Databases

Lecture 1 - Organisation and Introduction to Databases

Emily Lucia Antosch

HAW Hamburg

12.01.2025

Contents

1. Organisation

1.1 Where are we right now?

- This is the first chapter of “Databases” in this semester. Welcome!
- Today, we’ll be discussing
  - the way this lecture is going to work,
  - what we are going to learn during this semester,
  - and what databases are and why you learning about them!

1.1 Where are we right now?

1. **Introduction**
2. Basics
3. SQL
4. Entity-Relationship-Model
5. Relationships
6. Constraints
7. More SQL
8. Subqueries & Views
9. Transactions
10. Database Applications
11. Integrity, Trigger & Security

1.2 What is the goal of this chapter?

- I want to introduce myself to you and maybe also learn a bit about you!
- I want to tell you about what is in store for you this semester

1.3 About me

- Emily Lucia Antosch, 24
- I did my bachelor's degree in Electrical Engineering
- Software Engineer in the marine industry
- Looking to start my master degree in the near future
- Mail: emilylucia.antosch@haw-hamburg.de

1.4 How is this lecture going to work?

- I would like to kindly ask you to participate in the lectures ahead, it'll make the whole thing more fun.
- All parts of this course will split into lectures and labs.

> **! Memorize**
>
> I would ask you to please let me know if you find that you were not able to follow the lecture. I'm more than happy to repeat certain parts.

1.5 Moodle

- Lecture material can be found at: moodle.haw-hamburg.de
- Enrollment key: db_2025
- The slides are also the script.
- **Thus your notes are essential!**

1.6 Labs

- Assignments should be worked on in teams
  - Work in fixed teams of two. Division of the teams of two in lab on XXX
  - Working together means discussing things, explaining each other, helping each other out
  - Every team member must be ability to explain the solution of each assignment
- For PVL (precondition for examination):
  - Presence in all laboratories is obligatory!

  In case of illness: Send a sick note and make up the Lab on another date
  - All assignments for the labs must be successfully solved
  - Each student must present at least two assignments on blackboard

1.6 Labs

- Assignments are published before laboratory
- Each laboratory consists of two parts:

1. Upfront assignments
   - Submitting the solutions Friday before lab date e.g., solution of lab on 06.05.2024 must be submitted until 03.05.2024 11:59 p.m. via moodle
     - Only one submission per team of two
     - No re-submission after a laboratory
     - Submit only PDF-files
2. Live assignments
   - They can be solved in advance or during the Lab
   - Discussion during the laboratory

1.6 Labs

- For each laboratory date
  - Punctual participation
  - Each team member must be able to explain the solution to all upfront assignments
- You'll receive a yellow card for your first violation of the rules.
- **In case of a second infringement: Exclusion from exercise!**
- **Participation of all laboratory dates is mandatory, unexcused absence leads to immediate exclusion from the laboratory**

1.7 First Lab

- Joint lab with all three lab groups
- Attendance is mandatory!
- Division of the teams of two to work on the lab assignments
- Bring your own device **with a working PostgreSQL database**!
- You do not have to submit solutions in moodle or implement the assignments in advance
- Start time: 8:30 am
- I strongly recommend that you look at and solve the assignments in advance! This will make the laboratory much more effective for you.

1.8 Important people in this lecture

- Emily Antosch: Lecture emilylucia.antosch@haw-hamburg.de
- Julian Moldenhauer: Lab group 01 or group 2 julian.moldenhauer@haw-hamburg.de
- Furkan Yildirim: Lab group 01 or group 2 furkan.yildirim@haw-hamburg.de
- Ulrike Herster: Lecture and Lab group 03 Ulrike.Herster@haw-hamburg.de

1.9 Focus of these lectures

- At the end of this semester, you'll be able to
  - create database systems to effectively store data.
  - design complex databases solutions using Entity-Relationship-Models.
  - secure your database with advanced and modern techniques.

1.10 Requirements

Memorize

You will need an installation of **PostgreSQL 16**.

- It's open source software and the main database system that will look at.
- Depending on your system and how you want to install PostgreSQL, there are multiple ways to go about it. There are detailed descriptions in the Moodle-Room for you to follow.
- It's totally up to you, but I would suggest you also download pgAdmin4, becauses it allows you to use an UI to interact with your database. pgAdmin4 is also free.

1.11 Who are you?

? Question

- Do you already have experience regarding databases?
- What is the expectation of this lecture?
- Do you have any wishes regarding the lecture and potentially also the exam?

2. Introduction to Databases

2.1 Where are we right now?

- You just learnt how this lecture is going to work and what you can expect going forward.
  - Next, we'll be discussing the basics of databases

2.1 Where are we right now?

1. Introduction
2. **Basics**
3. SQL
4. Entity-Relationship-Model
5. Relationships
6. Constraints
7. More SQL
8. Subqueries & Views
9. Transactions
10. Database Applications
11. Integrity, Trigger & Security

2.2 What is the goal of this chapter?

- Today, we'll be discussing
  - what databases are and why you learning about them,
  - the history of databases,
  - and the differences between databases and database management systems.

2.3 What is a Database?

Memorize

- A **database** refers to a set of data and how it it organized.
- Access is granted via a **database management system** (DBMS) consisting of integrated software that allows for interaction with one or databases and provides access to the data.
- Supports storage, manipulation, and querying of information.

2.4 Database Management System (DBMS)

- Software system that manages databases.
- A DBMS provides a systematic approach of creating, updating, storing and retrieving data stored in a database.
- It enables the end user and programmers to share data, and it allows for data to be shared among multiple applications.
- It eliminates the need for data to be stored in new files and being propagated.

2.4 Database Management System (DBMS)

The essential functions of a DBMS:

- Storing, changing, and deleting data
- Managing metadata (data about the data)
- Keeping your data safe and secure
- Making sure your data is correct and consistent
- Allowing multiple users to work with the data at the same time (transactions)
- Optimizing queries (finding the fastest way to get the data you need)
- Enabling triggers (automatic actions when certain events happen) and stored procedures (pre-written SQL code)
- Providing key metrics about the DBMS technology and how it's running

2.5 Database Examples

- Customer Relationship Management (keeping track of your customers)
- Controlling and Accounting (managing your finances)
- Merchandise Management System (organizing your products)
- Enterprise Resource Planning (managing your entire business)
- Content Management Systems (managing your website content)

2.6 History of the Database

- In the 1960s, people used files to store data. This wasn't ideal because files are designed for specific applications, and it was a lot of work to manage them.
- In the 1970s, Edgar F. Codd, who worked at IBM, came up with the idea of relational databases.
- He developed the first relational database system called "System R."
- Oracle took Codd's ideas and made SQL (Structured Query Language) a big success.
- IBM followed with their own SQL databases (SQL/DS and DB2).
- Today, relational databases are the most common type of database.

2.7 Why even use a DB?

- What are the alternatives for storing data?
- Text files, MS Excel, MS Access, etc.
- What are the disadvantages of these alternatives?

2.7 Why even use a DB?

Disadvantages of alternatives like text files, Excel, and Access:

- **Data organization** Can be tricky to structure your data properly.
- **Data types:** Limited options for different kinds of data.
- **Large datasets:** Can't handle huge amounts of data efficiently.
- **Data validation:** Hard to make sure the data is accurate.
- **Security:** Not very secure.
- **Performance & querying:** Can be slow to search and get the data you need.
- **Backup & maintenance:** Can be difficult to back up and maintain your data.
- **Sharing:** Can be hard to share the data with others.
- **Performance with large datasets:** Access can struggle with thousands of entries.

2.7 Why even use a DB?

- **Concurrency & control features:** Limited ability for multiple users to work with the data at the same time.

2.8 Database vs. Spreadsheet

- It's easy to accidentally change data in spreadsheets.
- It's hard to repeat old analyses on new data in spreadsheets.
- Spreadsheets are slow with large datasets.
- It's difficult to share huge spreadsheets.

2.9 SQL Database vs. MS Excel: What are they best used for?

Databases are good for:

- Larger datasets (databases can handle a lot more data than Excel)
- Organization/structure (databases are stricter about how data is organized)
- Collaborative work (databases are better for teams working together)
- Preparing data for analysis in other software

Excel is good for:

- Smaller datasets (Excel can slow down with large datasets)
- Manually entering data
- Flexible structure (Excel is more forgiving about how data is organized)
- Creating graphs and visualizations
- Consistent reports or calculations

2.9 SQL Database vs. MS Excel: What are they best used for?

- Built-in spell check and other helpful tools
- Working independently

2.10 Database vs. MS Access

|Aspect|DB|MS Excel|MS Access|Note| |:—|:—|:—|:—|:—| |Initial training||++|+|Initial training is necessary. E.g., separation of the presentation and editing of data from processing and storage| |Large data sets|++||+|Access has performance problems with several thousand entries| |Access by multiple users|++|+++|Multiple users can access the database simultaneously| |Database Design|++|-|+|The direct display of tables leads to denormalized tables| |Platform independence|++|-|-|Access limited to Windows| |Application development|-|++|++|A SQL database can never stand alone| |Integration with MS Office||++|++||

2.11 Database vs. MS Access: Technical Comparison

|Aspect|mySQL|MS Access| |:—|:—|:—| |Database Size|16 terabytes|2 GB| |Simultaneous users| 32.767 users|255 users| |Number of objects|2.147483.647 objects per database|32.768 objects per database|

2.12 Different DB-Models

- Relational model
- Hierachical model
- Network model
- Object relational model
- Object oriented model
- XML-based model

2.13 RDBMS vs. ODBMS

- RDBMS (Relational Database Management System) and ODBMS (Object-Oriented Database Management System) are two ways to manage data.
- RDBMS stores data in tables with rows and columns, kind of like a spreadsheet.
- ODBMS stores data as objects, which can be more complex and have their own properties and methods.

2.14 Relational Database Market 2020

In 2020, the majority of the database market was dominated by SQL databases (Relational Databases). NoSQL databases held a smaller portion of the market share.

Source: https://www.industryarc.com/Report/19213/relational-database-market.html

2.15 Most Popular Databases 2024

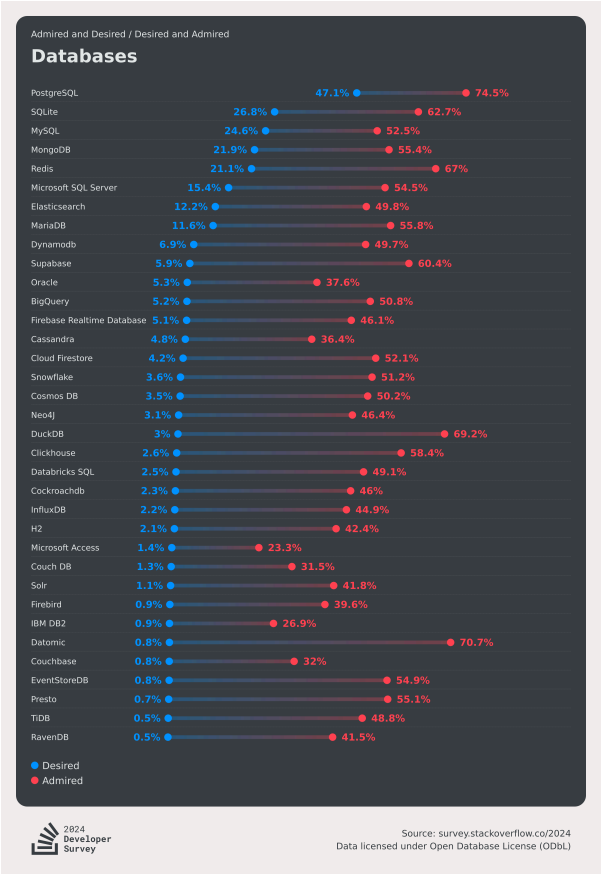

Figure 1: Most admired databases (Source: StackOverflow Developer Survey 2024)

2.15 Most Popular Databases 2024

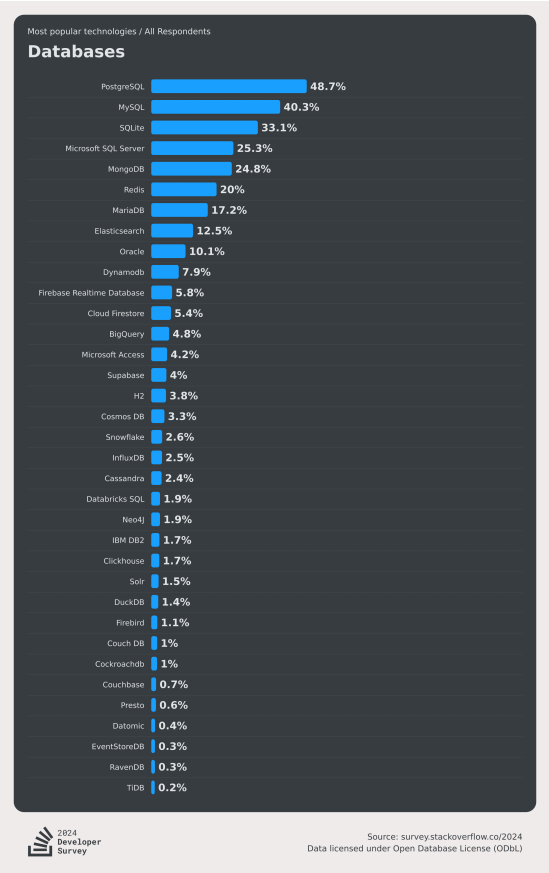

Figure 2: Most popular technologies (Source: StackOverflow Developer Survey 2024)

2.16 Database Design

To design a database, you typically follow these steps:

1. **Requirements:** Figure out what you need the database to do.
2. **Conceptual Database Design:** Come up with a high-level plan for your database using an ER Model (Entity-Relationship Model). This is like a rough sketch of your database.
3. **Logical Database Design:** Refine your plan and choose a specific type of database (like a relational database). You'll also use a more formal model here, like the Relational Model.
4. **Physical Database Design:** Get into the technical details of how the data will be stored and organized.
5. **Database Implementation:** Build the actual database using SQL (Structured Query Language).

2.16 Database Design

- **Conceptual Design:** A high-level plan for the database. This is where you use the ER Model to map out the entities (things) and their relationships.
- **Logical Design:** A more detailed, formal plan. Here, you use the Relational Model to structure the database into tables and relationships.
- **Implementation & Usage:** Building and using the database. This is where you use SQL to create the database and work with the data.

2.17 Example: Contact List

- **What things exist in the real world?** (e.g., people, houses)
- **What properties do they have?** (e.g., names, addresses, phone numbers)
- **How do they relate to each other?** (e.g., people live in houses)

2.17 Example: Contact List

Option 1: Conceptual design with a Class Diagram

- You can use a class diagram to model your database conceptually.
- This involves defining classes (like blueprints) for the things in your database (e.g., a Person class, a House class).
- Each class has properties (attributes) to describe those things (e.g., a Person has a name, a House has an address).
- You also define relationships between the classes (e.g., a Person “lives in” a House).

2.17 Example: Contact List

Option 2: Conceptual design with ER Model

- You can also use an ER Model (Entity-Relationship Model) for the conceptual design.
- This is a more visual way to model your database, where you use boxes to represent entities (things) and diamonds to represent relationships between them.
- Each entity has attributes (properties) that describe it.
- You also indicate how many entities can be related to each other (cardinalities).

2.17 Example: Contact List

1. Conceptual design: Class diagram vs. ERM

- You can easily translate a class diagram into an ERM.
- There are a few differences between ERMs and Class Diagrams:
  - ERMs don't have methods (actions).
  - ERMs allow for multivalued attributes (attributes that can have multiple values).

3. SQL: Structured Query Language

3.1 What is SQL?

- Standard language for managing relational databases
- Used for querying, updating, and managing data

3.2 Basic SQL Commands

- SELECT: Retrieve data
- INSERT: Add new records
- UPDATE: Modify existing records
- DELETE: Remove records

3.3 SQL Example: SELECT Statement

```sql
SELECT Name, Major
FROM Students
WHERE GPA > 3.5;
```

3.4 SQL Example: INSERT Statement

```sql
INSERT INTO Students (Student_ID, Name, Major, GPA)
VALUES (104, 'David', 'Biology', 3.7);
```

4. Database Design

4.1 Steps in Database Design

- Requirements Analysis
- Conceptual Design
- Logical Design
- Physical Design
- Implementation and Testing

4.2 Entity-Relationship (ER) Model

- Conceptual representation of data
- Entities: Objects or concepts
- Attributes: Properties of entities
- Relationships: Connections between entities

ER Diagram Example [Diagram placeholder: Simple ER diagram showing entities like Student, Course, and Instructor with relationships]

5. Normalization

5.1 What is Normalization?

- Process of organizing data to minimize redundancy
- Improves data integrity and reduces anomalies

5.2 Normal Forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)

5.3 Benefits of Normalization

- Eliminates data redundancy
- Ensures data consistency
- Simplifies data maintenance
- Facilitates database design

6. Transaction Management

6.1 What is a Transaction?

- A sequence of database operations
- Treated as a single unit of work

6.2 ACID Properties

- Atomicity: All-or-nothing execution
- Consistency: Database remains in a valid state
- Isolation: Transactions execute independently
- Durability: Committed changes are permanent

6.3 Transaction States

Active Partially Committed Committed Failed Aborted

7. Conclusion

7.1 Key Takeaways

- Databases are essential for efficient data management
- Relational model is widely used in database systems
- SQL is the standard language for database operations
- Proper database design is crucial for performance and integrity
- Understanding transactions ensures data consistency and reliability

7.2 Next Steps

Explore advanced SQL queries Study database indexing and optimization Learn about database security and access control Practice designing and implementing databases

8. License Notice

8.1 Attribution

- This work is based off of the work Prof. Dr. Marc Hensel.
- Some of the images and texts, as well as the layout were changed.
- The base material was supplied in private, therefore the link to the source cannot be shared with the audience.