CS354: Database

Dr. Raju Halder

CS354: Database

- Instructors: Dr. Raju Halder/Dr. Mayank Agarwal
  - halder@iitp.ac.in
  - Meeting time: Wednesday

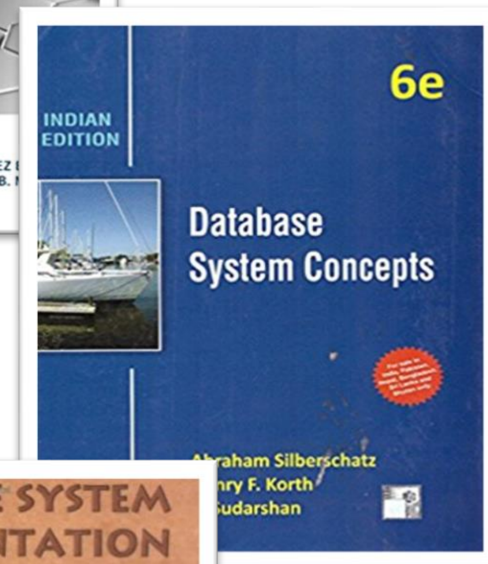
- Teaching Assistants
  - Ashish Kumar Ranjan(ashish.mtcs17@iitp.ac.in)
  - TBA

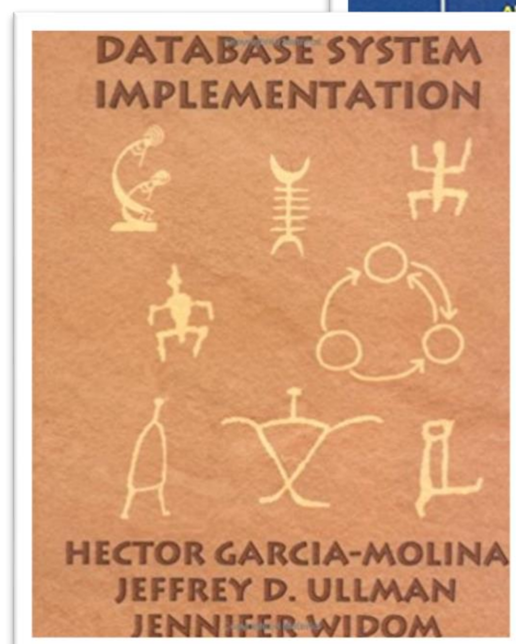
References

- Elmasri Ramez and Navathe Shamkant, **Fundamentals of Database System.** Pearson Education, Seventh edition.

- Silberschatz, Korth, and Sudarshan, **Database System Concepts.** McGraw Hill Education, Sixth edition

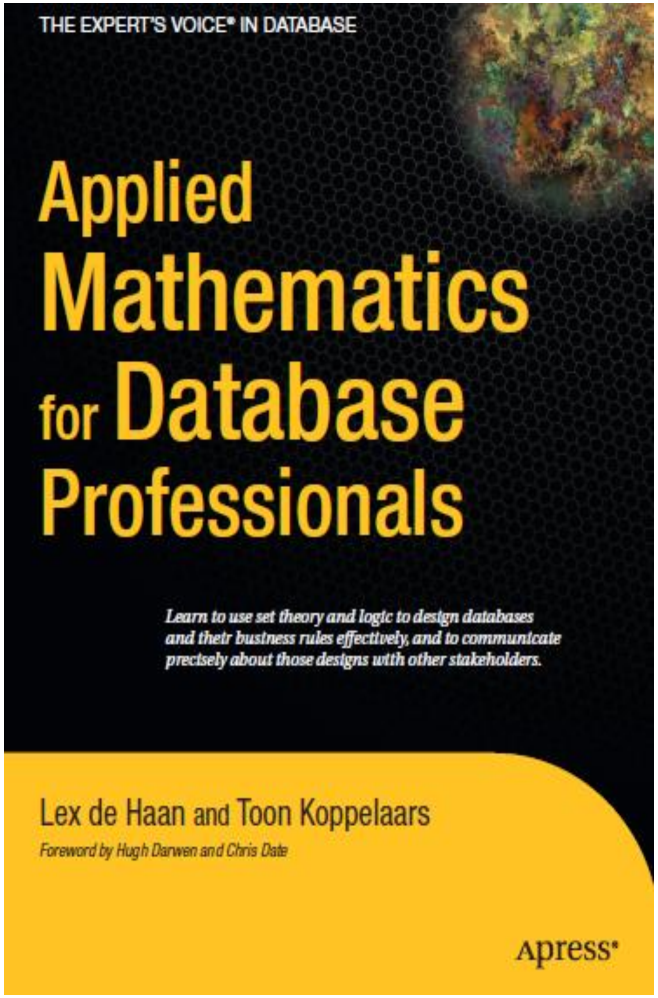
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom, **Database System Implementation.** Pearson Education.





References

- Lex de Haan and Toon Koppelaars, **Applied Mathematics for Database Professionals.** Apress, first edition (June 19, 2007)



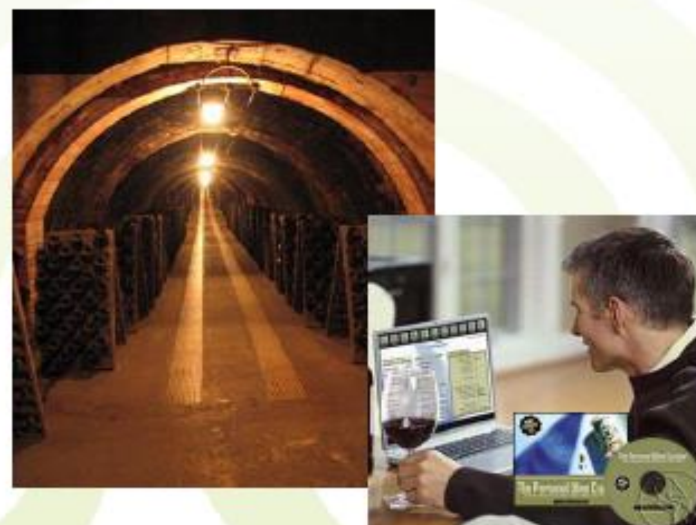
- Introduction
  - What is a database?
  - Characteristics of a database
  - Database users
- History of databases

What is 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
  - …


What is Database

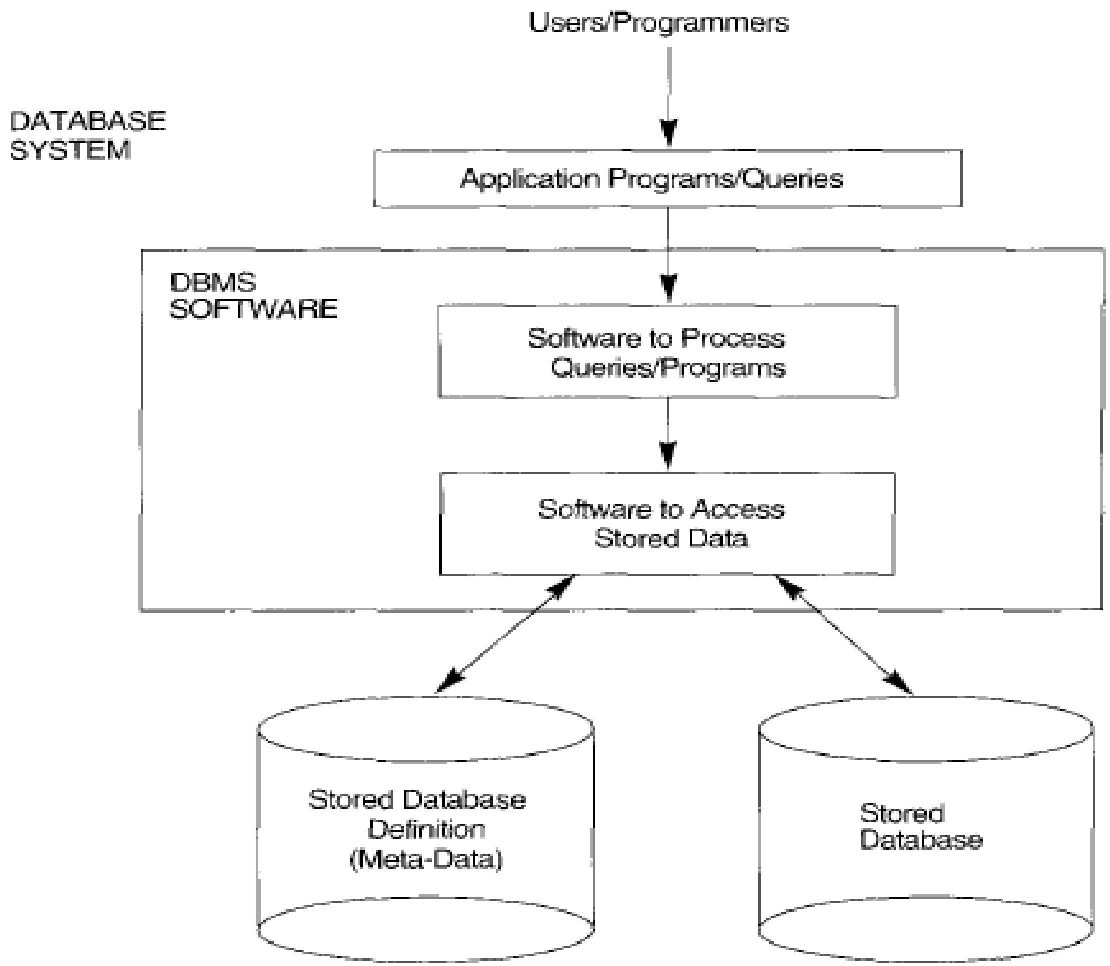
- **Databases** are needed to manage that **vast amount of data**
- A database (**DB**) is a collection of **related data**
  - Represents some aspects of the **real world**
    - **Universe of discourse**
  - Data is logically **coherent**
  - Is provided for an intended group of **users** and **applications**



What is Database

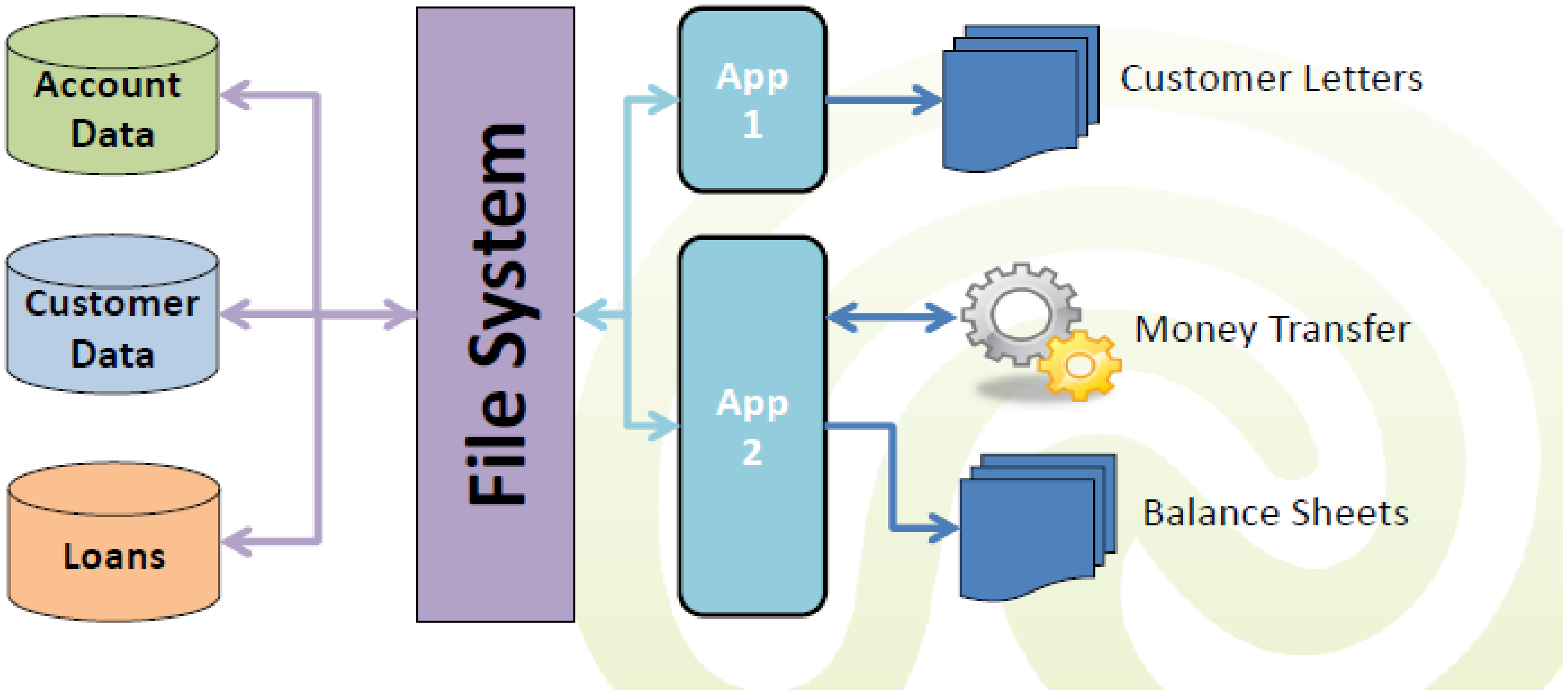
- A database management system (**DBMS**) is a collection of programs to maintain a database, that is, for
  - Definition of data and structure
  - Physical construction
  - Manipulation
  - Sharing/Protecting
  - Persistence/Recovery

A Simplified Database System Environment



File System

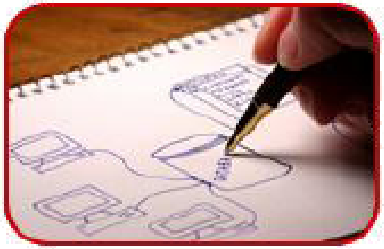
- A file system is not a database!
- File management systems are **physical** interfaces



File System

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

Database Vs. File System



- 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

Database Vs. File System

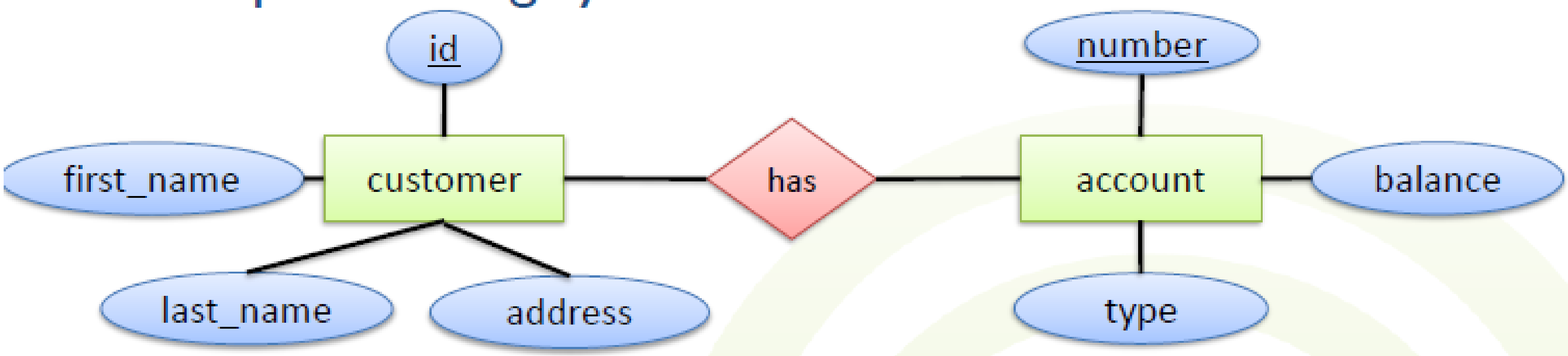
- **DBMS** replaced previously dominant file-based systems in **banking** due to special requirements
  - **Simultaneous** and quick access is necessary
  - Failures and loss of data **cannot** be tolerated
  - Data always has to remain in a **consistent** state
  - Frequent queries and modifications

Database Characteristics

- 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

Database Characteristics

- 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

Database Characteristics

- 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'

Database Characteristics

- 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

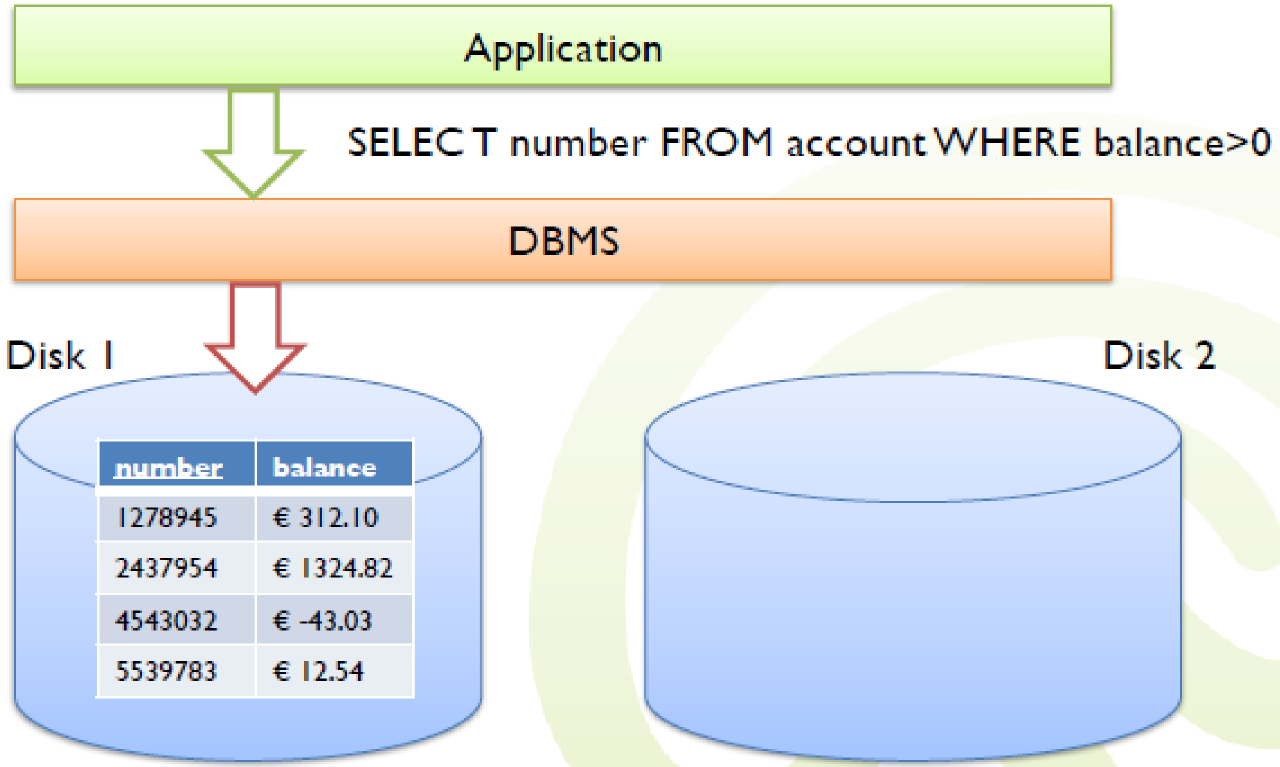
number	type	balance
1278945	saving	€ 312.10
2437954	saving	€ 1324.82
4543032	checking	€ -43.03
5539783	saving	€ 12.54
7809849	checking	€ 7643.89
8942214	checking	€ -345.17
9134354	saving	€ 2.22
9543252	saving	€ 524.89

Database Characteristics

- **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)

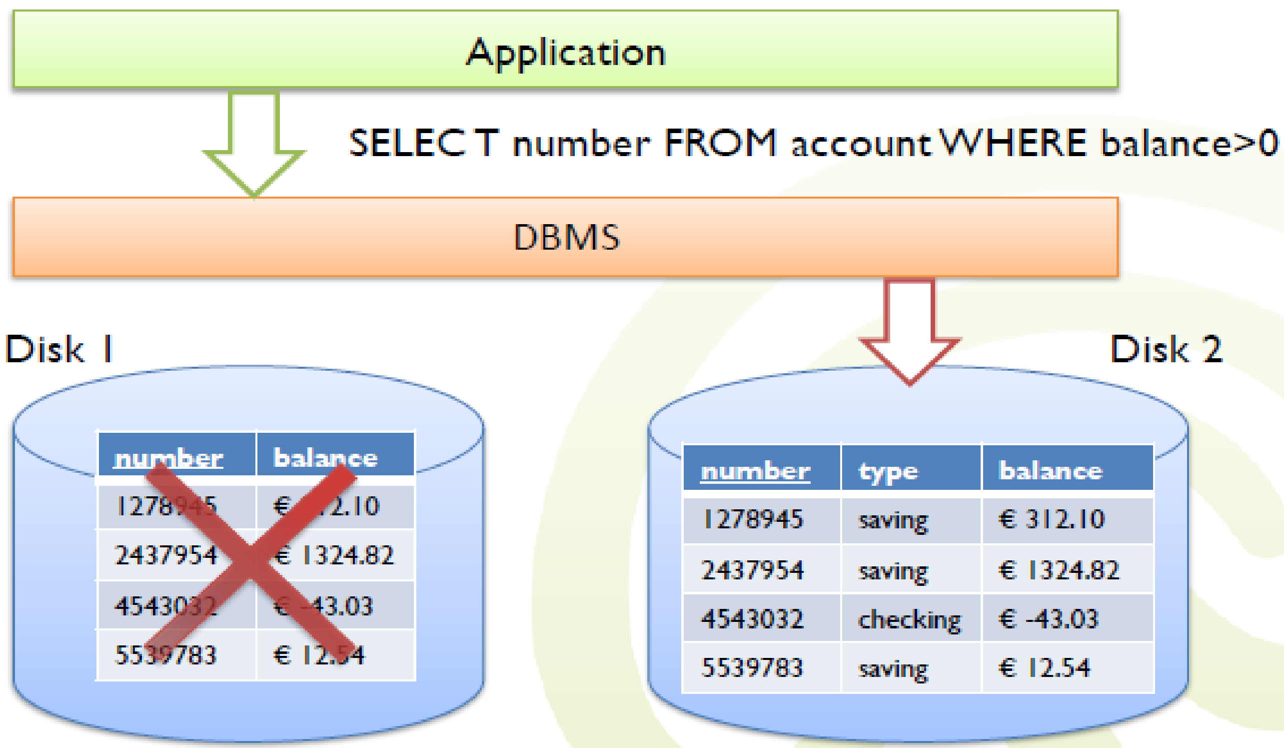
Database Characteristics

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



Database Characteristics

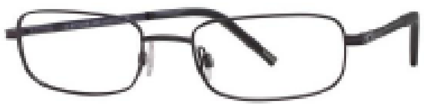
- **Example:** Schema can be changed and tablespace moved without being noticed by app



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

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

Database Characteristics



- 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

Database Characteristics

- 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

Database Characteristics

- **Sharing** of data and support for **atomic multi-user transactions**
  - Multiple users and applications may access the DB at the same time
  - **Concurrency control** is necessary for maintaining consistency
  - **Transactions** need to be **atomic** and **isolated** from each other

Database Characteristics

- 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

Database Characteristics

- 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

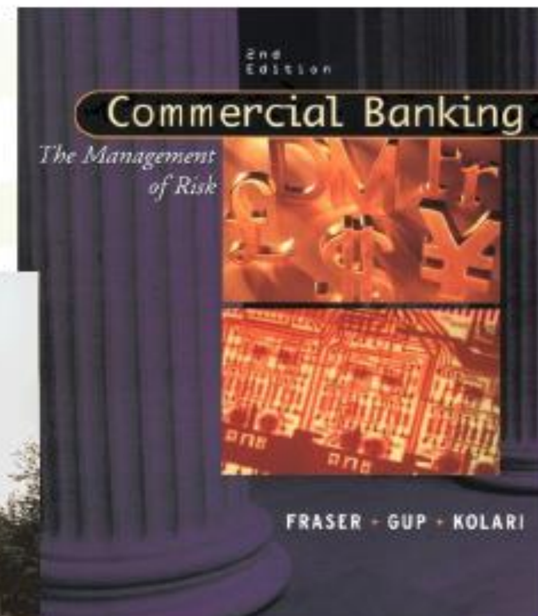
  - **But what happens if the system fails after performing the first step?**

Database Characteristics

- **Example:** Multi-user transactions
  - Program: Withdrawal of amount *x* from Account 1
    1. Read old balance from DB
    2. New balance := old balance – *x*
    3. Write new balance back to the DB
  - Problem: Dirty Read
    - Account 1 has €500
    - User 1 wants to withdraw €20
    - User 2 wants to withdraw €80 **at the same time**
  - Without multi-user transactions,
    the account's balance is either €480 or €420,
    but not €400 (which would have been correct)

Database Characteristics

- **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, …)



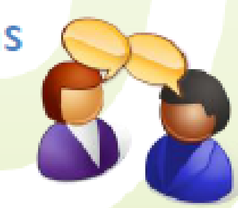
EN 1.3

- Introduction
  - What is a database?
  - Characteristics of a database
  - Database users
- History of databases

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



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

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, …

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**

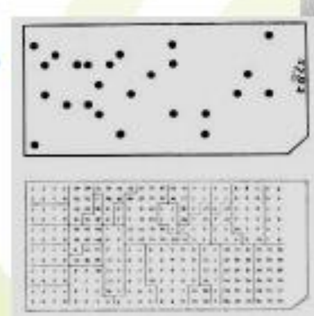
- Use DB only from time to time, but need to perform different tasks
- Are familiar with query languages
- Examples: People in middle or senior management

History of DBs

- 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

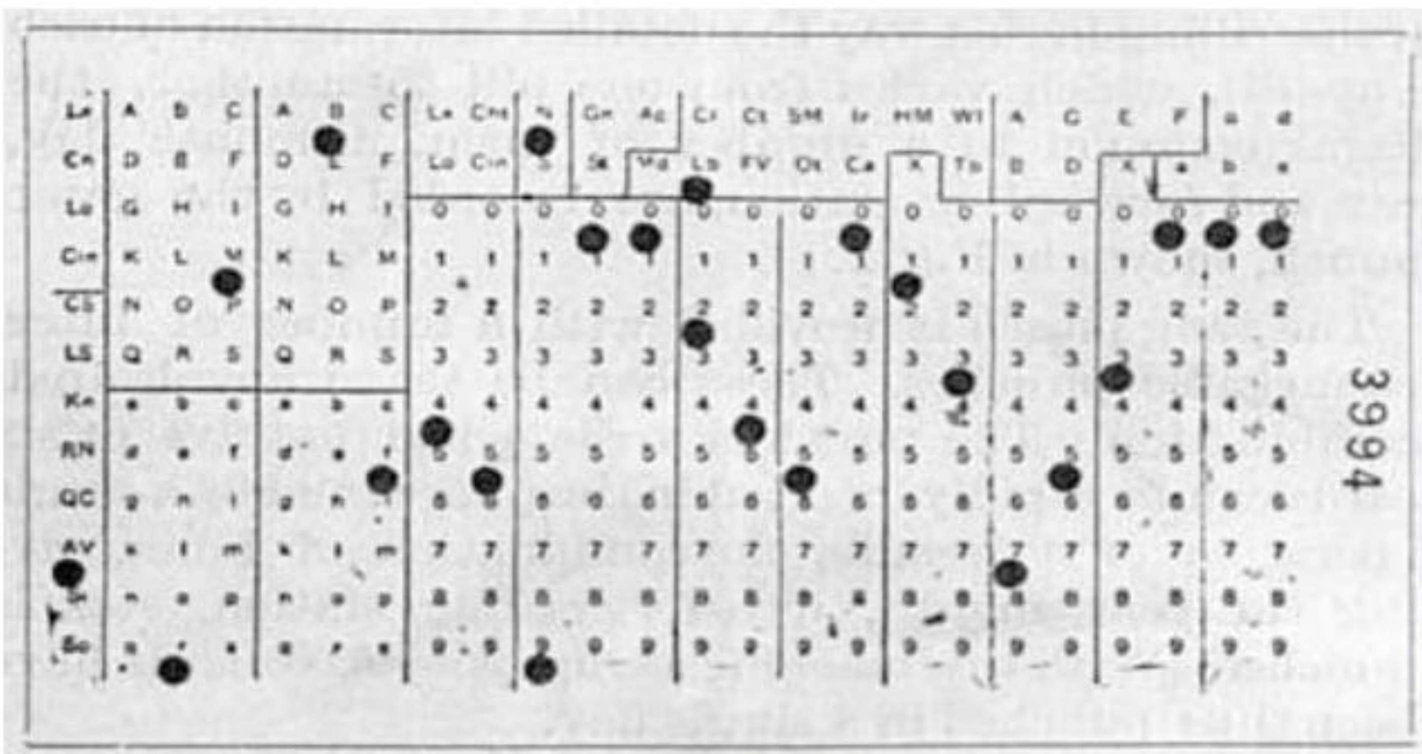
History of DBs

- 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



History of DBs

- One of Hollerith's punch cards:



History of DBs

- During WWI, many **data collection**, **sorting** and **reporting** tasks in industry and governmental organizations was performed using **punch cards**
- **1935:** U.S. Social Security Act required continuous report on all 26 million governmental employees
  - "World's biggest bookkeeping job"
  - Mechanical punch card systems not powerful enough
    - IBM develops the electric **UNIVAC I** punch card processor in 1951

History of DBs

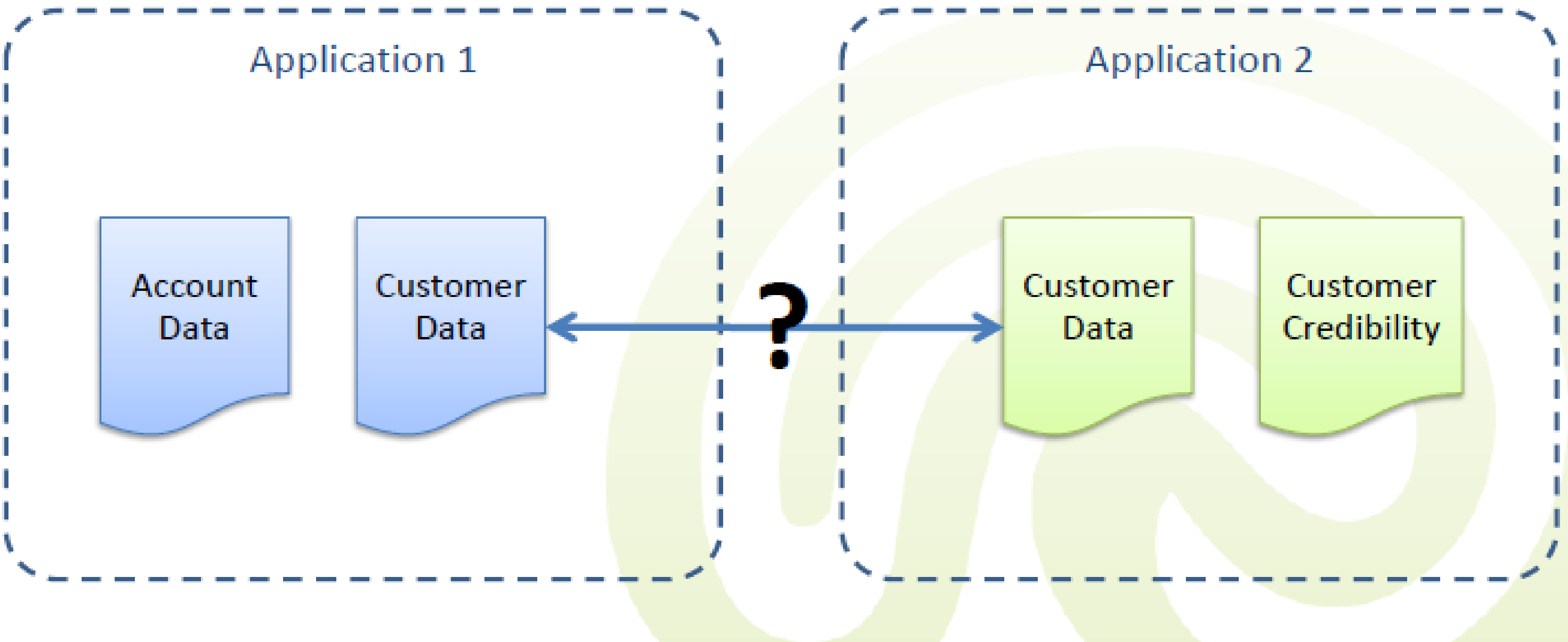
- In **1959,** U.S. dominated the (still highly active) punch card machine market
  - Within the U.S., the Pentagon alone used more than 200 data processing computers, costing $70 million per year
- In **1964,** the term "data base" appeared for the first time in military computing using **time sharing systems**
  - Data could be shared among users
  - Data model very close to punch cards
  - **Master files** bound to a specific application

History of DBs

- **Master files**
  - Used to maintain continuity between program runs
  - Each application had its own master files
  - Similar data needed by multiple applications had to be duplicated
    - Consistency problems when updating data
  - 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

History of DBs

- **Master files**
  - Highly hardware-oriented approach
  - Data stored in independent flat files



History of DBs

- 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

History of DBs

- **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

History of DBs

- **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
  - Data may be organized in a **tree structure**
    - Initially, tree had maximum depth of two

History of DBs

- **Hierarchical data model**
  - Each type of record has one or multiple own files/tables
  - Hierarchical **one-to-many** relationships
  - Vaguely resembles a file system organization

Customer

customerID	Name
1000	Mickey Mouse
1001	Scrooge McDuck
1002	Donald Duck
1003	Goofy

Accounts – Mickey Mouse

number	balance
1000	€ 73.68

Accounts – Scrooge McDuck

number	balance
6000	$ 934,3243,435,322
6001	€ 4,543,123,987
6002	¥ 12,432,355,112

History of DBs

- **Hierarchical data model**
  - **Advantages**
    - 1:n relationships can be expressed
    - Can easily be stored on tape media
  - **Disadvantages**
    - No n:m relationships
    - No data independence

History of DBs

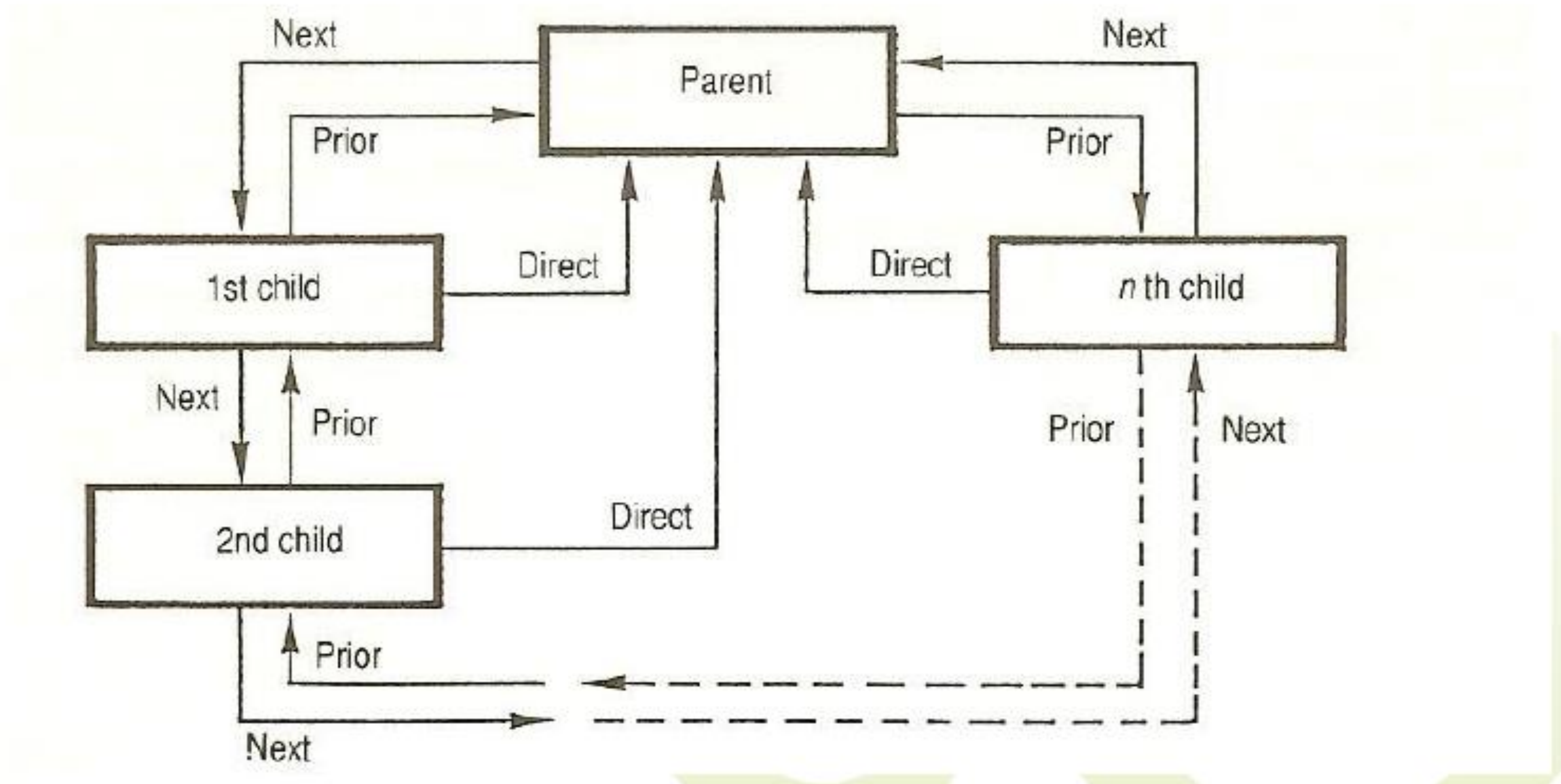
- **Network data model**
  - In the mid-1960th, direct access storage devices (DASD) gained momentum
    - Primarily hard disks
    - More complex storage schemes possible
  - Hierarchical model failed,
    e.g. for bill-of-material-processing (BOMP)
    - Many-to-many relationships needed
    - Development 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

History of DBs

- **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

History of DBs

- **An example of the network data model:**



History of DBs

- What's wrong with all that?
  - Strong degree of **hardware dependence**
  - **No proper abstraction** of data
  - No decoupling of data and its application
  - Each database needed to be **"hand-crafted"** for its application
  - To change something in the data-schema, "a sharp-looking guy in a white shirt and black rims had to do the programming by hand"
    - No formal/structural/mathematical foundation → **no high-level data languages**

History of DBs

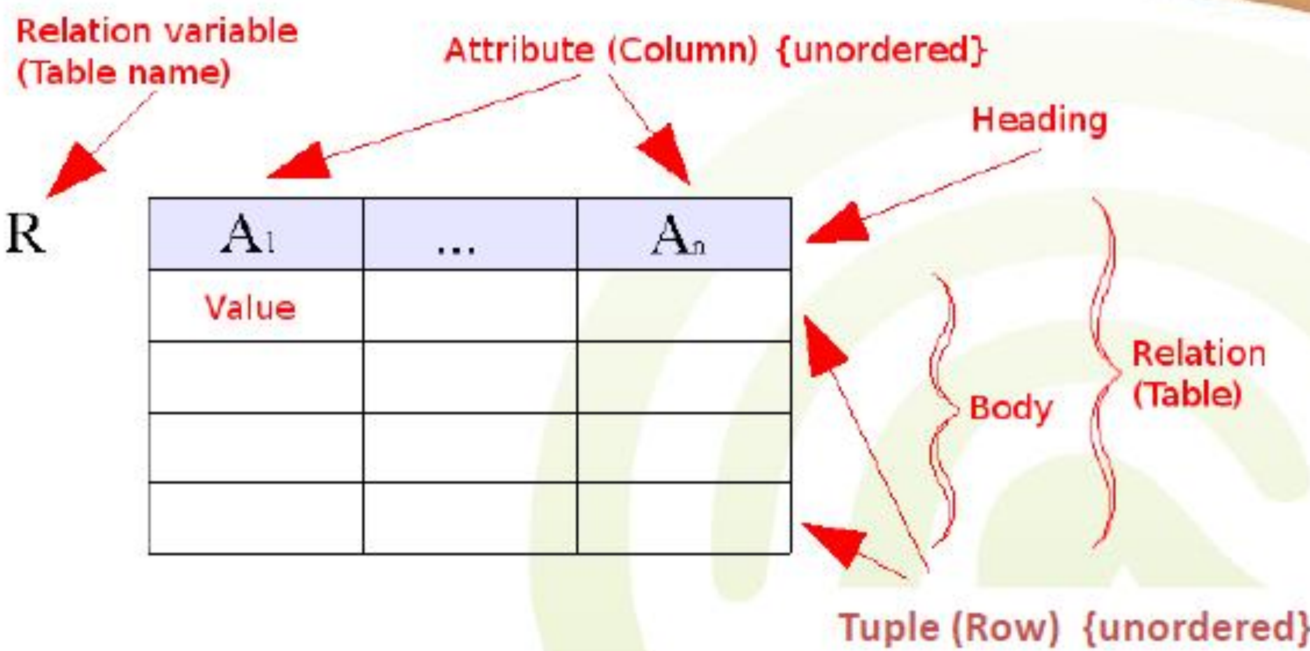
- **The relational data model**
- Published by **Edgar F. “Ted” Codd** in 1970, after several years of work
  - *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, 1970
  - Employee of IBM Research
    - IBM **ignored** his idea for a long time as not being “practical” 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

History of DBs

- **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', 'B. Köhncke') (TRUE)
      - is_supervisor_of('C. Lofi', 'B. Köhncke') (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**

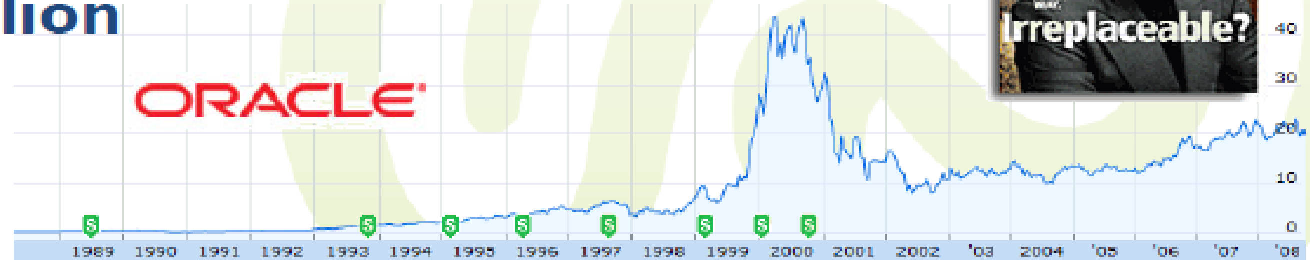
History of DBs

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



History of DBs

- 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
  - Oracle's expected net income in 2009: **$5.59 billion**



History of DBs

- 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 beginning in 2003
  - JD Edwards, PeopleSoft, Siebel Systems, BEA, …

History of DBs

- 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

History of DBs

Company	Revenue Estimates for Database Products, 2006 (in million USD)	Market Share (in %)
Oracle	7,168.0	47.1
IBM	3,204.1	21.1
Microsoft	2,654.4	17.4
Teradata	494.2	3.2
Sybase	486.7	3.2
Other Vendors	1,206.3	7.9
Total	15,213.7	100.0

Source: Gartner Research, 2007

Gartner

History of DBs

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

History of DBs

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 …**

History of DBs

- **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 …

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
- (Several groups of database users)
- (History of databases)