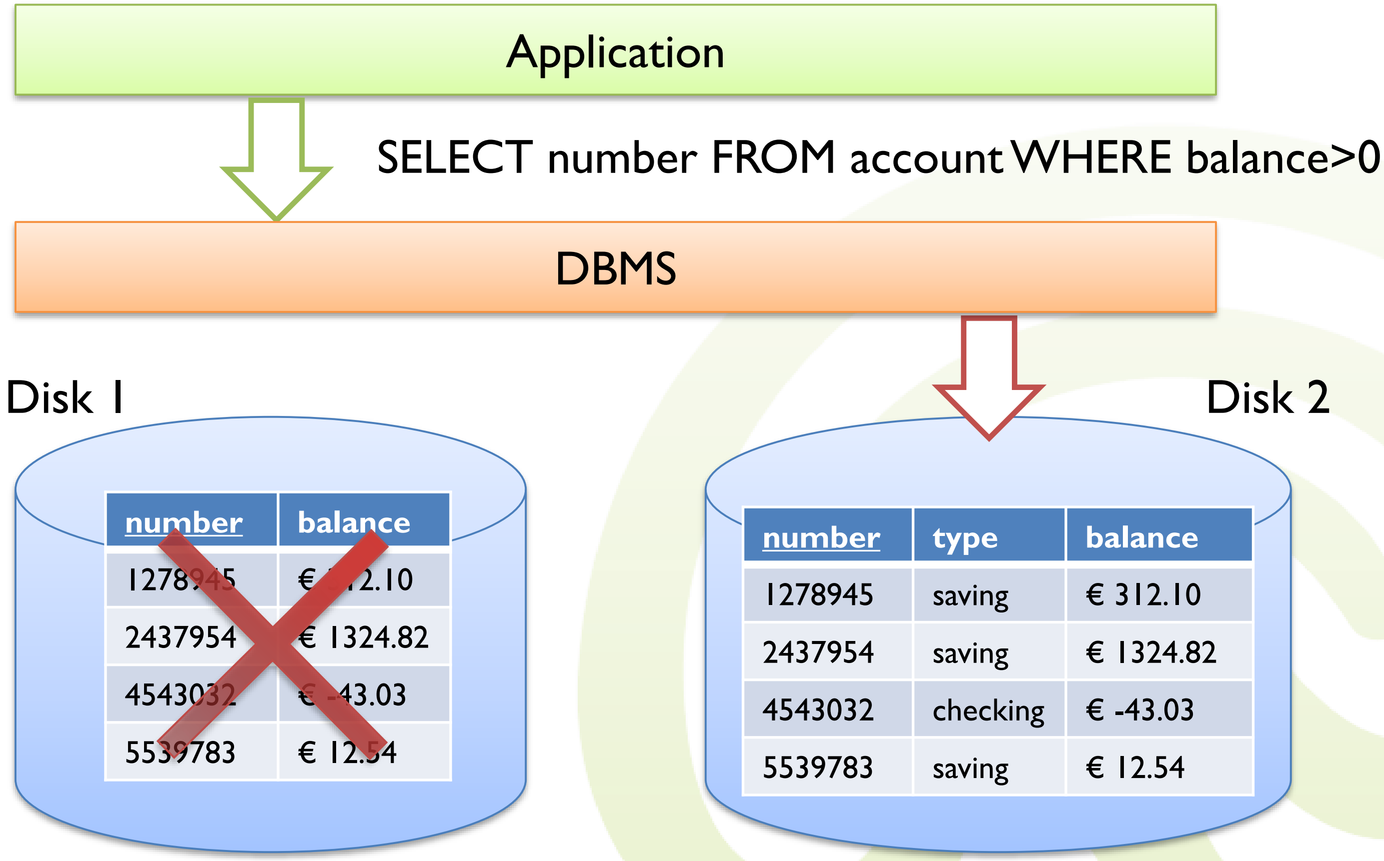
- **Example:** Schema can be changed and tablespace moved without adjusting the Application's SELECT



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

1.2 Characteristics of DBs

- **Example:** Schema can be changed and tablespace moved without adjusting the Application’s SELECT



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

1.2 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

1.2 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

1.2 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

1.2 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

1.2 Characteristics of DBs

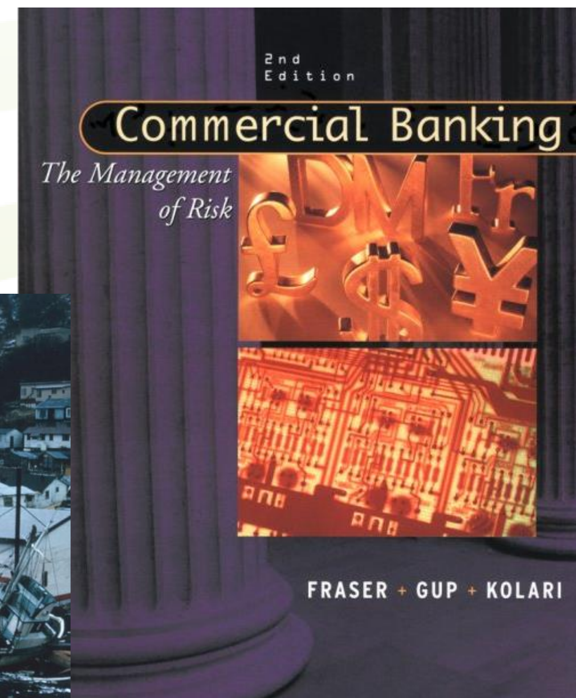
- **Example:** Atomic transactions
  - **Program:**
    Transfer *x* Euros from Account 1 to Account 2
    1. Debit amount *x* from Account 1
    2. Credit amount *x* to Account 2
  - **But what happens if the system fails after performing the first step?**

1.2 Characteristics of DBs

- **Example:** multi-user transactions
  - **Program:**
    Withdraw *x* Euros from Account 1
    1. Read *old balance* from DB
    2. Set *new balance* to *old balance – x*
    3. Write *new balance* back to the DB
  - **Problem:** Dirty Read
    - Account 1 has €500
    - User 1 deduces €20
    - User 2 deduces €80 **at the same time**
  - Without multi-user transactions, Account 1 will have either €480 or €420, but not the correct €400
    - Both users read old value of €500 simultaneously, both deduce either €20 or €80, both write back new value (in random order)

1.2 Characteristics of DBs

- **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, …)



1.2 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**
    - identify the data that needs to be stored and chooses appropriate data structures and representations
    - integrate the needs of all users into the design

1.2 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
- **Data Analyst**
  - Analyzes trends in data to uncover valuable feedback for business operations
  - i.e., discover sales trends, upcoming supply shortages, etc.

- 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

1.2 Database Users

- Persons using the database (End Users)
  - All people who use the DB to do their job
- End Users can be 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
    - e.g. bank tellers, reservation clerks, …

1.2 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
  - e.g. 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
  - e.g. people in middle or senior management

1 Introduction

- What is a Database?
- Characteristics of a Database
- **History of Databases**

1.3 History of DBs

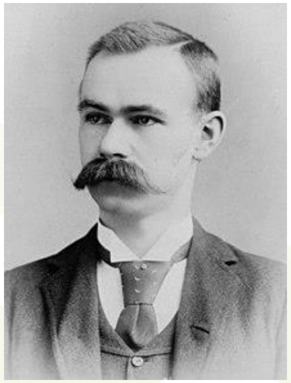
Detour

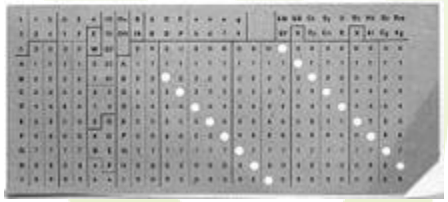
- Databases have an exceptional history of development
  - many synergies between **academic, governmental** and **industrial** research
  - much to be learned from it
  - most popular concepts used today have been invented decades ago

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** tabulating machine
    - the evaluation of 1880's census took 8 years
    - 1890's has been finished after only 6 years
  - leads to the foundation of **IBM**
    - International Business Machines
  - data processing machines soon established in accounting

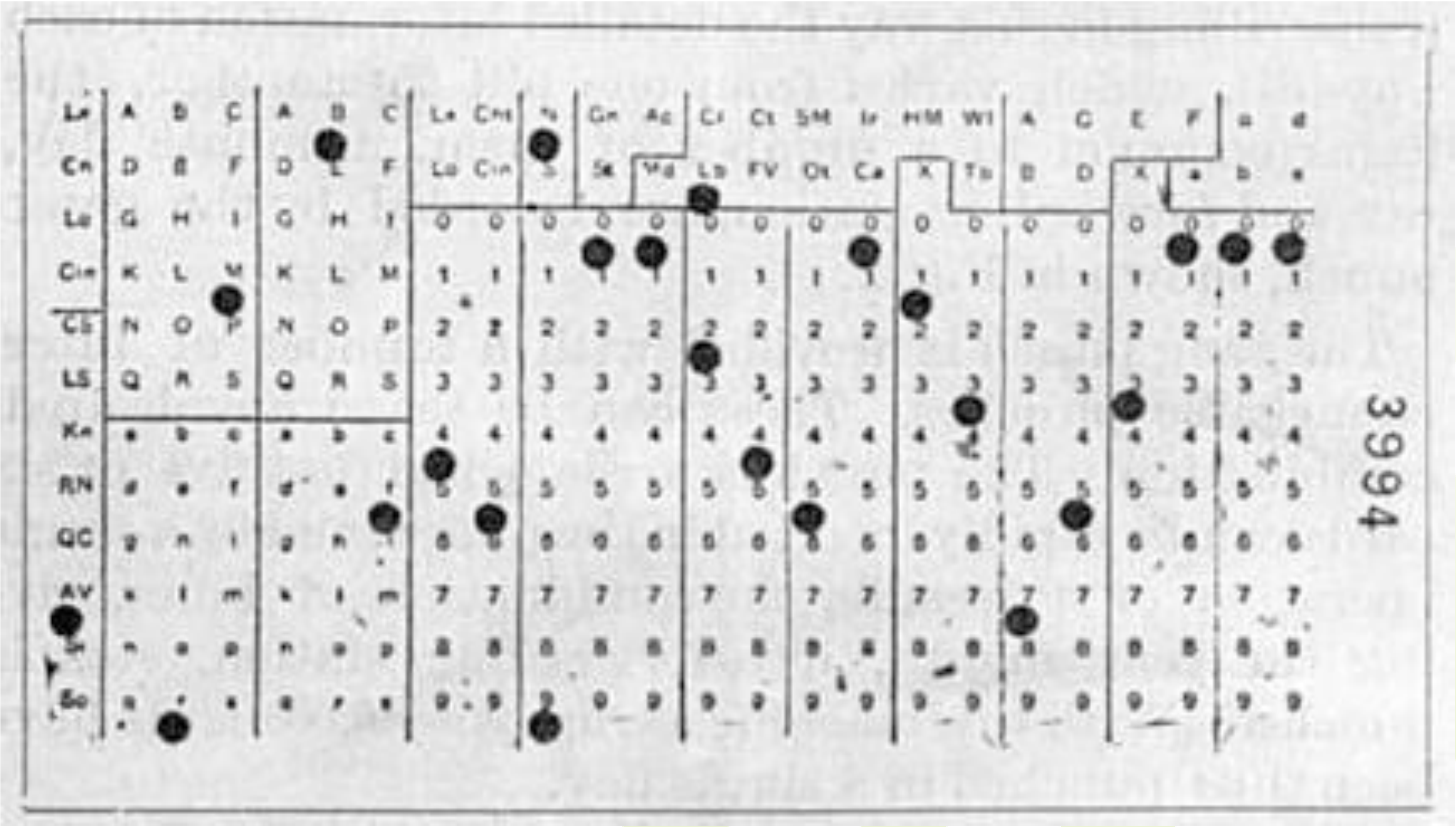




1.3 History of DBs

Detour

- One of Hollerith's punch cards:



1.3 History of DBs

Detour

- Tabulating 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 USA
    - programmable (turing complete)
    - input (programs and data) with tape or punched cards

- In **1959,** USA dominated the (still highly active) punch card machine market
  - within the USA, 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

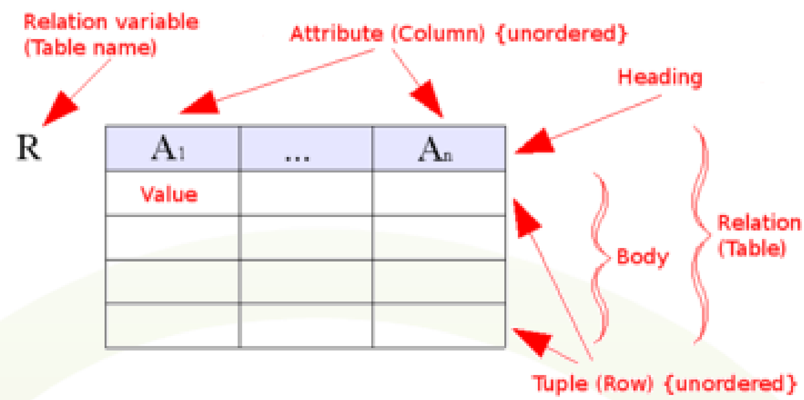
- To turn stored data into a proper **database,** the following goals had to be achieved (McGee, 1981):
  - **Data Consolidation**
    - data must be stored in a central place, accessible to all applications
    - knowledge about relationships between records must be represented

  - **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

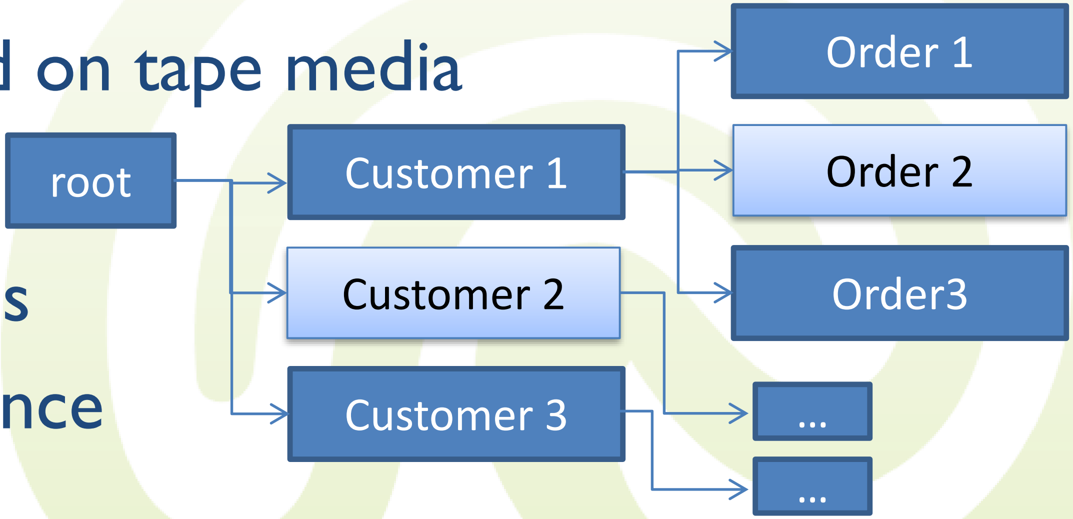
Detour

- **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 used 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 Data Model failed, e.g. for bill-of-material-processing (BOMP)
    - many-to-many relationships needed
    - development of the IBM DBOMP system (1960)
  - result: Network Data 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**
- **Advantages**
  - **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

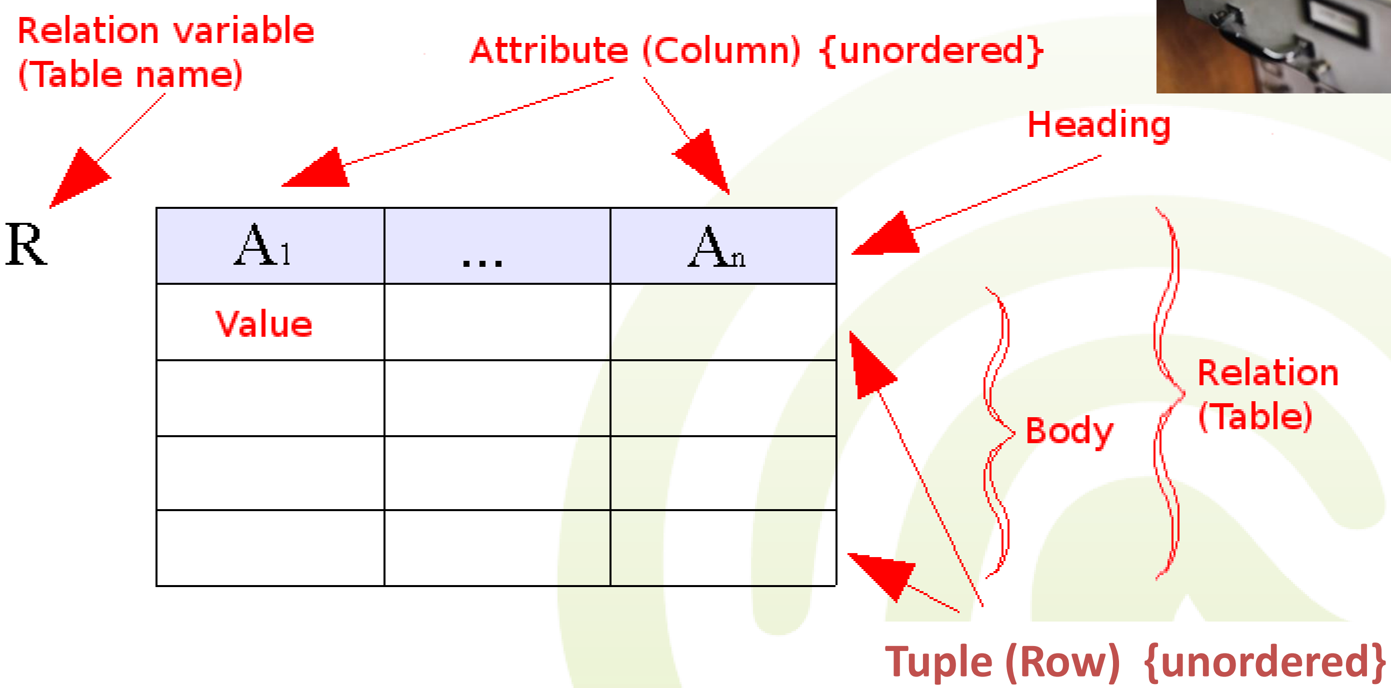
Detour

- **Idea underlying the relational model:**
  - database is seen as a collection of **predicates** over a finite set of **predicate variables**
    - example
      - dislikes(*x*, *y*)
      - dislikes('Ted Codd', 'hierarchical IMS database system') (TRUE)
      - dislikes('IBM', 'hierarchical IMS database system') (FALSE)
    - the set of all true assignments is called a **relation**
    - relations are stored in **tables**
  - contents of the DB are a **collection of relations**
  - queries are also **predicates**
    - queries and data are very similar
    - Allows for **declarative querying**

1.3 History of DBs

Detour

- 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**
  - and became insanely rich – long time in the Top 10 of the richest people
  - in 2015 Oracle ranked second on the list of largest software companies in the world, right after Microsoft

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**
- Ingres Database was developed at UC Berkeley as a research project
  - It became PostgreSQL in 1996

1.3 History of DBs

Detour

- Lately, the so-called NoSQL databases have become popular
  - NoSQL systems are mostly non-relational database systems
    - They might support SQL as a query language
  - Its development was driven by IT companies like Amazon, Google and Facebook/Meta
    - Databases have to deal with huge amounts of data and massive numbers of users
  - NoSQL comprise: key-value stores, document stores, column stores, distributed databases, graph databases, triple stores
    - For details, visit our advanced lectures

1 Summary

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

2 Coming soon…

- **Next Lecture**
  - **Phases of DB Design**
  - **Data Models**
  - **Basic ER Modeling**
    - Chen Notation
    - Mathematical Model