Chapter 1: Databases

Content:

- Learn what a database system is and why to use it

Terms

- What is a database system (DBS)?

System to store and manage data

- Why not use a traditional file system?

Reliability and scalability only achievable with high effort

- Database vs database system?

The DBS is a program that manages the DB (= the data)

Examples

Traditional application areas:

- business data
- accounting
- administration

...

Nowadays a lot broader:

- scientific / medical data
- data mining + machine learning
- geographical information systems
- web search

...

Examples (cont.)

Databases are the back of many applications:

- web search with Google, Yahoo, ...
- inquiries to Amazon, EBay, ...
- posts in Facebook, Twitter, …

Many varieties (DBS/Information Retrieval, centralized/decentralized, replicated, etc.)

Databases are used whenever

- data is very precious (→ reliability)
- amount of data is very big (→ scalability)

Examples (cont.)

The big commercial database systems:

- Oracle
- IBM DB2
- Microsoft SQL Server

Some open source database systems:

- PostgreSQL
- MySQL
- SQLite

Many more, some very specialized (XML, object oriented, data streams, …)

Why use a database system?
Banking Example: Transfer Money in C++

```
void ModifyAccount(const char* account, int amount) {
    char buffer[1024];  memset(buffer, 0, 1024);

    int fd = open(account, O_RDWR);
    read(fd, buffer, 1024);

    int old_balance = atoi(buffer);
    sprintf(buffer, "%i\n", old_balance + amount);

    lseek(fd, 0, SEEK_SET);
    write(fd, buffer, 1024);
    close(fd);
}

void Transfer(const char* from, const char* to, int amount) {
    ModifyAccount(to, amount);
    ModifyAccount(from, -amount);
}
```

Why use a database system?

Banking Example: Transfer Money in SQL

```
begin;

update accounts
    set balance = balance + 80
    where name = ‘Jack';

update accounts
    set balance = balance – 80
    where name = Sam';

commit;
```

Why use a database system?

- Avoid redundancy and inconsistency
- Avoid integrity violations
- Rich (declarative) access to the data
- Security and privacy issues
- Synchronize concurrent data access
- Avoid loss of data (recovery)
- Efficiency and scalability
- Cost/effort → Concentrate on your business logic

Properties of DBS (1)

Data redundancy and consistency

- Data that is stored more than once may diverge over time
- Example: Updating the customer name/address when it is stored on each bill

→ DBS usually avoid redundancies, otherwise rules for updates can be defined to enforce consistency

Properties of DBS (2)

Data integrity

- Data processing has constraints
- Example: Account balance must be positive

→ DBS allows to define rules and thus protects from: User/Programming errors

Students

Jack	TUM
Sam	TUM
Daniel	LMU

Universities

TUM	Arcisstraße 21
LMU	Geschwister-Scholl-Platz 1

Properties of DBS (3)

Declarative query language

- User determines ***which*** data should be retrieved and ***not how***
- Example: C++ vs SQL code (from before)

→ Less error-prone (developing applications)

→ No knowledge about the interior layers of the DBS necessary

→ Usually better performance

Properties of DBS (4)

Sophisticated access rights

- Every user can get different rights on the database
- Example: Name, room, and lectures of a professor should be public; salary not

→ DBS provides a variety of access control mechanisms to enable security and privacy

Properties of DBS (5)

Multi user concurrency

- If you allow several users at a time to update the data without any control you run into big problems

→ DBS allows concurrent access and avoids side effects

Properties of DBS (6)

Error handling

- DBS can restore its state consistently in case of a system failure
- Example: Database crashes during a transaction, changes need to be rolled back

→ Therefore log files are held and managed by the DBS

Properties of DBS (7)

Efficiency and scalability

- DBSs are designed for efficiently handling very large data volumes and a very high number of users

→ In DBSs techniques for scaling with ever higher data volumes are integrated

typically: 100 GB (Gigabyte) – transactional Data (even express versions) up to EB (Exabyte) maximum data size

Properties of DBS (8)

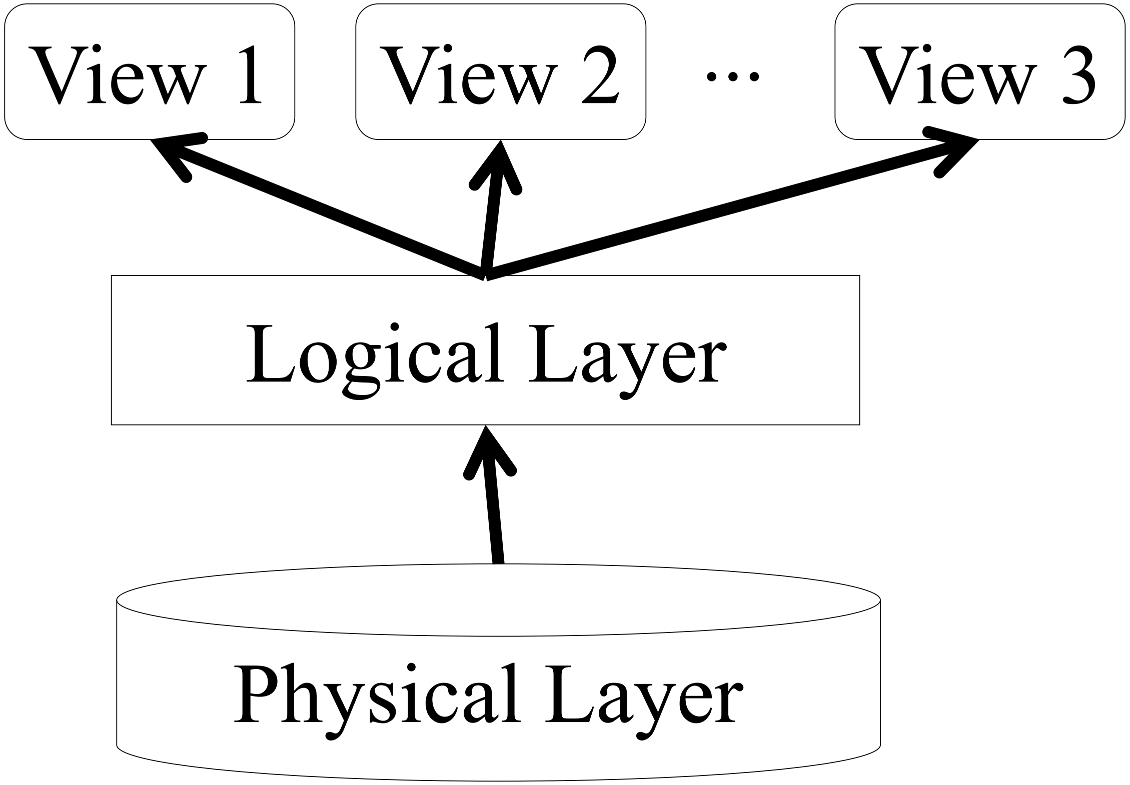
Development Cost

- Reinvent the wheel: developing a custom system for data management has to tackle many of the outlined problems
- Only feasible for large companies for specific problems

Properties of DBSs (résumé)

1. Data redundancy and consistency
2. Data integrity
3. Declarative query language
4. Access rights
5. Concurrency control
6. No data loss (recovery)
7. Efficiency and scalability
8. Cost

Abstract layers of a database system



Abstract layers of a database system (cont.)

View:
-> describes how a specific user/program sees the data

Logical layer:
-> describes how the data is structured

Physical layer:
-> describes how the data is stored

Abstract layers of a database system (cont.)

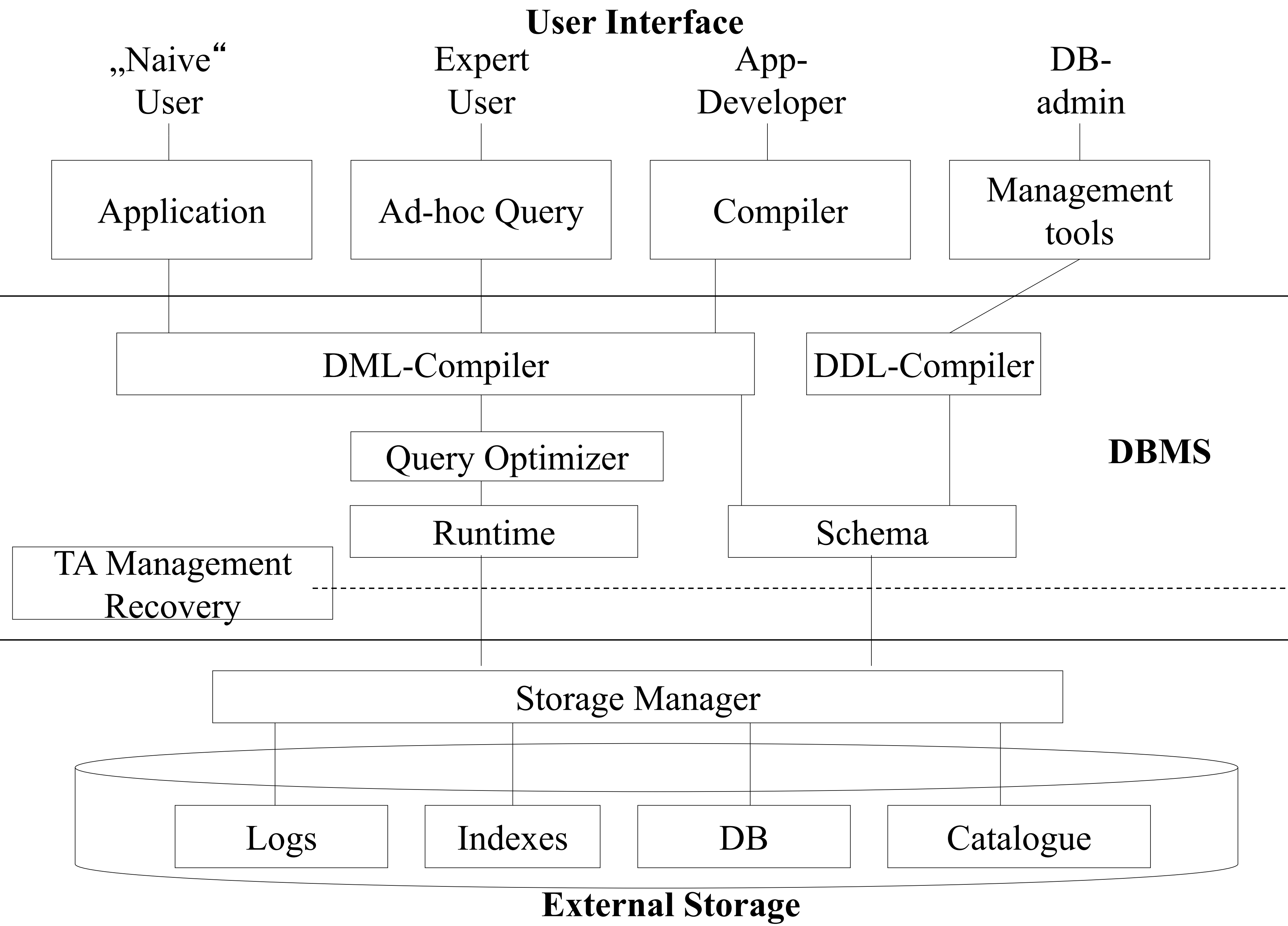
DBS decouples applications from the structure and storage of the data:

- **Logical data independency**
  (simple) changes at the logical layer have no influence on the applications
- **Physical data independency**
  changes at the physical layer have no influence on the applications

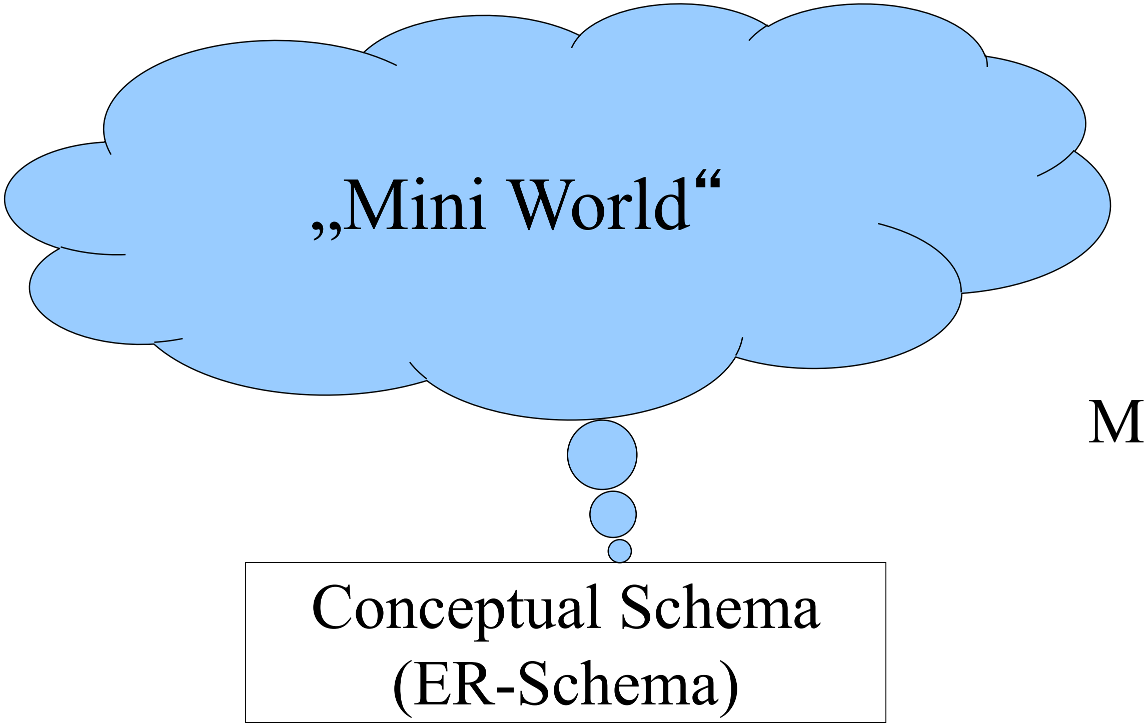
Implemented in almost all modern database systems

Architecture & Components of a Database System

- Layered architecture
  - User Interface
  - DBMS
  - External Storage


User Interface
„Naive“ User
Expert User
App-Developer
DB-admin
Application
Ad-hoc Query
Compiler
Management tools
DML-Compiler
DDL-Compiler
Query Optimizer
DBMS
Runtime
Schema
TA Management Recovery
Storage Manager
Logs
Indexes
DB
Catalogue
External Storage

Next: Data Modeling



„Mini World“

Conceptual Schema
(ER-Schema)

Manual Modeling