Introduction

Databases
Spring 2025

Ira Assent

ira@cs.au.dk

Data-Intensive Systems group, Department of Computer Science, Aarhus University, DK

Intended learning outcomes

- Be able to
  - Answer: "Why am I having this course and what will I learn?"
  - Describe applications that use databases
  - Understand and use basic database terminology

ira@cs.au.dk

Who we are, what we do

- Instructor: Ira Assent
- "Super" TA Amalie Brogaard Pauli
  - Support communication, course structure, checking contents, exercises…
  - Data-Intensive Systems research group, 3rd floor Nygaard
    - We work with data, hence the name
      - Data management, like in databases, similarity search,…
      - Data analysis, like in data mining, machine learning, artificial intelligence…
- TAs for exercises classes and at study café
  - See "Who, where, when" on course homepage

ira@cs.au.dk

AUVote

- Interactive multiple choice
  - Allows me to see how we're doing
    - Speed up, slow down / explain
  - I like open or even confusing questions
    - The goal is discussion not testing
    - When you think about when which option is correct, you can reason about the application of the theory
    - Of course, there are also plain simple questions with a single correct answer
  - Anonymous
    - Please vote even if you're unsure

- Do not worry, the questions in the final multiple choice exam are of course NOT designed to be open or confusing!

ira@cs.au.dk

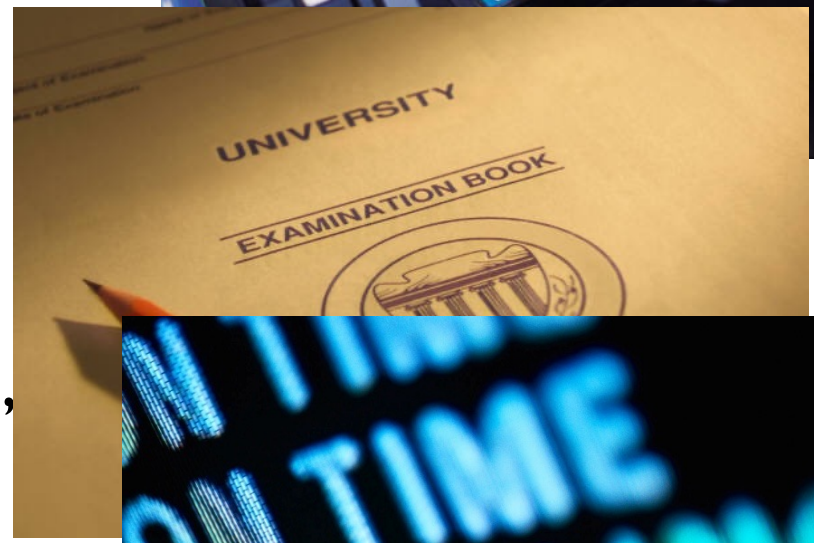
Have you worked with a database before?

- No, not that I know of.
- As part of a program that I have used.
- Yes, I've written software that interacts with a database.
- Yes, I've set up a database as part of a program.

ira@cs.au.dk

Database Application Examples

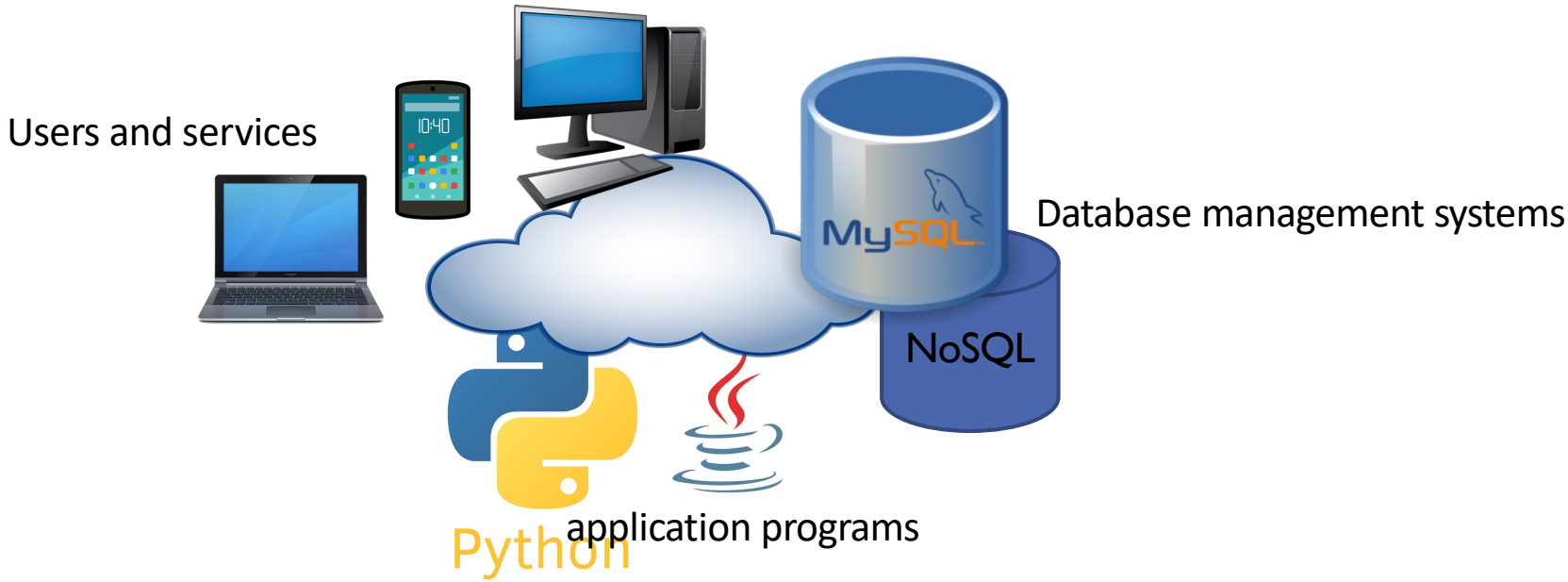
- Banking accounts
- University records
- Airline reservations
- My address book
- The e-shop "around the corner"



ira@cs.au.dk

Aims and Objectives

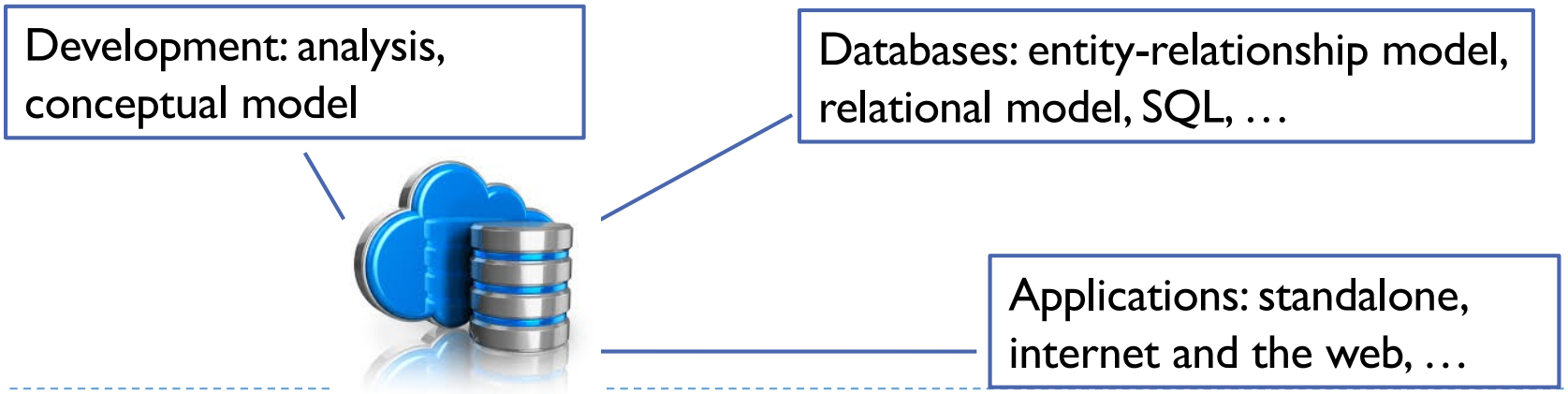
In this course we learn **concepts** and **techniques** for the **design** and implementation of **database** applications with relational database management systems, or NoSQL, and basic **theoretical foundations, protocols** and **strategies** for achieving these properties



ira@cs.au.dk

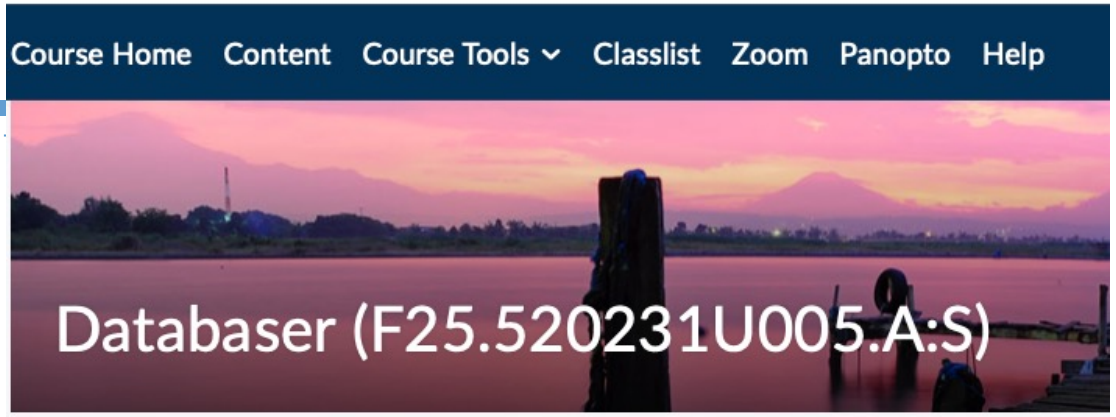
Course Topics

- Modeling and defining a database (E/R)
- Working with a relational database (SQL)
- Ensuring a well-working database (normalization, triggers, indexes)
- Putting a database to use (connection program with database, authorization)
- Database applications (Information Retrieval, Web Search, Data Mining and Data Warehouses)
- Behind the scenes (Concurrency control / recovery, query evaluation / optimization)
- Theoretical foundations (Relational algebra and calculus)
- Scaling (NoSQL, distributed databases)



ira@cs.au.dk

Course information



- Course Brightspace page
  continuously updated with all information
  - Lecture slides, presentation exercises, hand-ins, quizzes ……
  - Any comments or feedback please reach out
- Make sure you are enrolled in your TA groups and hand-in groups (both!)
  - **You do not see assignments otherwise!**
- Forum: make good use of it
  - Your questions here benefit everyone
  - And your answers even more so!

ira@cs.au.dk

Lectures and Exercises

- Lectures Tuesdays 12-14, Fridays 8-10
- TA sessions Tuesday afternoon – Thursday
  - Topics introduced at lecture, supported by reading material, studied in exercise classes, then in handin assignments, due on Friday 6pm
  - New: we monitor attendance (Qwickly tool) at the exercise sessions to identify and support students who may be at risk of falling behind, from second course week
  - *This week's exercises are different* – we use them to support you in
    - installing MySQL workbench and trying it out
    - Subscribing and using the discussion forum
    - Feel free to skip if you take care of this yourself – it is part of first handin!
  - From next week on, exercise sessions cover material and tasks that help you study the material and prepare for hand-ins

ira@cs.au.dk

Study café and handins

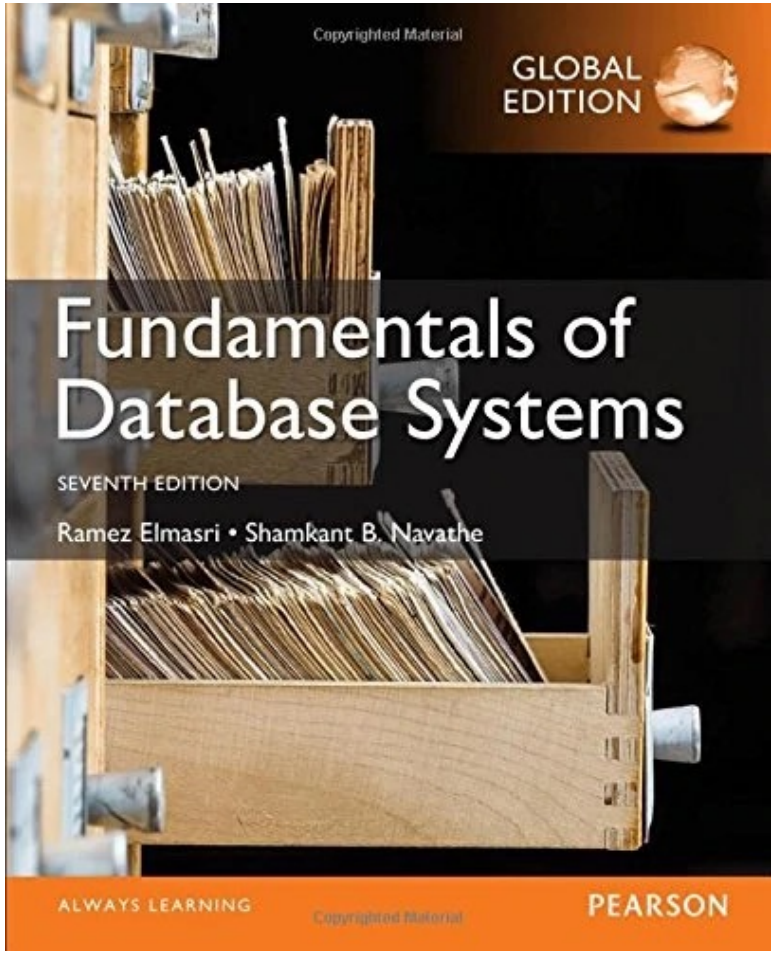
- Study café sessions throughout the week
  - The place to be!
  - Note that we do not have a session on Monday (post-handin) based on experience from previous years; we will monitor attendance on Friday (pre-handin) if it should be too busy (you are welcome to reach out to us also if that should be the case)
- Hand-in assignments
  - Deepen your understanding, gain practical experience
  - We want you to discuss and document how you work with the material!
- Default groups size 2-3, depending on your study program

ira@cs.au.dk

Textbook

- Fundamentals of Database Systems, **7th Edition**
  R. Elmasri, R. B. Navathe
  Pearson
  - **Global edition**

  - Provides background reading material
    - Definitions and examples
    - Context



ira@cs.au.dk

How to read

- Weekly reading material announced on brightspace
  - "religious" reading can be overwhelming
- What we need is "context" reading
  - Skim over the text relatively quickly
    - Take note of the context in which different terms, notations and approaches are introduced
  - Informs preparation for / follow up on lecture
  - When working with exercises, hand-ins, reviewing of lectures
    - Know where which information is found
    - Be able to quickly look up what you lack
  - "Proper" Googling: if you search for information without knowing the context, you may not understand whether an online source is what you need or a different concept with similar or same name
    - Often leads to more confusion than understanding
    - ChatGPT may or may not be right
      - In any case, not helpful for learning

Assessment

- Databases (DB):
  - Handins count for 20%, the multiple choice exam in June for 80% of the grade
- Database Systemer (DS, 5 ECTS, IT Product Development)
  - All weekly handins passed (re-handins possible)
- Implementation and Applications of Databases (IADB, 5 ECTS)
  - Handins in 2nd quarter count 20%, MC exam on 2nd quarter 80%
- Handins provide continuous assessment and feedback
  - Work in groups
    - Put the names of all students who contributed
    - Do not put the names of any student who did not contribute
      - This is (exam) cheating and is a serious issue that can have severe consequences
    - You need to include a statement on use of generative AI (see handins), else your handin will not be graded and will not receive any points
  - Make sure you follow the rules regarding plagiarism and (exam) cheating, including generative AI
    - See also course webpage

ira@cs.au.dk

How to pass this course (and learn something useful while you are at it)

1. Make sure to gain an overview over each week's material
   - Lectures, reading material, initial review at the TA session
2. Make sure to work with the example problems
   - Prepare the exercise problems, work through them at the TA sessions, review the solutions afterwards
   - Try out examples, in particular SQL queries in MySQL, step through algorithms / protocols
3. Do the handins
   - They are carefully aligned with the exercises (and the lectures and the reading material…)
   - They get very difficult and confusing if you skip the previous steps
   - You should be well-prepared for them if you did the previous steps
   - They prepare you for the exam
   - They contribute to your grade

ira@cs.au.dk

How to pass this course (and learn something useful while you are at it) -2-

4. Get used to the reality of databases in the real world
   - Modeling the real world to fit it into a database can feel "fuzzy", "ill-defined", "vague"
     - Other courses may be more formal and easier in this regard
     - The world is not a formal language
     - You need to make assumptions, and argue for them when modeling and implementing databases
       - This is what you should practice
       - Also in later jobs, you will need to make assumptions, validate them, and build the database that you consider best based on your own analysis!
5. Get involved in, maintain, create study community with your fellow students

   - Ask questions on the discussion forum
   - Answer questions on the discussion forum
   - Read questions and answers on the discussion forum
   - Be part of the study café, make good use of your study groups and peers

ira@cs.au.dk

Guidelines for your own review of today's session

- In this course, we will study…
- Databases are found in…
- DBMS provide useful properties including…
  - If we were to use files, then…
  - However, in the following situations, a DBMS may not be needed…
- Some of the terminology is the following…
  - They are defined as…
- The data model studied in the majority of the course is…

- In order to follow this course and pass it successfully …

ira@cs.au.dk

Some terminology

- **Database**
  - Collection of related data
  - Known facts that can be recorded and that have implicit meaning
- Example of a large commercial database
  - Amazon.com

- **Miniworld** or **universe of discourse (UoD)**
  - Represents some aspect of the real world
  - Logically coherent collection of data with inherent meaning
  - Built for a specific purpose
    - E.g. handling orders in e-commerce

  - This is a core aspect in this course: we cannot and should not model everything in our world – just exactly what is needed, not more, not less

ira@cs.au.dk

DBMS

- **Database management system (DBMS)**
  - Collection of programs
  - Enables users to create and maintain a database
  - E.g. MySQL, SQLite, PostgreSQL,…
  - No need to implement data management from scratch!

- Databases make use of the DBMS to support data management
  - Easy to put in data, ensure data quality and retrieve data

- Many current databases are relational databases (SQL mostly): focus of this course
- Some large databases are recently NoSQL (examples later in this course)

ira@cs.au.dk

Could I just use a text editor instead of a database to store student information?

- Sure, if you don't have the time to install a database.
- Sure, if there is only a single user.
- No, not unless you never have any updates.
- Certainly, especially if you are working on a tiny system (e.g. embedded system)

ira@cs.au.dk

When Not to Use a DBMS

- More desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Stringent, real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data

ira@cs.au.dk

Creating a database

- A database application is a collection of data and the programs that allow the manipulation of these data
- Defining a database
  - Specify the data types, structures, and constraints of the data to be stored
  - Next lecture: a principled approach to defining the database using E/R modeling
- **Meta-data**
  - Database definition or descriptive information
  - Stored by the DBMS in the form of a database catalog or dictionary
- Adding data
- Manipulating a database
  - Query and update the database

ira@cs.au.dk

An example from the textbook

- UNIVERSITY database
  - Goal: build a database to manage university information
  - Miniworld sketch: information concerning students, courses, and grades in a university environment
    - Approach: design a database that handles data records to maintain this information
- **Data records**
  - STUDENT
    - Different student records using the same record format
  - COURSE
  - SECTION
  - GRADE_REPORT
  - PREREQUISITE

ira@cs.au.dk

The University example continued

- Construct UNIVERSITY database
  - Store data to represent each student, course, section, grade report, and prerequisite as a record in appropriate file
- Relationships among the records
  - A crucial aspect
    - By being able to relate a STUDENT record to a GRADE-REPORT record, we establish its usefulness!
    - If we did not know which student got which grade, the database would be useless
      - We will revisit this when discussing keys
- Manipulation involves querying and updating
  - Databases are only useful if they provide easy access to data (querying), and easy maintenance (updating)

ira@cs.au.dk

University example records

- Each record in a table has the same format
  - E.g. Smith, Brown in STUDENT
- Entries differ naturally

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

ira@cs.au.dk

Using the University example database

- Examples of queries:
  - Retrieve the transcript
  - List the names of students who took the section of the 'Database' course offered in Fall 07 and their grades in that section
  - List the prerequisites of the 'Database' course
- Examples of updates:
  - Create a new section for the 'Database' course for this semester
  - Enter a grade of 'A' for 'Alexandra' in the 'Database' section of last semester

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

ira@cs.au.dk

Relationships

- Values in a table (e.g. GRADE_REPORT) connect to records in another table (e.g. STUDENT)
  - we will see how this works (keys)

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

ira@cs.au.dk

DBMS makes our data handling easier!

- Controlling redundancy
  - No need to enter the same information repeatedly
    - E.g. student address with each student grade
    - Avoids data integrity issues
      - E.g. the same student is registered with two addresses
    - **Data normalization**
    - **Denormalization**
      - Sometimes necessary to use **controlled redundancy** to improve the performance of queries
- Restricting unauthorized access
  - Security and **authorization** subsystem
  - Privileged software

ira@cs.au.dk

And there is more from the DBMS

- Storage structures and efficient search techniques
  - E.g. indexes, an efficient data structure for looking up the right data quickly
    - Means user/program wait times greatly reduced
- Backup and recovery
  - In case of system crash, return to valid state of data
- Multiple user access
  - A major benefit of databases: more than one user / program (thread) can access the database at the same time
  - Graphical user interfaces for standalone access

ira@cs.au.dk

And even more

- Representing complex data relationships
  - E.g. a student can have several grade reports, grade reports refer to a particular course, student and year, etc
    - Enforcing integrity constraints
    - Referential integrity constraint
      - Every section record must be related to a course record
- **Key** or **uniqueness constraint**
  - Allows us to find the right data, e.g. which "Ann Jensen"?
  - Every course record must have a unique value for Course_number
- **Business rules** / semantic rules
  - E.g. cannot enter students who are more than 150 years old
- **Inherent rules** of the data model
  - As we will see, depending on how we model and represent the data, this may imply restrictions
    - E.g. in E/R models (next lecture), a relationship connects at least two entities

ira@cs.au.dk

And more yet

- **Triggers**
  - allow updates to the data based on other changes
- **Abstraction**
  - All data handling taking care of by DBMS
  - We / users / programs do not need to worry about data handling
  - Data storage can be changed
    - Access via well-defined interfaces
      - Reduced application development time
      - Flexibility
      - Availability of up-to-date information
      - Economies of scale

ira@cs.au.dk

Which of these two models is better?

- Model 1, because all information is in just one table
- Model 2, because information is grouped
- Both are equally good
- It depends on the context of use

Model 1

id	name	last name	course	year	teacher	room	study	grade
1	Ann	Jensen	DB	2022	Ira	PBA	CS	12

Model 2

id	name	last name	study
1	Ann	Jensen	CS

id	course	year	grade
1	DB	2022	12

course	year	teacher	room
DB	2022	Ira	PBA

ira@cs.au.dk

Database Models

- Conceptual Model
  (for analysis and design)
  - What is the database supposed to handle and how?
  - Entity-Relationship
    - Covered next lecture
- Logical Model
  (for design and implementation)
  - How do we actually store the conceptual model and associated data?
  - Relational Model
    - More to come, basis for most of this course
- Physical Model
  (usually not visible, need to be understood for tuning)
    - How is the data actually stored, such that requirements like persistence, efficient access, etc are met

ira@cs.au.dk

How to get started?

- So, how do we know which tables we need?
  - Which columns and rows?
- Design of databases coming up next
  - "modeling": identify what is relevant, how to be used
  - There is often more than one correct solution
  - And almost always many ways to get it wrong…
- Structured process to create well-formed databases
  - Why?
    - A poor database can be
      - Slow
      - Redundant
      - Contain faulty or inconsistent data
      - Miss relevant data
      - …
  - Time spent on design is time spent well
    - Saves a lot of hassle in fixing issues in a (more or less) working system!

ira@cs.au.dk

Database Management System (DBMS)

- A DBMS is a generic platform for the development and management of database applications
- Provides the software to easily build, use and integrate a database into a software system
- Example commercial DBMS are:
  - Oracle
  - Sybase
  - DB2
  - Microsoft SQL Server
  - MySQL
  - Microsoft Access

MySQL™

ORACLE

SYBASE

ira@cs.au.dk

MySQL

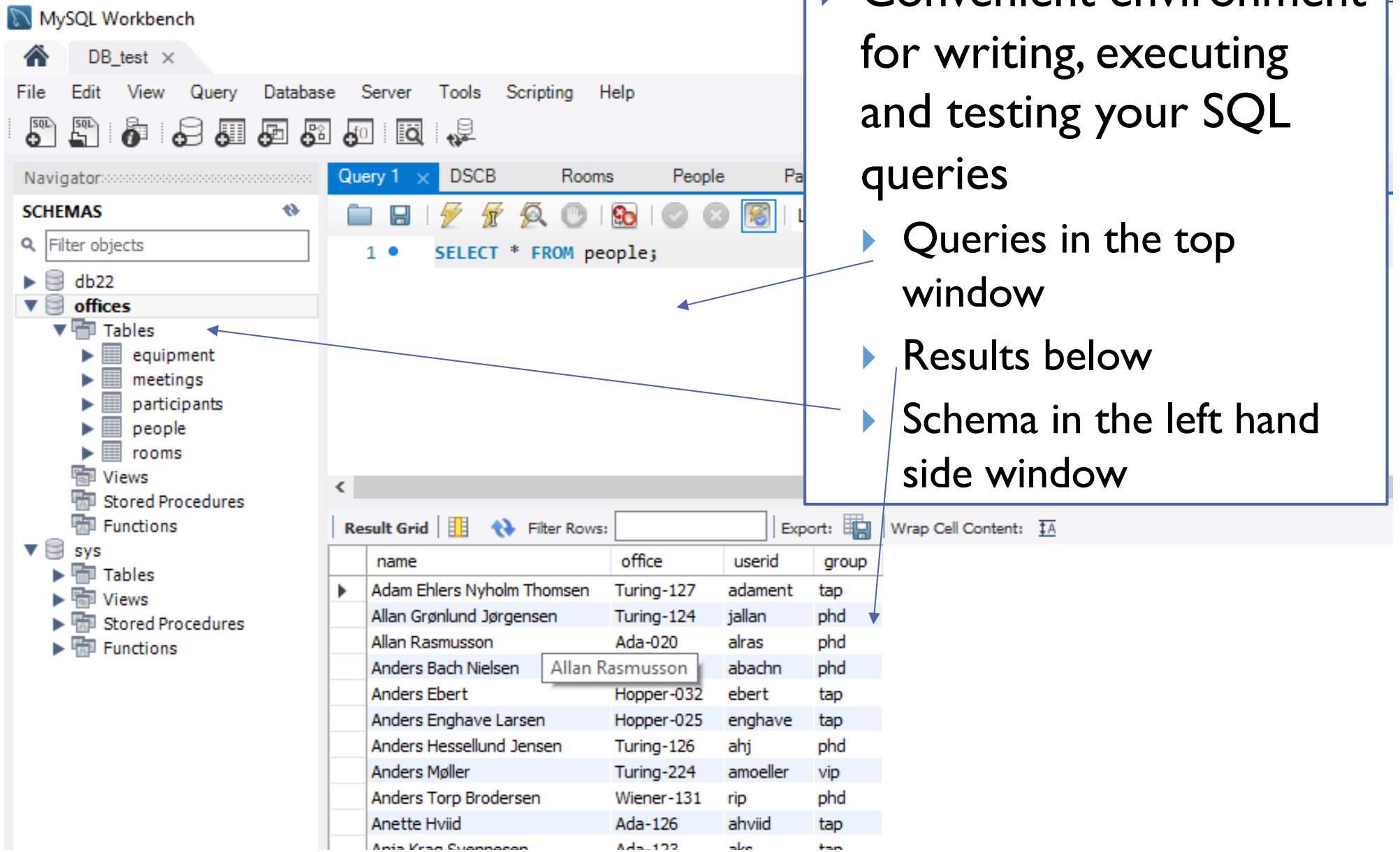
- We make use of MySQL as a database to try out what we learn
  - open-source, free database management system
  - We make use of MySQL workbench for ease of use

- Introduction and support by your TAs in this week's exercise session

- Provides you with a means of trying out SQL in practice, and checking your solutions work

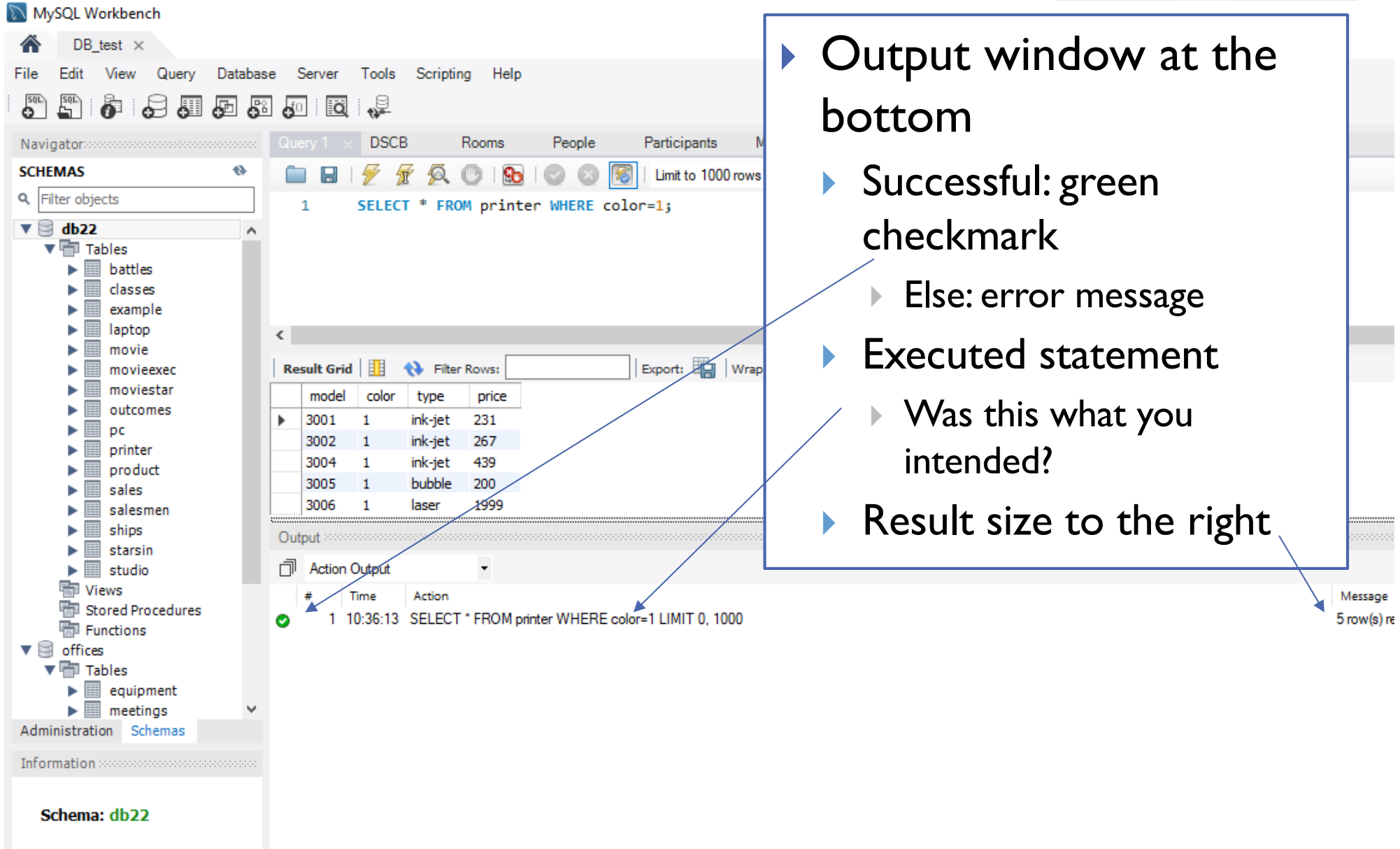
ira@cs.au.dk

MySQL workbench



- Convenient environment for writing, executing and testing your SQL queries
  - Queries in the top window
  - Results below
  - Schema in the left hand side window

MySQL workbench



- Output window at the bottom
  - Successful: green checkmark
    - Else: error message
  - Executed statement
    - Was this what you intended?
  - Result size to the right

Relational Model

- Introduced in the 1970s, still the most widely used
  - Others include object model (1980s), object relational model (1990s), XML, graph based (mostly since 2000s), NoSQL (mostly since 2010s)
- Our focus is mostly on this model, but we will also see examples of alternatives (NoSQL, incl. graph based))

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

- Use a simple data structure: the table
  - simple to understand
  - useful data structure (captures many situations)
  - leads to useful yet not too complex query languages
    - We'll see more when we start with the SQL language

ira@cs.au.dk

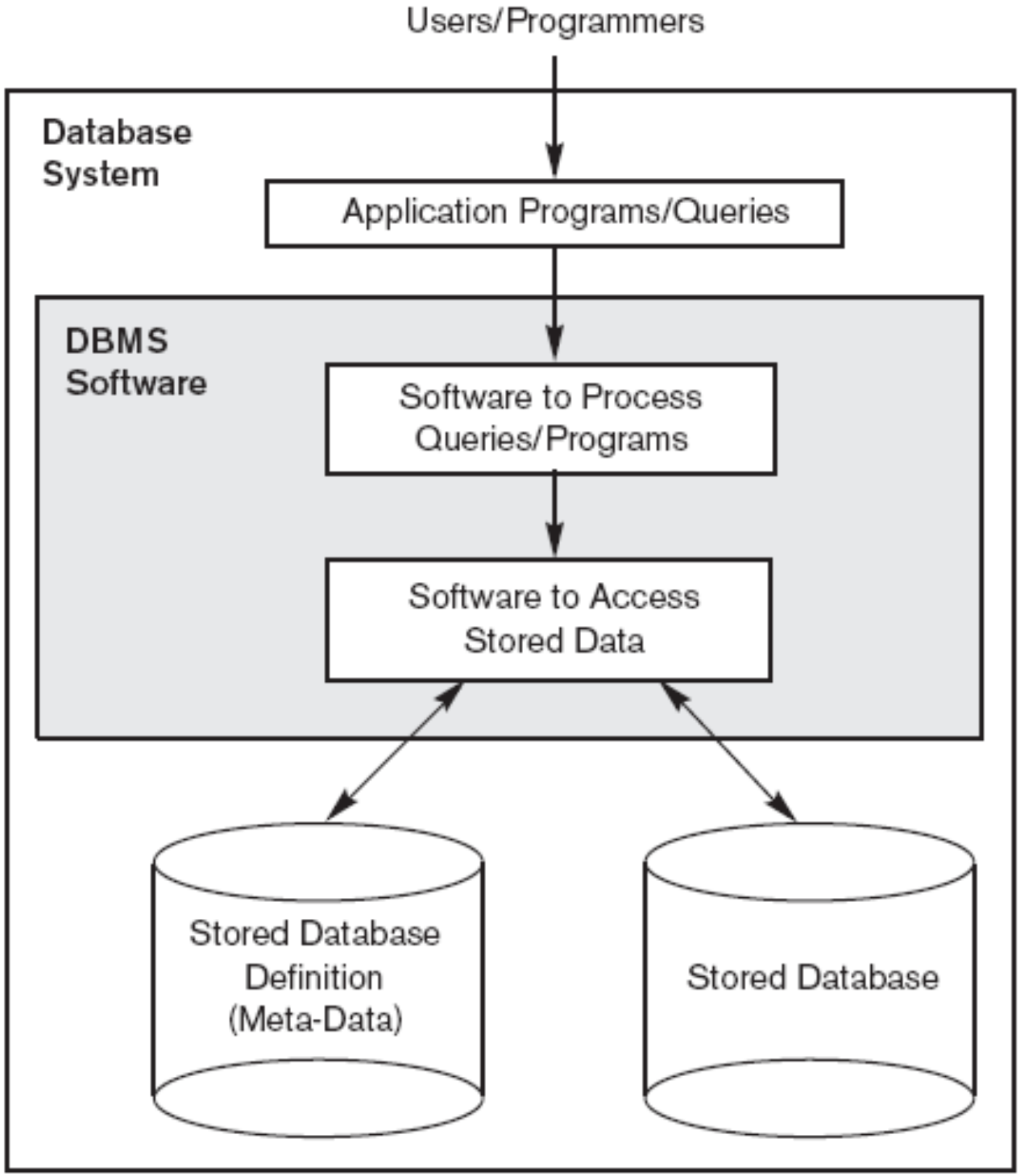


Figure 1.1
A simplified database system environment.

Summary

- Database
  - Collection of related data (recorded facts)
- DBMS
  - Generalized software package for implementing and maintaining a computerized database
  - Provides easy access to properties such as efficient access, multi-user access, data integrity, …
  - Removes the need to implement your own data handling for each program / application you develop!

ira@cs.au.dk

Conclusion

- Intended learning outcomes
  - Be able to
    - Answer: "Why am I having this course and what will I learn?"
    - Describe applications that use databases
    - Understand and use basic database terminology
- Quiz on terminology – optional, but recommended

- Exercises:
  - On course home page available
    - This week atypical and optional: installation and test of MySQL workbench, subscribing to discussion forum
    - From next week: exercises to review course content, apply to example problems, and prepare for handins

ira@cs.au.dk

What was this all about?

Guidelines for your own review of today's session

- In this course, we will study…
- Databases are found in…
- DBMS provide useful properties including…
  - If we were to use files, then…
  - However, in the following situations, a DBMS may not be needed…
- Some of the terminology is the following…
  - They are defined as…
- The data model studied in the majority of the course is…

- In order to follow this course and pass it successfully …

ira@cs.au.dk