Part I

What are Databases?
What are Databases?
Educational Objective for Today . . .

- Motivation for using database systems
What are Databases?

Educational Objective for Today . . .

- Motivation for using database systems
- Knowledge of basic architectures
What are Databases?

- Data = logically grouped pieces of information
- Base\(^1\)

: *the bottom or lowest part of something*: the part on which something rests or is supported
: *something (such as a group of people or things) that provides support for a place, business, etc.*

Base of a column in architecture

\(^1\)http://www.merriam-webster.com/dictionary/base
Areas of Application
How are Databases Managed?

Without databases

- Every application manages its own data
- Data is stored multiple times \( \Rightarrow \) redundancy
- Problems
  - Waste of storage space
  - “Forgetting” of changes
  - No centralized, “standardized” data management
Problems of Data Redundancy

- Other software systems cannot process large amounts of data efficiently.
- Many users or applications cannot access the same data in parallel without interfering with each other.
- Application developers / users cannot develop / use applications without knowing:
  - internal representation of data
  - storage media or computers

  (no data independence)

- No data security; potential loss of data.
Idea: Data Integration Using Database Systems

Database Management System = Software for Managing Databases

DBS = Database System

structured data, which is managed by DBMS
Motivation

- Database systems are center piece of modern IT systems

... ubiquitous

- Database specialists are in high demand
Questions

1. How to organize (model and use) data?
2. How to store data safely and persistently?
3. How to process huge amounts of data ($\geq$ terabytes) efficiently?
4. How can many users ($\geq$ 10,000) access data concurrently?
Questions

1. How to organize (model and use) data?
2. How to store data safely and persistently?
3. How to process huge amounts of data (≥ terabytes) efficiently?
4. How can many users (≥ 10,000) access data concurrently?
Principles: Codd’s Nine Rules

1. **Integration**: uniform, non-redundant data management
2. **Operations**: insert, query, update, delete
3. **Catalog**: access to the database description in the data dictionary
4. **User views**: different users/applications must be able to have a different perception of the data
5. **Integrity**: ensure conformity of database contents with real world
6. **Security**: prevention of unauthorized access
7. **Transactions**: multiple DB operations handled as an atomic unit
8. **Synchronisation**: coordination of concurrent transactions
9. **Data backup**: data recovery after system errors
Data Independence and Schemata

- Based on coarse DBMS architecture
- Decouple user and implementation view
- Goals include:
  - Separate modeling view from internal storage
  - Portability
  - Simplify tuning
  - Standardized interfaces
Schema Architecture

- External Schema 1
- External Schema N
- Conceptual Schema
- Internal Schema

Query Processing → Data Representation
Schema Architecture /2

Connection between

- Conceptual schema (result of data definition)
- Internal schema (definition of file structure and access paths)
- External schemata (result of view definition)
- Application programs (result of application programming)
Schema Architecture /3

- Distinction schema — instance
  - Schema (metadata, data description)
  - Instance (user data, database state or shape)

- Database schema consists of
  - Internal, conceptual, external schemata and application programs

- Conceptual schema contains, e.g.:
  - Structure descriptions
  - Integrity descriptions
  - Authorization rules (DB accesses that a user may perform)
Data Independence /2

- Stability of user interface with respect to changes
- **Physical**: Changes to file structure or access paths do not influence the conceptual schema
- **Logical**: Changes to conceptual schema and certain external schemata do not influence other external schemata or application programs
Data Independence /3

Potential impact of changes to the conceptual schema:
- external schemata may be affected (changing attributes)
- application programs may be affected (recompilation of application programs, adaptations may be necessary)

But: necessary changes are recognized and monitored by the DBMS
### Application Example: Music Store

<table>
<thead>
<tr>
<th>Musician</th>
<th>Title</th>
<th>Year</th>
<th>Tracks</th>
<th>Price</th>
<th>Critique(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neil Young</td>
<td>Living With War</td>
<td>2006</td>
<td>10</td>
<td>9.99€</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Heart of Gold</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Heart of Gold</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Heart of Gold</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Captain and The Kid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elton John</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In a move that deliberately echoes the rush release of "Ohio" in the wake of the Kent State shootings, Neil Young bashed out his 2006 protest record Living With War in a matter of days, sometimes recording songs the day they were written, and then seized the opportunities of the internet to release them directly to the public.

In a move that deliberately echoes the rush release of "Ohio" in the wake of the Kent State shootings, Neil Young bashed out his 2006 protest record Living With War in a matter of days, sometimes recording songs the day they were written, and then seized the opportunities of the internet to release them directly to the public.
Layer Architecture by Example

- Conceptual view: presentation in tables (relations)

<table>
<thead>
<tr>
<th>Artist</th>
<th>MNr</th>
<th>Name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td></td>
<td>Apocalyptica</td>
<td>Finland</td>
</tr>
<tr>
<td>104</td>
<td></td>
<td>Subway To Sally</td>
<td>Germany</td>
</tr>
<tr>
<td>105</td>
<td></td>
<td>Rammstein</td>
<td>Germany</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Album</th>
<th>ANr</th>
<th>Title</th>
<th>Year</th>
<th>Genre</th>
<th>MNr → Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1014</td>
<td></td>
<td>Amplified</td>
<td>2006</td>
<td>Rock</td>
<td>103</td>
</tr>
<tr>
<td>1015</td>
<td></td>
<td>Nord Nord Ost</td>
<td>2005</td>
<td>Rock</td>
<td>104</td>
</tr>
<tr>
<td>1016</td>
<td></td>
<td>Rosenrot</td>
<td>2005</td>
<td>Rock</td>
<td>105</td>
</tr>
<tr>
<td>1021</td>
<td></td>
<td>Engelskrieger</td>
<td>2003</td>
<td>Rock</td>
<td>104</td>
</tr>
<tr>
<td>1025</td>
<td></td>
<td>Reflections</td>
<td>2006</td>
<td>Rock</td>
<td>103</td>
</tr>
</tbody>
</table>
Layer Architecture by Example /2

External view: data in a flat relation

<table>
<thead>
<tr>
<th>ANr</th>
<th>Title</th>
<th>Year</th>
<th>Genre</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1014</td>
<td>Amplified</td>
<td>2006</td>
<td>Rock</td>
<td>Apocalyptica</td>
</tr>
<tr>
<td>1015</td>
<td>Nord Nord Ost</td>
<td>2005</td>
<td>Rock</td>
<td>Subway To Sally</td>
</tr>
<tr>
<td>1016</td>
<td>Rosenrot</td>
<td>2005</td>
<td>Rock</td>
<td>Rammstein</td>
</tr>
<tr>
<td>1021</td>
<td>Engelskrieger</td>
<td>2003</td>
<td>Rock</td>
<td>Subway To Sally</td>
</tr>
<tr>
<td>1025</td>
<td>Reflections</td>
<td>2006</td>
<td>Rock</td>
<td>Apocalyptica</td>
</tr>
</tbody>
</table>
## Layer Architecture by Example /3

External view: data in a hierarchically structured relation

<table>
<thead>
<tr>
<th>Artist</th>
<th>Album</th>
<th>Year</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apocalyptica</td>
<td>Amplified</td>
<td>2006</td>
<td>Rock</td>
</tr>
<tr>
<td></td>
<td>Reflections</td>
<td>2003</td>
<td>Rock</td>
</tr>
<tr>
<td>Subway To Sally</td>
<td>Nord Nord Ost</td>
<td>2005</td>
<td>Metal</td>
</tr>
<tr>
<td></td>
<td>Engelskrieger</td>
<td>2003</td>
<td>Rock</td>
</tr>
<tr>
<td>Rammstein</td>
<td>Rosenrot</td>
<td>2005</td>
<td>Rock</td>
</tr>
</tbody>
</table>
Layer Architecture by Example /4

- Internal presentation

Tree-like Access on Album Number
Partial Storage of Data Sets in a Tree
Space for Data Sets Overflow

1000 1500 2000
1014 Amplified 2006
1015 Nord Nord Ost 2005
19.99 Rock 103 ....
15.99 Rock 104 ....
System Architectures

- Description of components of a database system
- Standardized interfaces between components

- Architecture proposals
  - ANSI-SPARC architecture
    - three layer architecture
  - Five layer architecture
    - describes transformation components in detail

Lecture “Database Implementation Techniques”
ANSI-SPARC Architecture

- ANSI: American National Standards Institute
- SPARC: Standards Planning and Requirement Committee
- Proposal from 1978
- Basically refines coarse architecture
  - Internal layer / operation system refined
  - Multiple interactive and programming components
  - Interfaces named and standardized
What are Databases?

Architectures

ANSI-SPARC Architecture /2

External Layer

- Query
- Updates

Conceptual Layer

- Optimizer
- Mapping
- Disk Access

Internal Layer

- Data Dictionary
- View Definition
- Data Definition
- File Organisation

DB-Operations

Embedding

GUIs

P₁

P₂

...
Classification of Components

- **Definition components**: data definition, file organization, view definition
- **Programming components**: DB programming with embedded DB operations
- **User components**: interactive application programs, query and update
- **Transformation components**: optimizer, analysis, disk access
- **Data dictionary**: contains data from definition components, provides other components with this information
Five Layer Architecture

- **Refinement of transformation steps**

  - **SOI** (Set-Oriented Interface)
    - **Data System**
    - **Translation**, **Access Path Selection**, **Access Control**, **Integrity Control**

  - **ROI** (Record-Oriented Interface)
    - **Access System**
    - **Data Dictionary**, **Currency Pointer**, **Sorting**, **Concurrency control**

  - **IRI** (Internal-Record Interface)
    - **Storage System**

  - **SBI** (System Buffer Interface)
    - **Buffer Manager**
    - **System Buffer Management**, **Page Replacement**

  - **FI** (File Interface)
    - **Operating System**
    - **External Storage Management**

  - **DI** (Device Interface)
    - **External Storage**
Application Architectures

- Architecture of database applications typically based on client-server model: server = database system

1. Request
2. Processing
3. Answer
Application Architectures /2

- Separation of functionality of an application
  - Presentation and user interaction
  - Application logic ("business logic")
  - Data management functionality (store, query, ...).

Two Layer Architecture

Three Layer Architecture
Some concrete systems

- (Object-)relational DBMS
  - Oracle11g, IBM DB2 V.10, Microsoft SQL Server 2012, SAP HANA
  - MySQL (www.mysql.org), PostgreSQL (www.postgresql.org)

- Pseudo DBMS
  - MS Access

- NoSQL systems
  - Graph database systems (InfiniteGraph, neo4j), document databases (MongoDB), key-value stores, ....
Areas of Application

Classical areas of application:
- Many objects (15,000 books, 300 users, 100 books borrowed per week, . . .)
- Few object types (BOOK, USER, BORROWING)
- For instance, book keeping systems, order tracking systems, library systems, . . .

Current applications:
- E-Commerce, decision supporting systems (data warehouses, OLAP), NASA’s Earth Observation System (petabyte databases), data mining
What are Databases?

Areas of Application

Database Sizes

**eBay Data Warehouse**
10 PB ($\approx 10 \cdot 10^{15}$ bytes)
Teradata DBMS, 72 nodes, 10,000 users,
millions of queries/day

**WalMart Data Warehouse**
2,5 PB
Teradata DBMS, NCR MPP hardware;
product information (sales etc.) of 2,900 stores;
50,000 queries/week

**Facebook**
400 TB
x.000 MySQL server
Hadoop/Hive, 610 nodes, 15 TB/day

**US Library of Congress**
10-20 TB
not digitized

PB for Petabyte is in the order of $10^{15}$
Historical Developments: 60s

- Start of 60s: elementary files, application specific file structure (device-dependent, redundant, inconsistent)
- End of 60s: file management systems (SAM, ISAM) with service programs (sorting) (device-independent, but redundant and inconsistent)
- DBS based on **hierarchical model, network model**
  - Pointer structures between data
  - Weak separation of internal / conceptual layer
  - Navigational DML
  - Separation DML / programming language
Historical Developments: 70s and 80s

- 70s: database systems (device and data independence, redundancy free, consistent)
- Relational database systems
  - Data in table structures
  - 3 layer concept
  - Declarative DML
  - Separation DML / programming language
History of RDBMS

- 1970: Ted Codd (IBM) → relational model as conceptual basis of relational DBS
- 1974: System R (IBM) → first prototype of an RDBMS
  - Two modules: RDS, RSS; ca. 80,000 LOC (PL/1, PL/S, assembler), ca. 1.2 MB of code
  - Query language SEQUEL
  - First deployment 1977
- 1975: University of California at Berkeley (UCB) → Ingres
  - Query language QUEL
  - Predecessor of Postgres, Sybase, ...
- 1979: Oracle Version 2
Historical Developments: (80s and) 90s

- **Knowledge base systems**
  - Data in table structures
  - Strongly declarative DML, integrated database programming language

- **Object-oriented database systems**
  - Data in more complex object structures (separation of object and its data)
  - Declarative or navigational DML
  - Often integrated database programming language
  - Often incomplete separation of layers
Historical Developments: Today

- New hardware architectures
  - Multicore processors, terrabytes of main memory: in-memory database systems (e.g., SAP HANA)
- Support for specific applications
  - Cloud databases: Hosting of databases, scalable data management solutions (Amazon RDS, Microsoft Azure)
  - Data stream processing: online processing of live data, e.g., stock exchange data, sensor data, RFID data, ... (StreamBase, MS StreamInsight, IBM Infosphere Streams)
  - Big Data: Handle petabytes of data through highly scalable, parallel processing, data analysis (Hadoop, Hive, Google Spanner & F1, ...)
  - NoSQL databases ("Not only SQL"): non-relational databases, flexible schema (document-centered), "light-weight" because SQL functionality like transactions is omitted, powerful declarative query languages with joins, etc. (CouchDB, MongoDB, Cassandra, ...)
Historical Developments: NoSQL

Historical Developments: NoSQL

Historical Developments: NoSQL


**HOW TO WRITE A CV**

DO YOU HAVE ANY EXPERTISE IN SQL?

NO

DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"

Leverage the NoSQL boom
Trends

- User-generated content, e.g., Google:
  - Daily processing of 20 PB
  - 15 hours of video uploaded to YouTube every minute
  - Reading 20 PB would take 12 years with a 50 MB/s hard disk drive

- Linked data and data web
  - Provision, exchange and linking of structured data on the Web
  - Enables querying (with query languages like SPARQL) and further processing
  - Examples: DBpedia, GeoNames
Summary

- Motivation for using database systems
- Codd’s rules
- 3 layer schema architecture & data independence
- Areas of application
Control Questions

- What is the advantage of using database systems compared to application-specific data management?
Control Questions

- What is the advantage of using database systems compared to application-specific data management?
- What does data independence mean and how is it achieved?
Control Questions

- What is the advantage of using database systems compared to application-specific data management?
- What does data independence mean and how is it achieved?
- Which are areas of application of database systems?