Databases

Lecture 1 - Organisation and Introduction to Databases

Emily Lucia Antosch

HAW Hamburg

03.04.2025

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.
- We'll also discuss the structure of the course, including assignments and exams.
- The importance of participation and collaboration in class.

1.3 About me

- Emily Lucia Antosch
- I did my bachelor's degree in Electrical Engineering, also at the HAW.
- 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. (More on labs later).

> **! 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 Use of AI in this lecture

Opportunities

- **Breaking Down Complex Concepts**
  - Simplify intricate ideas into easy-to-understand language
  - Provide step-by-step explanations for difficult topics
  - Offer analogies and real-world examples to make abstract concepts concrete

- **Providing Practical Examples**
  - Generate code snippets and practical implementations
  - Create sample projects and workflows
  - Demonstrate use cases across different industries

- **Offering Interactive Learning Experiences**
  - Engage in conversational practice to reinforce learning
  - Ask questions and receive immediate feedback
  - Solve problems collaboratively with the AI

1.5 Use of AI in this lecture

Challenges

- **Hallucinations and Inaccuracy**: AI models can sometimes generate incorrect or misleading information (often called “hallucinations”), requiring careful verification of the provided answers.
  - This is especially problematic when learning something new where you may not have the expertise to identify errors.
- **Lack of Contextual Understanding**: AI may struggle with nuanced concepts or the specific context of your learning goals.
  - It can provide technically correct answers that aren’t relevant to your particular situation or level of understanding.
- **Dependency and Reduced Critical Thinking**: Over-reliance on AI can hinder the development of problem-solving skills and critical thinking.
  - Don’t become a passive recipient of information! Rather,actively engage with the material and the learning process.

1.5 Use of AI in this lecture

- **Difficulty with Complex or Abstract Concepts**: While AI excels at pattern recognition, it can struggle with explaining complex or abstract concepts in a way that is easy to understand, especially without sufficient prompting or clarification.

1.5 Use of AI in this lecture

AI in the exam

Please remember that the use of AI is not permitted during the exam. There is a zero-tolerance-policy regarding cheating. **Please keep in mind that your are the person solving the problems and not the AI!**

1.6 Moodle

Lecture Material

- 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!**

1.7 Labs

- Assignments should be worked on in teams
  - Work in fixed teams of two. You need to enter your group into the Moodle room before the first lab.
  - Working together means discussing things, explaining problems 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.

1.7 Labs

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

1. Upfront assignments
   - Submitting the solutions by Saturday 23:59 before the lab date!
   - Failure to do so will have consequences.
   - Only one submission per team of two
   - No re-submission after a laboratory
   - Submit only PDF-files

Example

Solution of lab on 05.05.2025 must be submitted by the 03.05.2024 23:59 via Moodle.

2. Presentation of solutions to the lecturers and the group.

1.7 Labs

- Each lab requires
  - punctual participation
  - each team member to 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**

1.8 First Lab

- Joint lab with all three lab groups
- Attendance is mandatory!
- Bring your own device **with a working PostgreSQL database**!
- There's an installation guide on it in the Moodle room. If you find yourself being lost despite that, shoot me an email! There's also lots of good instructions online.

1.9 Important people in this lecture

- Emily Antosch: Lecture emilylucia.antosch@haw-hamburg.de
- Julian Moldenhauer: Lab julian.moldenhauer@haw-hamburg.de
- Furkan Yildirim: Lab furkan.yildirim@haw-hamburg.de

1.10 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.

1.11 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 MoodleRoom for you to follow.
- It's totally up to you, but I would suggest you also download pgAdmin4, because it allows you to use an UI to interact with your database. pgAdmin4 is also free.
  - pgAdmin4 also usually comes with the installation of PostgresQL in most cases.

1.12 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 and the differences to database management systems.
- We'll learn about the history of databases.
- And we'll find out, why we should be using databases in the first place.

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.

Definition

- 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 What is a 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 What is a Database Management System (DBMS)?

- **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

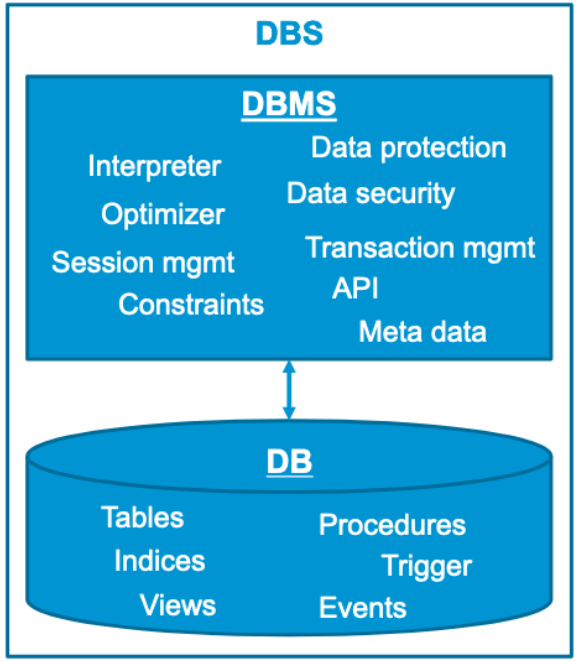
2.5 In essence: DB vs DBMS

In Essence

- A database manages data logically and physically
- A DBMS offers tools for managing, editing and evaluating data

Database Examples

- Customer Relationship Management (CRM) (keeping track of your customers)
- Controlling and Accounting (managing your finances)
- Merchandise Management System (organizing your products)
- Enterprise Resource Planning (ERP) (managing your entire business e.g. SAP)
- Content Management Systems (CMS) (managing your website content e.g. WordPress)



2.6 Difference between data and information

Data

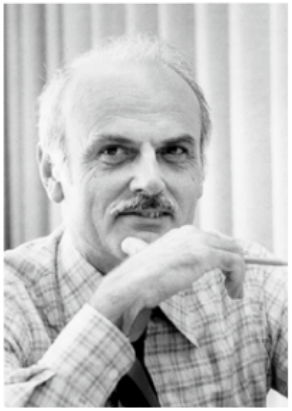
- Data is raw, uncategorised facts such as numbers, text or images.
- More often than not, data does not make sense on its own and requires some form of context.

Information

- Information is born when data is given context, meaning and/or relevance.
- Information is able to actively serve us by providing insight in how decisions should be made.

2.7 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, 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.



Source: www.wikipedia.org

2.8 Why even use a DB?

? Question

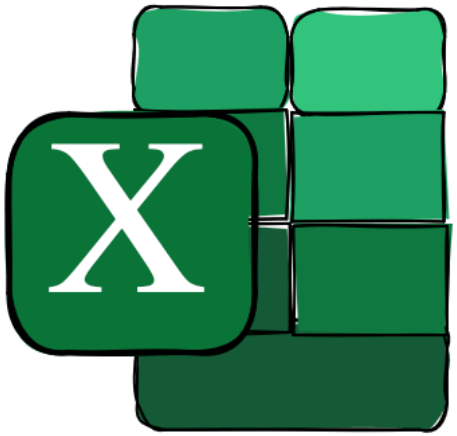
- What are the alternatives for storing data?

2.8 Why even use a DB?

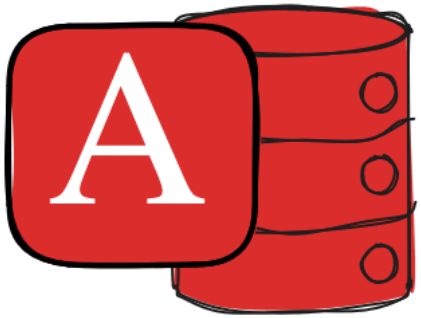
? Question

- What are the alternatives for storing data?

MS Excel



MS Access



Text file

2.8 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.
- **Concurrency & control features:** Limited ability for multiple users to work with the data at the same time.

2.9 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.

2.10 Database vs. 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
- Built-in spell check and other helpful tools
- Working independently

2.11 Database vs. Excel vs. MS Access

Aspect	DB	MS Excel	MS Access	Comment
Initial Training		++	+	Initial Training is necessary, since presentation and editing data is separated.
Large data sets	++		+	Access has performance problems starting from several thousand entries.
Access by multiple Users	++		+	Using a database together is easy and works out-of-the-box.
Database Design	++		+	It's way easier to design a data storage solution by profiting off dedicated features.
Platform Independence	++	+		While DBMS work on any system, MS products are limited to Windows and MacOS.

2.11 Database vs. Excel vs. MS Access

Aspect	DB	MS Excel	MS Access	Comment
Application Development	+	+	+	While you can't really develop applications using SQL only, the other two choices aren't preferable either.
Integration with MS Office		++	++	

2.12 Database vs. MS Access: Technical Comparison

Aspect	DB	MS Access
Database Size	16TB	2GB
Simultaneous users	32.767 users	255 users
Number of objects	2.147483.647 objects per database	32.768 objects per database

2.13 Different DB-Models

- There are a number of different database models available. These include:
  - Relational model
  - Hierarchical model
  - Network model
  - Object relational model
  - Object oriented model
  - XML-based model

2.14 RDBMS vs. ODBMS

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

> **Info**
>
> Databases that implement more than one type (paradigm) of databases are called *multi-paradigm*. An example would be SurrealDB.

2.15 Most Popular Databases 2024

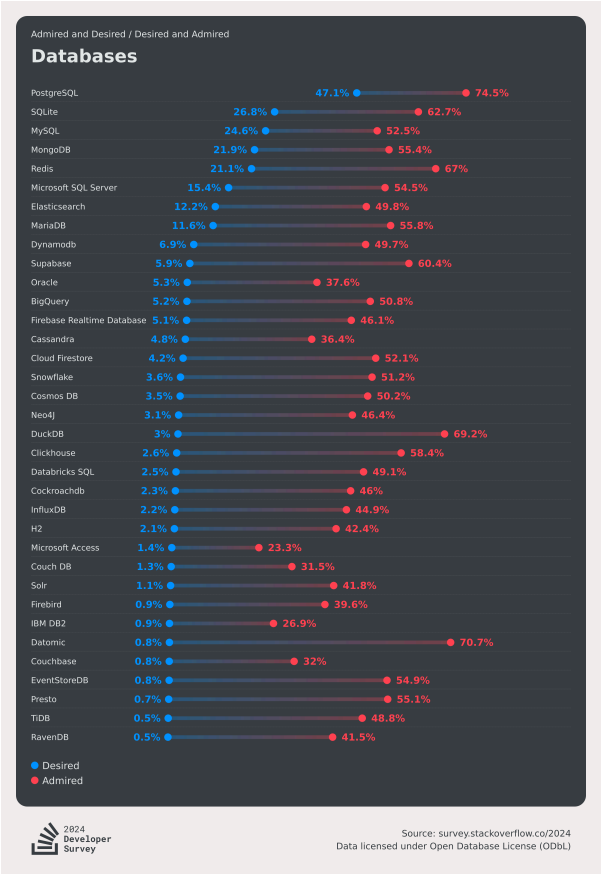


Figure 7: Most admired databases (Source: StackOverflow Developer Survey 2024)

2.15 Most Popular Databases 2024

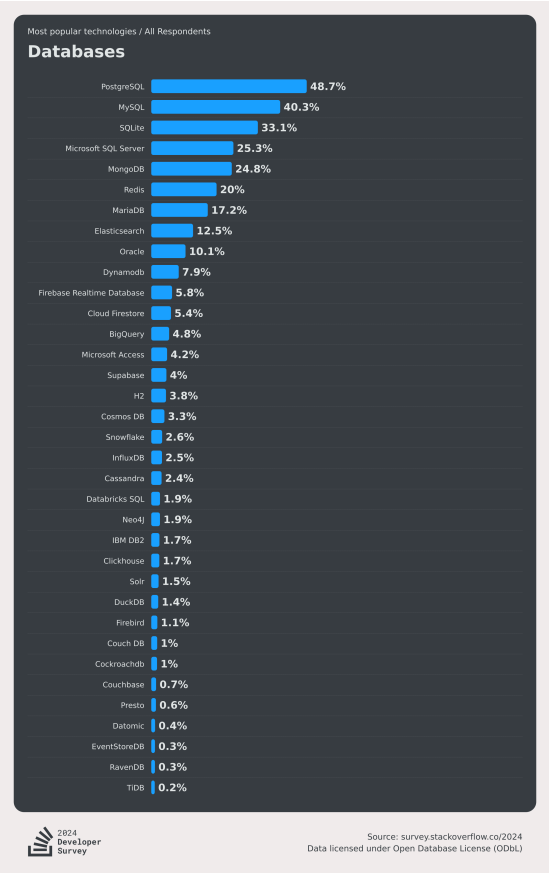


Figure 8: Most popular technologies (Source: StackOverflow Developer Survey 2024)

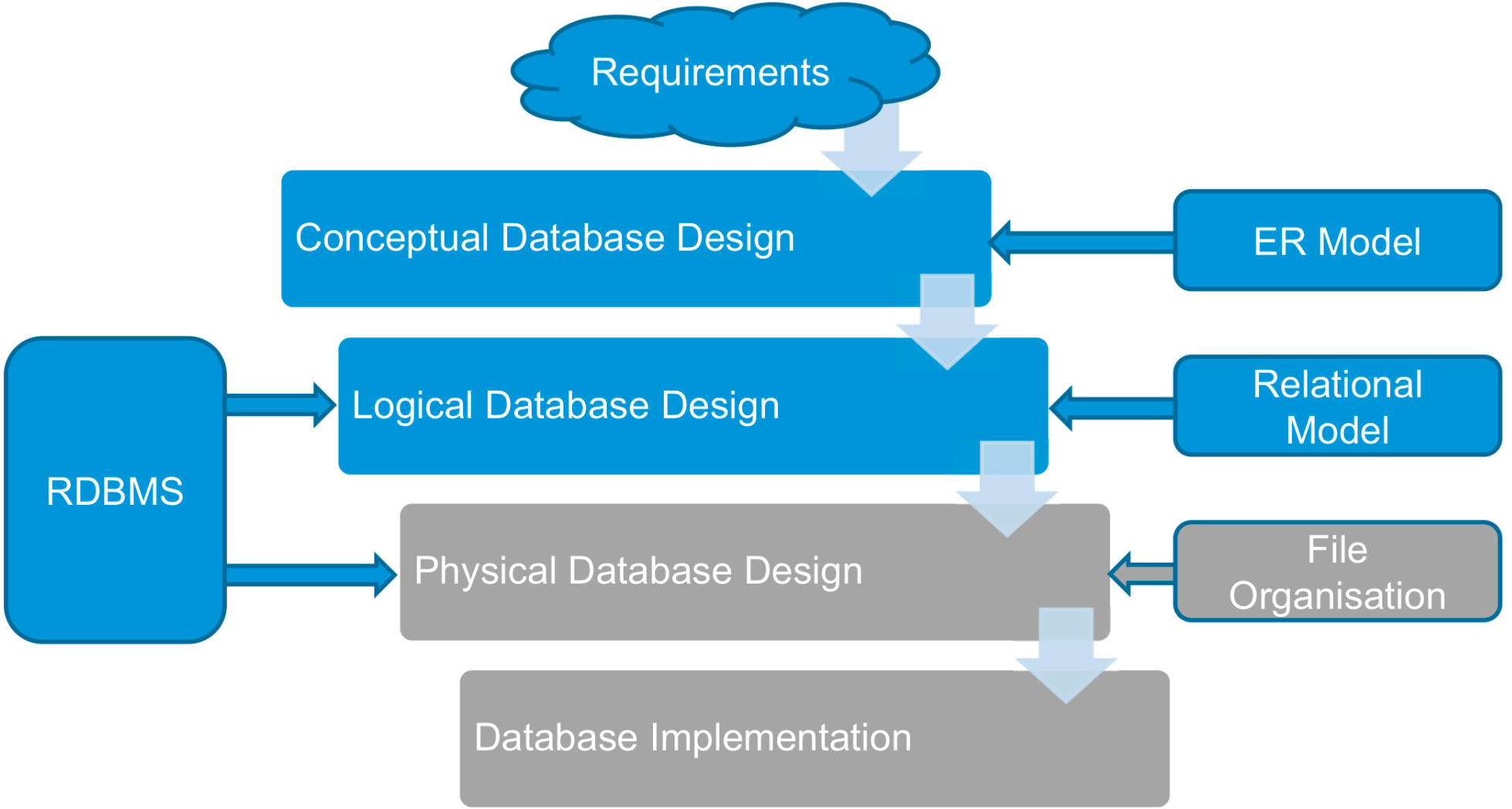
2.16 Why PostgreSQL

- Why PostgreSQL instead of MySQL, SQLite or Oracle?
  - SQLite is easy but limited in its feature set.
  - Oracle is used in many enterprises, but it's not free.
  - MySQL is also a very valid option, but I like PostgresQL more. It seems to me that PostgresQL has become the preferred choice for most developers.

2.17 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).

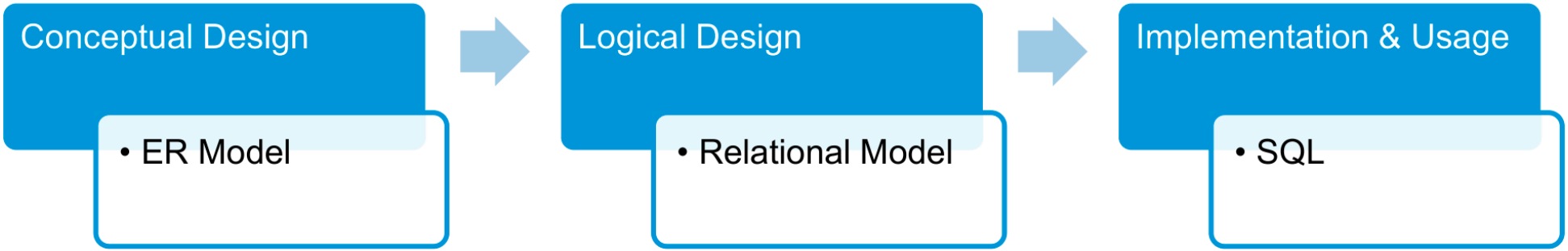
2.17 Database Design



2.17 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.

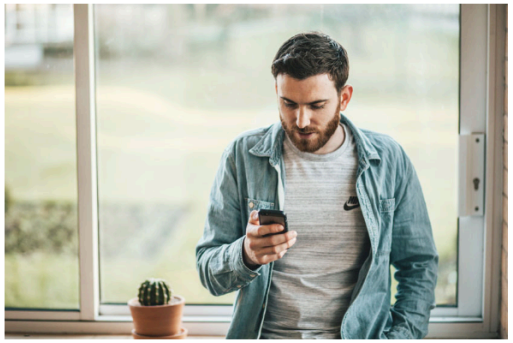
2.17 Database Design



2.18 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)

2.18 Example: Contact List



Source: Photo by Thom Holmes on Unsplash

What things exist in the real world?
What properties do they have?
How do they relate to each other?

Steven
12.05.1968

Maria
8.05.1972

Jane
30.07.1944

John
10.06.1996

Ann
12.04.1998

221 Baker Street, 1NW London
0044 20 7946 0000

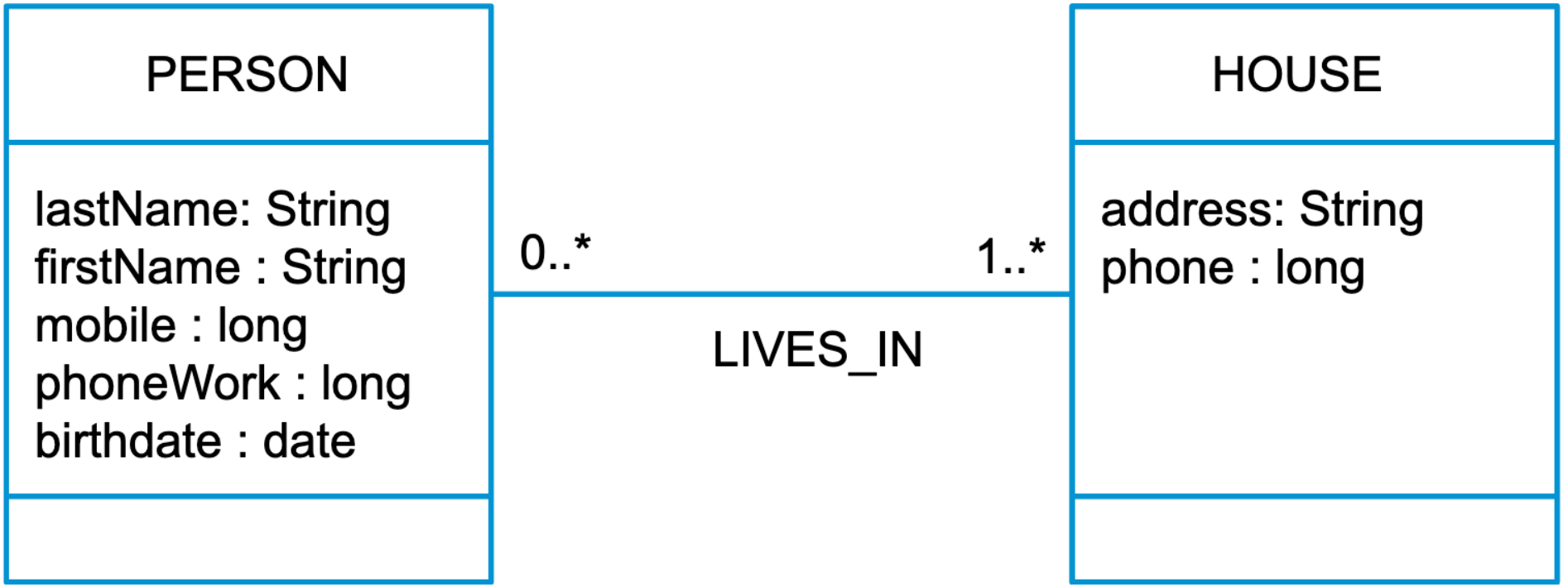
112 Baker Street, 1NW London
0044 20 7946 1000

55 Oxford Street, W1 London
0044 20 7946 3000

2.18 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).

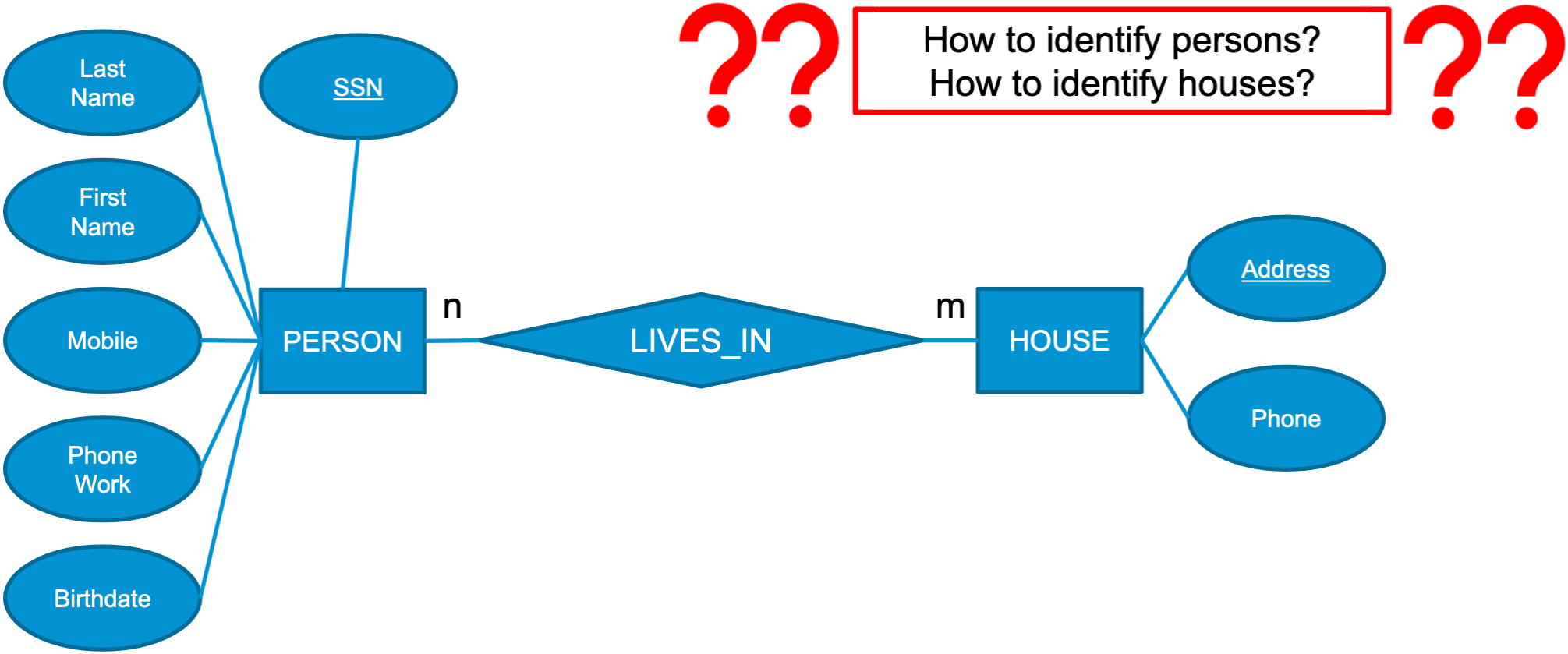
2.18 Example: Contact List



2.18 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).

2.18 Example: Contact List



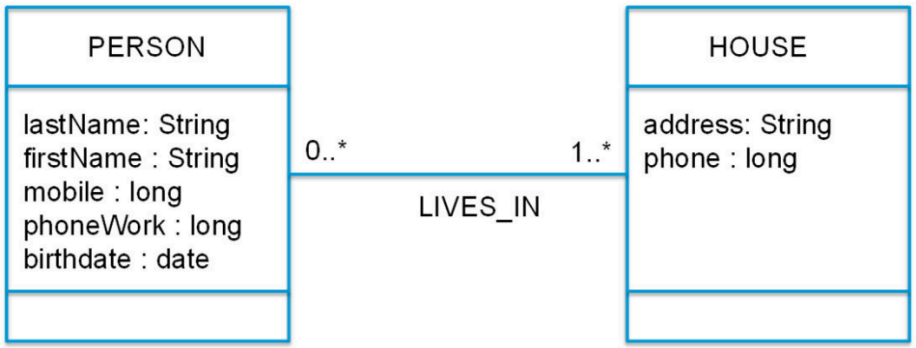
2.18 Example: Contact List

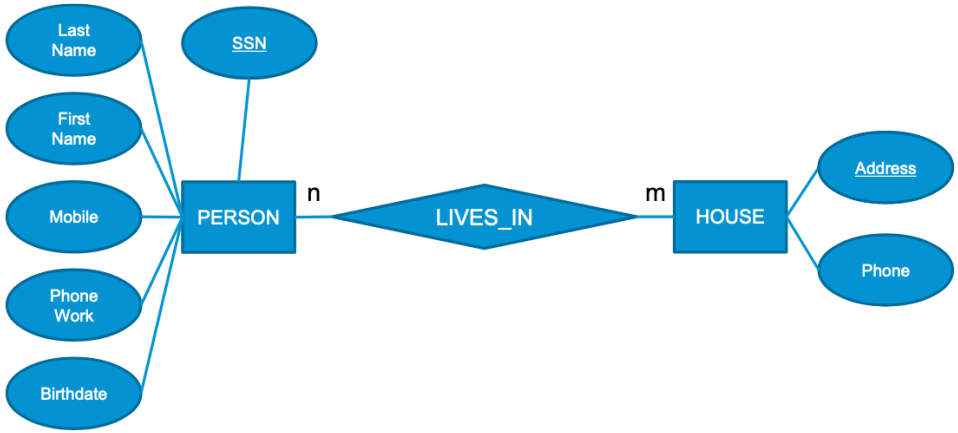
1. **Conceptual design: Class diagram vs. ERM**
   - You can easily translate a class diagram into an ERM.
   - There are a few differences between ERMs and Class Diagrams:
     - ERMs don't have methods (actions).
     - ERMs allow for multivalued attributes (attributes that can have multiple values).

2.18 Example: Contact List

You can easily translate a class diagram into an ERM

Some differences between ERM and Class Diagrams
- There are no methods in an ERM
- ERM offers multivalued attributes
- Identifying keys are marked in ERM
- …





2.18 Example: Contact List

PERSON	SSN	FirstName	LastName	Mobile	PhoneWork	Birthdate

LIVES_IN	SSN (FK)	Address (FK)

HOUSE	Address	Phone

2.18 Example: Contact List

PERSON

SSN	Last Name	First Name	Mobile	Phone work	Birthdate
123456789	Miller	Jane	0044 7701 123456	0044 20 7946 0001	11.04.1979
234567891	Miller	Steven	0044 7701 123457	0044 20 7946 0002	13.05.1977
345678912	Miller	Maria			21.07.2015
…	…	…	…	…	…

HOUSE

Address	Phone
221 Baker Street, 1NW London	0044 20 7946 0000
112 Baker Street, 1NW London	0044 20 7946 1000
55 Oxford Street, W1 London	0044 20 7946 3000
…	…

LIVES_IN

SSN	Address
123456789	221 Baker Street, 1NW London
234567891	221 Baker Street, 1NW London
345678912	221 Baker Street, 1NW London
…	…

2.18 Example: Contact List

- Creating the tables with the correct columns:

```sql
CREATE TABLE Person(
   SSN  CHAR(9) NOT NULL,
   FirstName VARCHAR(15) NOT NULL,
   LastName VARCHAR(15) NOT NULL,
   Mobile VARCHAR(30),
   PhoneWork VARCHAR(30),
   Birthdate DATE,
   PRIMARY KEY ( SSN ));
```

2.18 Example: Contact List

- Selecting the first and last name from the table `Person` where the `LastName` is Miller:

```sql
SELECT
  FirstName,
  LastName
FROM
  Person
WHERE
  LastName = 'Miller';
```

2.18 Example: Contact List

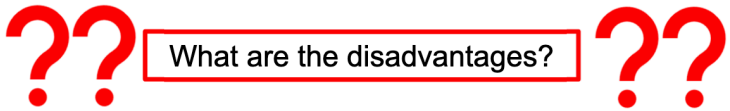
- Selecting the first and last name from the table `Person`, joined with the tables `Lives_In` and `House` with filters for different columns:

```sql
SELECT
    FirstName,
    LastName
FROM
    Person P,
    lives_in L,
    House H
WHERE
    Phone = '0044 20  7946 0000'
    AND H.Address = L.Address
    AND L.SSN = P.SSN;
```

2.18 Example: Contact List

Question

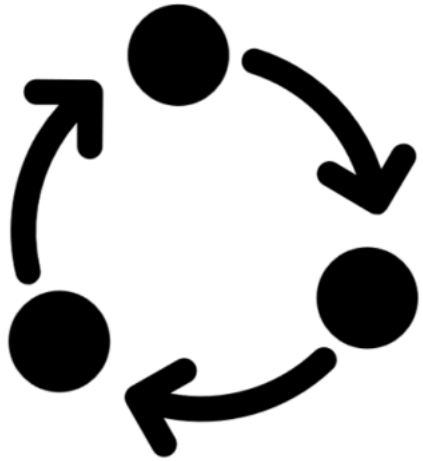
- What are the disadvantages of this approach?



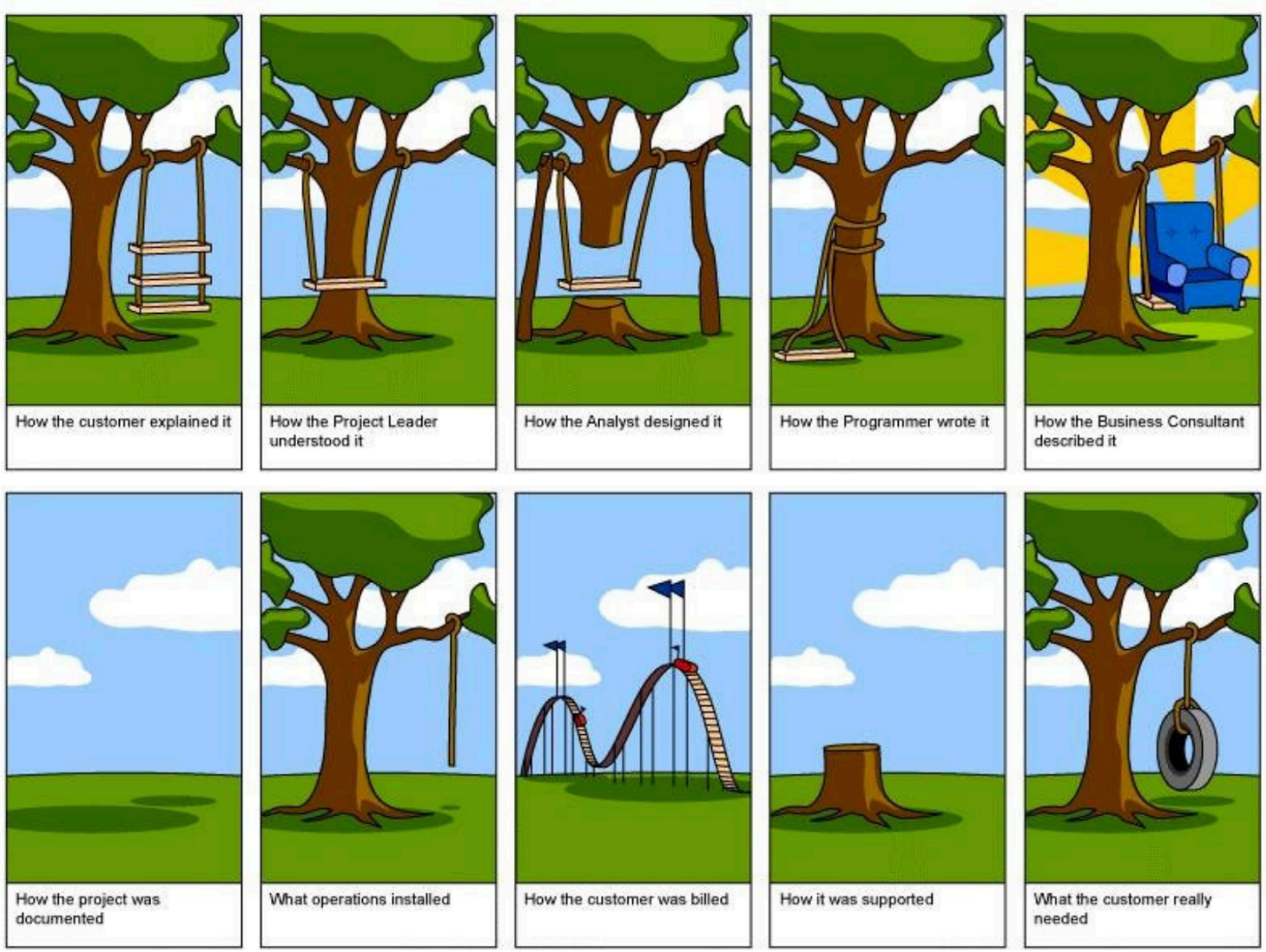
Last Name	First Name	Address	Phone	Mobile	Phone work	Birthdate
Miller	Jane	221 Baker Street, 1NW London	0044 20 7946 0000	0044 7701 123456	0044 20 7946 0001	11.04.1979
Miller	Steven	221 Baker Street, 1NW London	0044 20 7946 0000	0044 7701 123457	0044 20 7946 0002	13.05.1977
Miller	Maria	221 Baker Street, 1NW London	0044 20 7946 0000			21.07.2015
Miller	Josef	221 Baker Street, 1NW London	0044 20 7946 0000	0044 7701 123458		10.11.2010
Miller	Judy	221 Baker Street, 1NW London	0044 20 7946 0000			01.12.2013
Smith	John	112 Baker Street, 1NW London	0044 20 7946 1000	0044 7701 456789		06.06.1987
Smith	Jane	112 Baker Street, 1NW London	0044 20 7946 1000	0044 7701 456780	0044 20 7946 1003	05.03.1989
Smith	Hannah	112 Baker Street, 1NW London	0044 20 7946 1000			10.01.2012
Miller	Ann	55 Oxford Street, W1 London	0044 20 7946 3000	0044 7701 678912	0044 20 7946 3001	18.06.1984
Miller	Paul	55 Oxford Street, W1 London	0044 20 7946 3000			
Miller	Steven	55 Oxford Street, W1 London	0044 20 7946 3000			
...	...	...	...	...	...	...

2.19 Development Cycle

1. Design database by logic
   - What must be in the database?
2. Design database for system
   - How should the data be saved?
3. Develop database applications
   - How is the data processed?
4. Fill database
   - How to insert the data?
5. Maintain database
   - the database runs and runs and runs…



2.19 Development Cycle

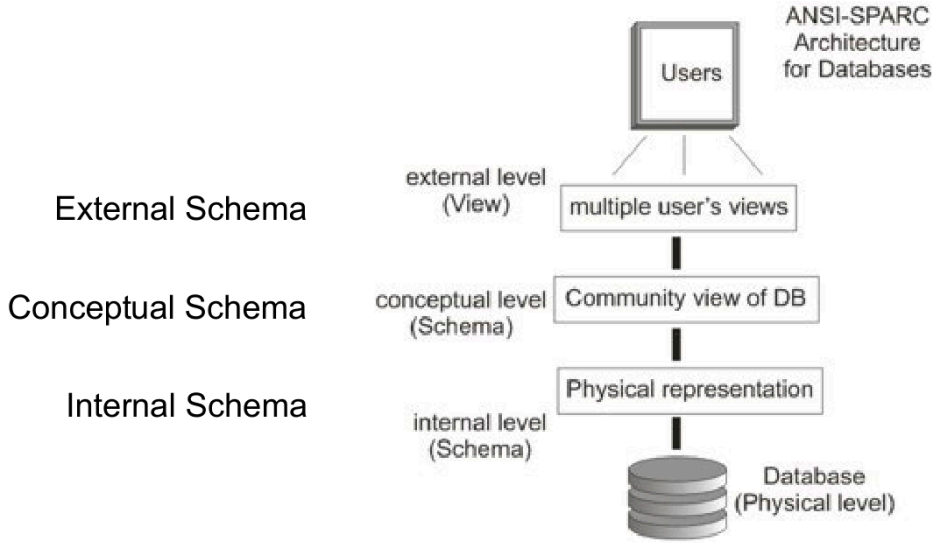


Source: http://www.jhouseconsulting.com/jhouseconsultin
wp-content/uploads/2008/10/howthecustomerexplainedit.

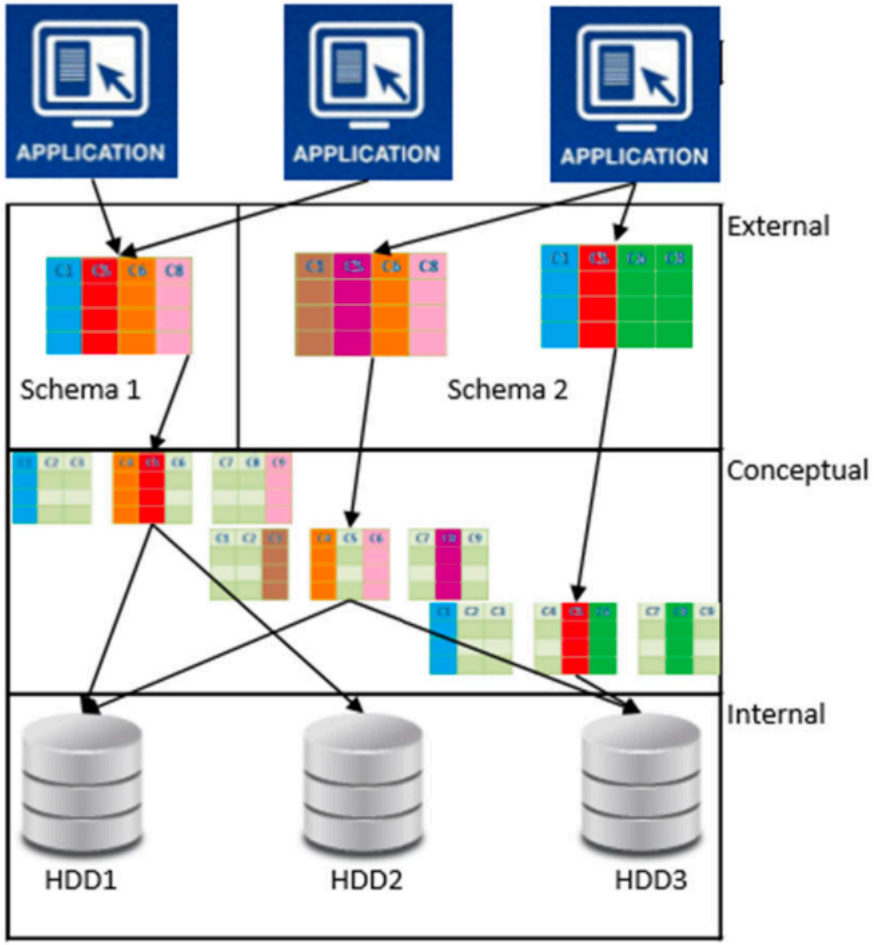
HAW HAMBURG

2.20 ANSI-SPARC Architecture

- The **external schemas** describe the different external views of the data and there may be many external schemas for a given database
- The **conceptual schema** describes all the data items and relationships between them, together with integrity constraints
  - There is only one conceptual schema per database
- The **internal schema** at the lowest level contains definitions of the stored records, the methods of representation, the data fields, and indexes
  - There is only one internal schema per database



2.20 ANSI-SPARC Architecture



2.21 Data Independence

- Database Systems (and its data) must be accessible for a long time
  - Example: Insurance Company (decades)
  - Often longer than the lifetime of applications, operating systems and hardware
- Need for data Independence
  - Avoid tight coupling of applications, data and operating system
  - Physical data Independence
  - Logical data independence

2.22 Data Independence - Physical Data Independence

- **Definition**: Ability to modify the physical storage structures or locations of data without affecting the logical view of the data.

- **Examples**:
  - Changing underlying hardware, file systems, storage devices, or relocating the database to a different server without affecting applications.
  - Physical implementation details (e.g., storage format, indexing methods) are hidden from the application layer.

- **Benefits**:
  - Ease of adapting to changing technology or infrastructure without rewriting application code.
  - Allows performance optimization without altering application's logical data access methods.

2.23 Data Independence - Logical Data Independence

Definition: Ability to change the logical schema (table structures, relationships, and attributes) without impacting the applications that use the database.

- **Examples**:
  - Adding a new column to a table or modifying table relationships without requiring application code changes.
- **Benefits**:
  - Flexibility to adapt database schema to evolving business requirements without disrupting applications.
  - Simplifies data model changes and maintains data consistency.

2.24 Data Independence

Memorize

Applications are robust against changes in the database or operational environment if data independence is maintained.

2.25 Homework

Memorize

Install a DBMS on your home computer (preferred: PostgreSQL, MariaDB, mySQL, Oracle)!

3. License Notice

3.1 Attribution

- This work is shared under the CC BY-NC-SA 4.0 License and the respective Public License.
- https://creativecommons.org/licenses/by-nc-sa/4.0/
- This work is based off of the work by Prof. Dr. Ulrike Herster.
- 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.