Lecture 1 - Introduction to Database Systems

SQL - a simple, high-level language for querying data declarative
Relational data - an effective, formal foundation based on relational algebra and calculus
RDBMS - an efficient, low-level execution environment tailored towards the data processing

developed since 1960's - banks, airlines, military, space

Database - a collection of related data, used to manage it
- → data represents aspects of the real world (universe of discourse)
- → data is logically coherent
- → data is provided to users and applications

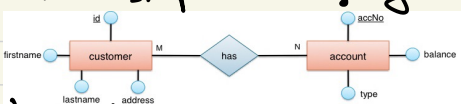
Database Management Systems (DBMS) - software maintaing DBs
- → definition of data and structure
- → physical construction
- → data manipulation
- → sharing / protecting
- → persistance / recovery

Relational Databases
- different underlying data models - Relational Data Model (Network, Hierarchial, Object-Oriented)
- developed since 1970
  - → Declarative Query Language SQL
  - → ACID Transactions
  - → Data Recovery

set of valuable features:
- → strict data modeling
- → controlled redundancy
- → data normalization
- → data consistency & integrity constrains
- → SQL: simple & powerful query language
- → effective and secure data sharing
- → backup and recovery

Databases – well-structured (ER-Model)
- Catalog (data dictionary) contains all meta-data
- defines the structure of the data in the database

simple banking system



Customer

ID	firstname	lastname	address
1	Aard	Vark	123AB
2	Bandi	Coot	231CX
3	Cham	Eleon	12dXX

Account

accNo	type	balance
10	Check	0
11	Saving	232
12	Check	-232

Customer2Account

ID	accNo
1	10
1	11
2	12

Characteristics of Relational Databases

Databases aim at efficient manipulation of data
- physical tuning allows for good data allocation
- indexes speed up search and access
- query plans are optimized for improved performance

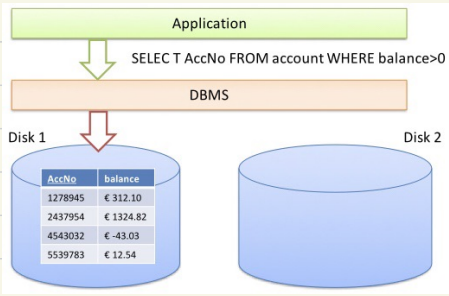
Data Independence

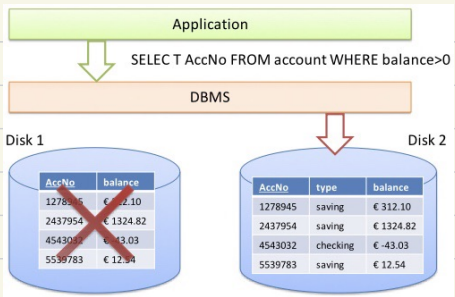
Databases employ data abstraction by providing data models

Applications work only on the conceptual representation of data
- → data is strictly typed (Integer, ...)
- → details on where data is actually stored and how it is accessed is hidden by the DBMS
- → applications can access and manipulate data by invoking declarative operations

DBMS-controlled parts of the file system are strongly protected against outside manipulation

Simple Index





Supports multiple views on data
→ views provide a different perspective of the DB
- user's conceptual understanding or task-based excerpt (aggregations)
- security considerations and access control (projections)

→ for app, 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
- computed at query time or materialized upfront

Original Table

AccNo	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

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

Checking View

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

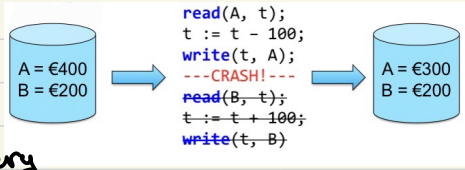
Transactions

transaction → a unit of work, possibly containing multiple data accesses and updates, that must commit or abort as a single unit, and follows the ACID principles

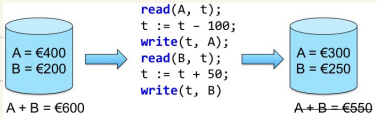
Sharing of data and support for atomic multi-user transactions
→ multiple users and applications may access the DB at the same time
→ concurrency control is necessary for maintaining consistency & problems:
→ recovery
→ concurrency

ACID principle

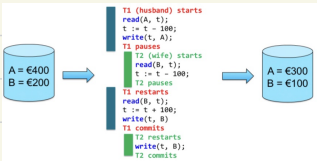
- **Atomicity**
  - → transaction is either executed completely (commit) or not at all (abort)
    - ↳ can undo during recovery



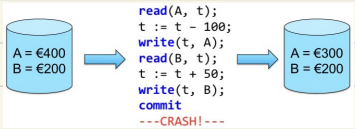
- **Consistency**
  - → transaction transforms a consistent database state into a (possibly different) consistent database state



- **Isolation**
  - → transaction is executed in isolation



- **Durability**
  - → a successfully completed transaction has a permanent effect on the DB



Data Models

Data Model – describes data objects, operations and their effects
Data Definition Language (DDL) – create table/view, constraint/check
Data Manipulation Language (DML) – select, insert, delete, update
DML & DDL → usually clearly separated, since they handle data & meta-data respectively

Schema – describes a part of the structure of the stored data as tables, attributes, views, constrains, relationships
System Catalogs – a collection of schemas; contain special schemas describing the schema collection
Schemas – describe the structure of part of the DB data
- → entity types as tables and their attributes
- → types of attributes and integrity constrains
- → relationships between entity types as tables
- → schemas are intended to be stable and not change often
- → basic operations (selections, insertions, updates)
- → optionally user defined operations (User Defined Functions (UDFs), stored procedures) and types (UDTs)

Schema → intensional DB
Instance → extensional DB; actually stored data

Intensional DB

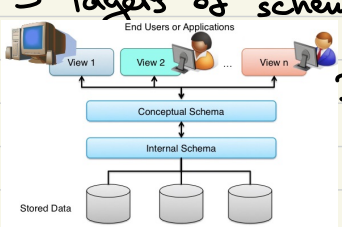
Table account

Primary key AccNo
Check balance > 0

Extensional DB

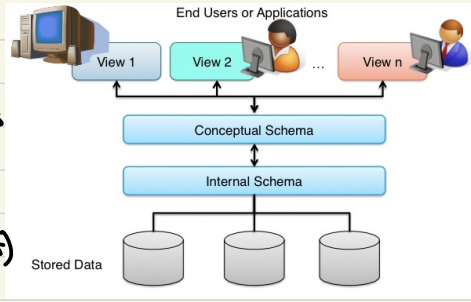
AccNo	type	balance
1278945	saving	€ 312.10
2437954	saving	€ 1324.82
4543032	checking	€ 43.03
5539783	saving	€ 12.54
7809849	checking	€ 7643.89
8942214	checking	€ 345.17
9134354	saving	€ 2.22
9543252	saving	€ 524.89

3 layers of schemas:



Internal Schema (physical layer)
- → describes the physical storage access and paths
- → uses physical models

Conceptual Schema (logical layer)
→ describes structure of the whole DB, hiding physical details
→ uses logical data models



External Schema (presentation layer)
→ describes parts of the DB structure for a certain user group as views
→ hides conceptual details

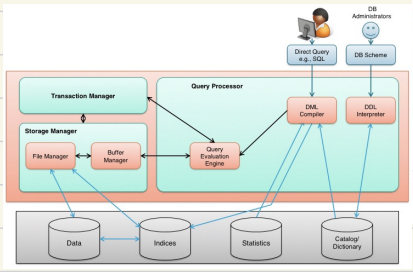
Data Independence

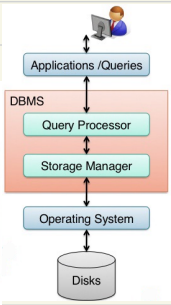
- ability to change schema of one level without changing others
- Logical Data Independence
  - → change of conceptual schema without change of external schemas
  - → ex.: adding attributes, changing constraints
  - → dropping a used in view attribute violates that independence
- Physical Data Independence
  - → changes of the internal schema do not affect the conceptual schema
    - important for reorganizing data on the disk
    - adding or changing access paths
  - → physical tuning is one of the most important maintenance task
  - → physical independence is also supported by having a declarative query language in relational databases

Anatomy of a Database System

High-level architecture

- Database characteristics lead to layered architecture
- Query Processor
  - Query Optimization
  - Query Planning
- Storage Manager
  - Access paths
  - physical sets, pages, buffers
  - accesses disk through OS





- The storage manager provides the interface between the data stored in the database and the application programs and queries submitted to the system
- The storage manager is responsible for:
  - interaction with the file manager
  - efficient storing, retrieving and updating the data
- Tasks:
  - storage access
  - file organization
  - indexing and hashing
- The query processor passes queries, optimizes query plans and evaluates the query
  - alternative ways of evaluating due to equivalent expressions
  - different algorithms for each operation
  - enormous cost difference between good and bad ways of evaluating
- Needs to estimate the cost of operations
  - depends critically on statistical information about DBMs relations
  - need to estimate statistics for intermediate results to compute cost of complex expressions (join order, ...)

- Transaction - a collection of operations that perform a single logical function in a database application
- Transaction Manager
  - ensures that the database remains in a correct state despite system failures (power/transaction/OS failures and crashes)
  - controls the interaction among concurrent transactions to ensure consistency

History of Databases

- synergies between academic, governmental and industrial research

1. Beginning
   - US Bureau of Census (1880), Herman Hollerith
   - machine for storing census data
   - Punch card tabulating machine
   - leads to foundation of IBM (International Business Machines)
   - data processing machines soon established in accounting

   1.1. 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 steps

2. Development
   - UNIVAC 1, 1951 (IBM)
     → first commercial computer produced in the USA
     - programmable (Turing complete)
     - input (programs and data) with tape or punched cards

   2.1. USA-dominated punch card machine market (1959)
   - Pentagon using 200 data processing computers, costing €70M annually

   2.2. Term "data base", military computing using time sharing systems (1964)
   → data shared among users
   → data bound to one specific application
   - duplicated data for multiple applications to use
   - consistency problems when updating data
   → data structure highly-dependent on software and (low-level) programming language used
   - inspired by punch cards and optimized for magnetic tapes

- no relationships between different records stored

2.1. Proper database:

2.3.1. Data Consolidation

- data stored in a central place, accessable to all apps
- knowledge about relationships between records

2.3.2. 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

2.3.3. Data Protection

- data must be protected against loss and abuse

2.4 Implementation

2.4.1. Data Consolidation motivated the development of data models

- A data model describes the organization data that interact with it can be stored, but also the way how we can
  - Hierarchical Data Model
  - Network Data Model
  - Relational Data Model
  - Object-oriented Data Model
  - Semantic Data Model

2.4.2. Data Independence inspired the development of query models and high-level languages

- Relational Algebra, SQL

2.4.3. Data Protection led to development of transactions, backup schemes, and security protocols

3. Data Models

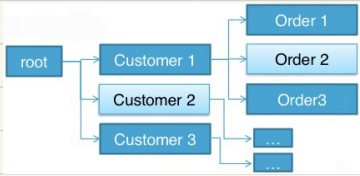
3.1. Hierarchical Data Model

- first appearance in IBM's IMS (1966), Apollo Program
- benefits from advances in hardware design
- random access main memory and tape media available

- each type of record has some defined structured data
- hierarchical one-to-many relationships

3.1.1. Advantages
- 1:n relationships can be expressed
- can easily be stored on tape media

3.1.2 Disadvantages
- no n:m relationships
- no Data Independence



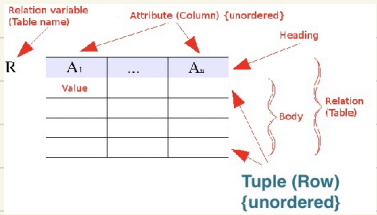
3.2. Network Data Model
- direct access storage devices (DASD), mid-1960th
  - primarily hard disks
  - more complex storage schemes possible
- Hierarchical Data Model failed, bill-of-material-processing (BOMP)
  - many-to-many relationships needed
  - development of IBM DBOMP system (1960)
- result: Network Data Model
  - two types of files: master and chain
  - chain file entries could chain master file entry to other
- model standardized by Charles W. Bachman, CODASYL, 1969
  CODASYL = Conference of Data Systems Languages
- allowed for more natural modeling of associations

3.2.1. Advantages
- many-to-many relationships

3.2.2. Disadvantages
- no declarative queries
- queries must state the data access path

3.3 Relational Data Model
- database is seen as a collection of predicates over a finite set of predicate variables
- the set of all true assignments is called a relation

- relations are stored in tables
- contents of the DB are a collection of relations
- quiries are also predicates
  - quiries and data are very similar
  - allows for declarative querying
- 1970s, IBM, System R
  - first implementation of the SQL declarative query language (SEQUEL)
  - research prototype, base of IBM DB2
- de-facto standard of modern databases



4. Oracle, 1977, Lawrence J. Ellison
5. Beyond Relational Data Model
   - data models based on formal logic
     - Deductive Databases and Expert Systems
   - Object-Oriented Data Models
     main idea: Object-oriented design
     - easy integration in OO programming languages
     - mostly integrated into relational model
   - semi-structured data models
     most important: XML
     - allows a large degree of structural freedom
6. Current Trends
   - NoSQL Systems and Polyglot Persistence
   - Stream Proccessing Systems
   - specialized databases for specialized problems
     - specialize on scalability
     - specialize on transactional performance by exploiting main memory
   - combine different types databases, each focusing on its own strength