Web & Database Technologies

Engineering and you and what that has to do with Data Modelling

Christoph Lofi

TU Delft WIS Web Information Systems

Why databases then (and not just files)?

- **1. Data Integrity & Consistency**
  - File systems lack built-in mechanisms for enforcing data rules.
  - Databases use constraints, transactions, and ACID properties to ensure reliable data
    - ACID Transactions: simplified “Complex operation will be executed reliably even with multiple uses or system failures”
- **2. Concurrency Control**
  - Multiple users accessing files can lead to conflicts or inconsistencies.
  - Databases handle concurrent access safely with locking and isolation levels.
- **3. Querying & Flexibility**
  - Databases offer powerful query languages (e.g., SQL) for fast, flexible data retrieval.
  - Languages are declarative: describe what you want, not how to get it!
- **4. Scalability & Performance**
  - File systems degrade with large data volumes and complex access patterns.
  - Databases are optimized for indexing, caching, and efficient storage.
- **5. Security & Access Control**
  - File systems offer basic permissions.
  - Databases provide fine-grained access control, authentication, and auditing.
- **6. Backup & Recovery**
  - Manual and error-prone in file systems.
  - Databases support automated backups, replication, and recovery tools.

Today

- Flipped Classroom Session: Let's explore and discuss
  - Engineering and Data Management
  - Modelling Introduction
- Learning Goal:
  - Understand the **engineering lifecycle**
  - "Realize" and understand that **modelling** and designing is an important, but very free and "soft" task which has to be done with care….
- ER Diagramming is a Tutorial Topic / study-at-home topic

The Role of Relational Databases

The Role of Relational Databases

The Role of Relational Databases

???

Reminder: Lab and Tutorial Today!

- 13:45 in Flux!
  - Discuss with your group
  - Try to solve your homework
  - Discuss with our TAs!
  - Attend the Tutorial of Albert!

Lecture Goals Today

- Today
  - Engineering and CS
  - Why **Modelling**?
  - Introduction to Entity-Relationship Diagrams (ER)

- Next Lecture
  - Advanced ER Modelling
  - How to convert ER diagrams into Relational Database Tables

Quiz Time!

How do you start building a house?

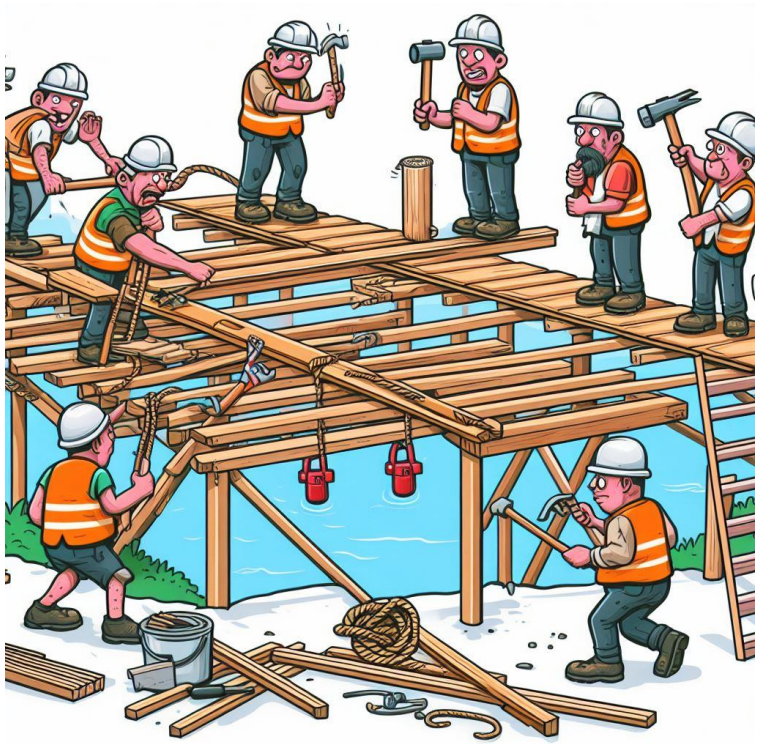
a) By stacking bricks on top of each other until you have a house

b) By carefully planning and modeling the house

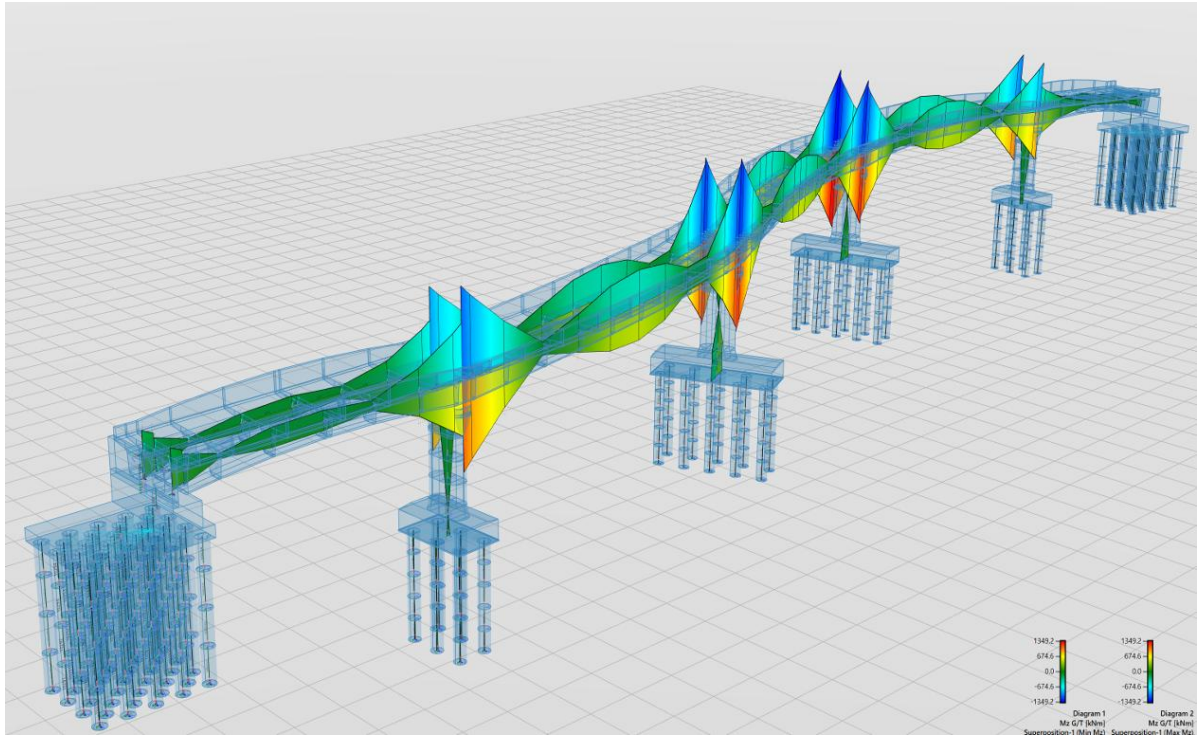
Quiz Time!

How do you start building a bridge?

a) By going to the river and start hammering



b) By carefully planning and modeling the bridge



Quiz Time!

How do you start building software?

a) By hacking away code until you created the software you wanted

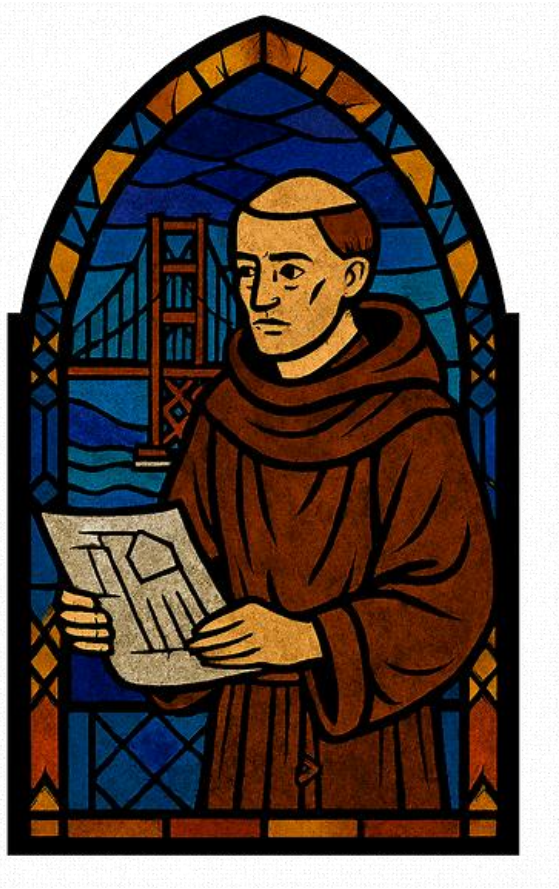
b) By carefully planning and modeling the software

Modeling Disasters

Engineering and You.

TU Delft WIS Web Information Systems

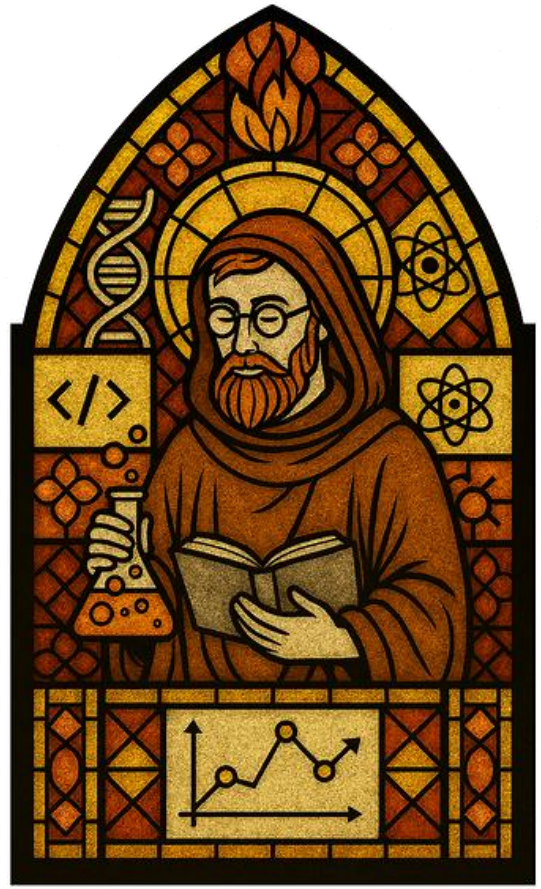
What is an Engineer?



Engineer

- *Goal*:
  - Applies scientific and technical principles to design and build practical solutions, and assesses and maintains them.
- *Focus*:
  - Systems, infrastructure, machines, processes.
- *Output*:
  - Physical or digital products that solve real-world problems
- *Mindset*: “How can I address this problem systematically, reliably, and safely?”

What is a Scientist?



Scientist

- *Goal*:
  - Expands knowledge by discovering new truths about nature, phenomena, or (philosophical) constructs.
- *Focus*:
  - Research, experimentation, theory.
- *Output*:
  - Knowledge, models, and insights.
- *Mindset*: “Why does this happen? What are the underlying laws? How can I control this?”

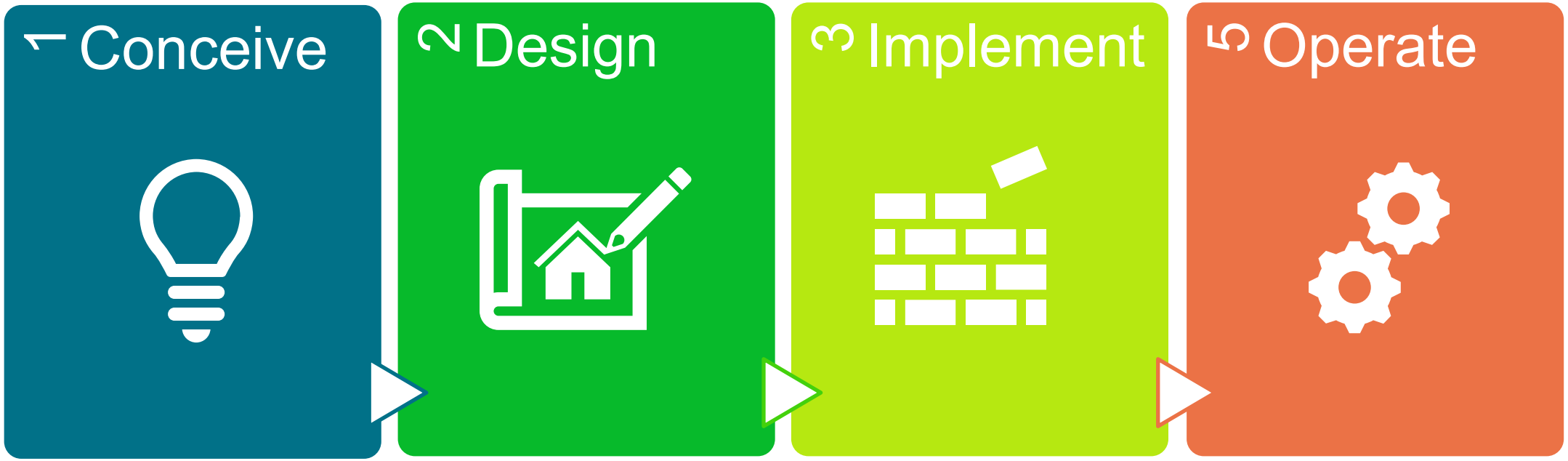
What is a Programmer?

Programmer

- *Goal*:
  - Writes code to implement logic, automate tasks, or build software
- *Focus*:
  - Algorithms, programming languages, debugging.
- *Output*:
  - Software applications, code, scripts.
- *Mindset*: “How do I make this program run?”

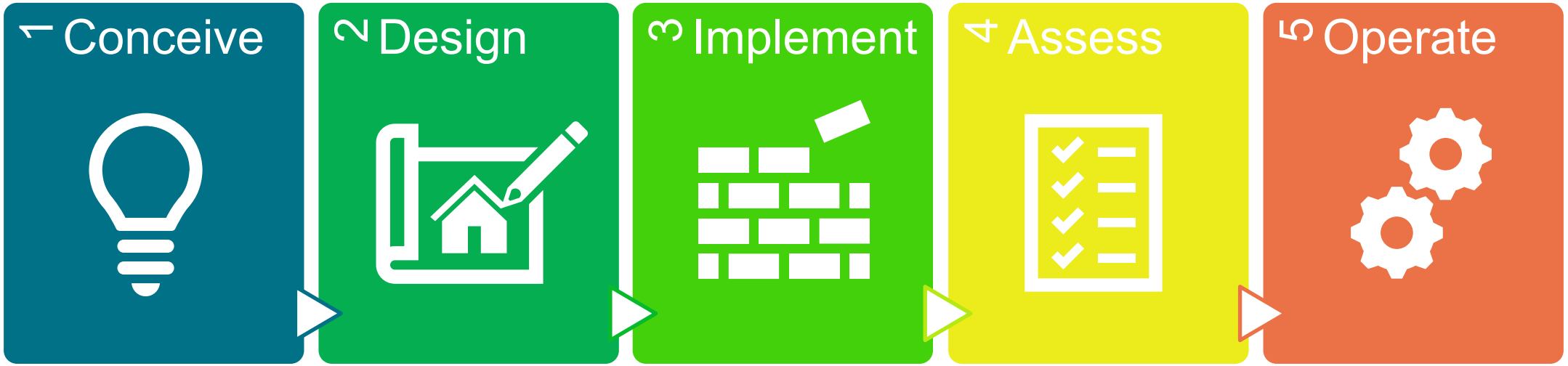
The Engineering Workflow

- *CDIO INITIATIVE as an example for formalizing engineering education*



The Engineering Workflow

- CDIO not sufficient, make it CDIAO
  - We are Computer Science Engineering!!



The Engineering Workflow

1. Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

2. Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

3. Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

4. Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and documentation.

5. Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

6. Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

7. Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

8. Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

The Engineering Workflow

Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

The Engineering Workflow

Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and documentation.

Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

The Engineering Workflow

Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

We typically provide you with “user need stories” in exam / labs

Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

We typically take this one from you…we will do RelationalDBs ☺

Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

Conceptual DB Design

Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and

Logical DB Design (physical design in CSE1505)

Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

Developing the DB schema and SQL queries

Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

The Engineering Workflow

Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

We typically provide you with “user need stories” in exam / labs

Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

We typically take this one from you…we will do RelationalDBs ☺

Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

Conceptual DB Design

Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and

Logical DB Design (physical design in CSE1505)

Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

Developing the DB schema and SQL queries

Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

😱 😱 😱 ….shame. We actually do not really go into those much in THIS course…

The Engineering Workflow

1. Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

2. Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

3. Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

4. Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and documentation.

5. Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

6. Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

7. Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

8. Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

“MOST IMPORTANT SLIDE OF THE DAY!

1. Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

2. Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

3. Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

4. Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and documentation.

5. Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

6. Testing and Validation

- **Goal:** Ensure the solution meets requirements.
- **Activities:** Unit testing, system testing, performance evaluation, compliance checks, user studies.
- **Output:** Verified and validated solution.

7. Deployment / Delivery

- **Goal:** Deliver the solution to users or clients.
- **Activities:** Installation, training, documentation handover.
- **Output:** Operational system or product.

8. Maintenance and Feedback

- **Goal:** Monitor performance and improve over time.
- **Activities:** Maintenance, updates, user feedback, troubleshooting.
- **Output:** Sustained performance and iterative improvements.

Crosscutting: Management Layer

- Throughout all stages, engineers often follow project management practices like:
- **Planning and scheduling**
- **Risk management**
- **Budgeting**
- **Team coordination**

Towards Data Modelling

A non-technical intro

TU Delft WIS Web Information Systems

The Engineering Workflow

Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

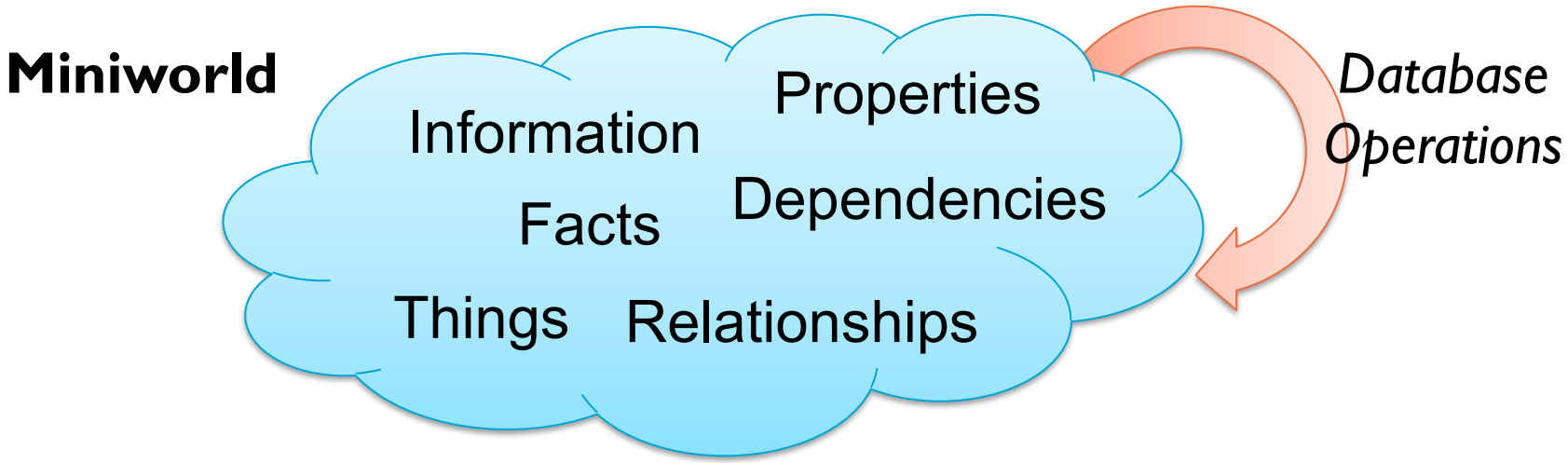
Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

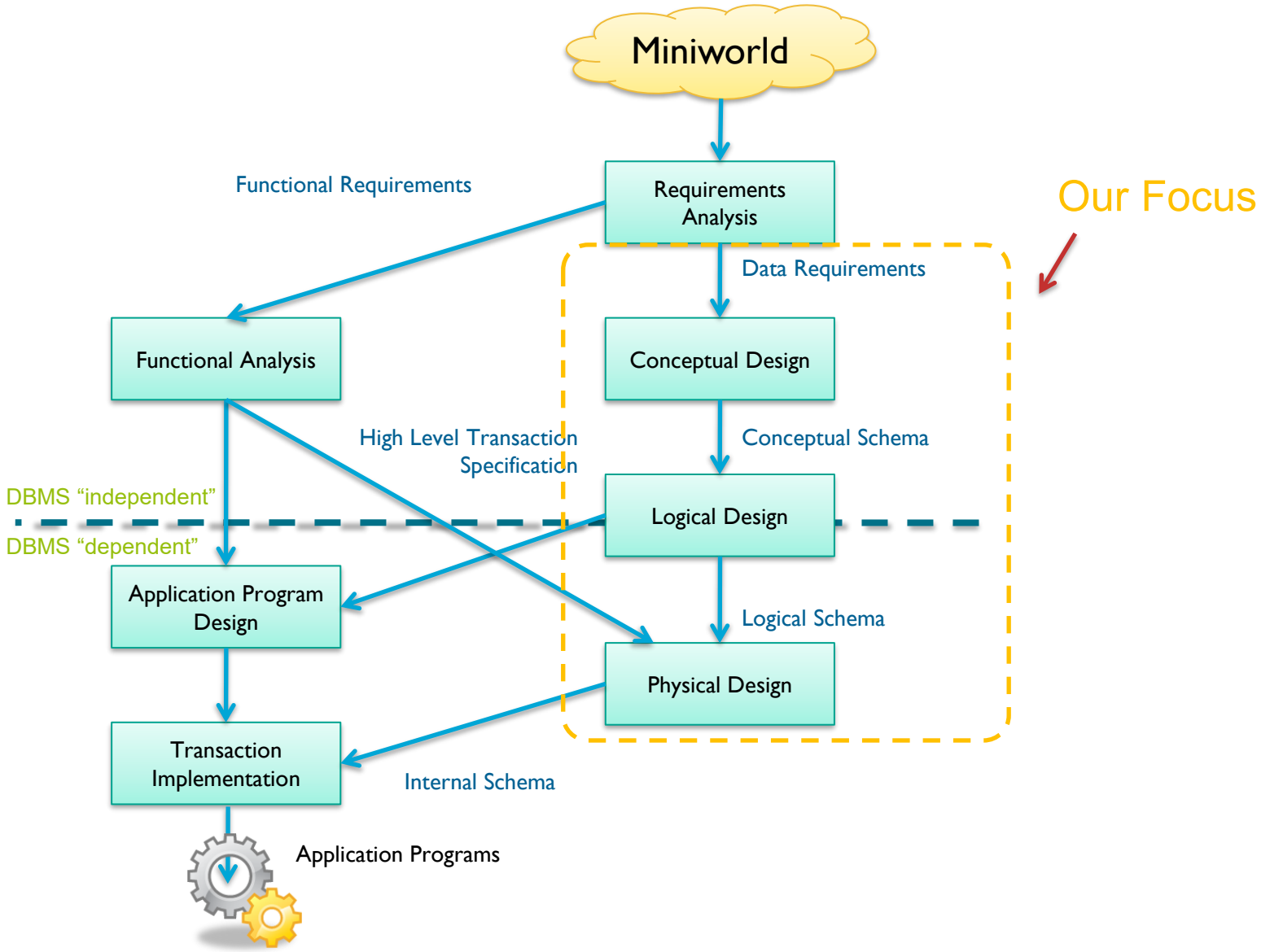
Check our courses on Software Engineering Methods / Software Qualiy

Towards Schema Design

- Data modelling models the data requirements of a **miniworld** into a formal representation
  - Restricted view on the real world with respect to the problems that the current application should solve
  - mini world: “universe of discourse”



Towards Schema Design



Towards Data Modelling

Requirements Analysis

- Database designers interview prospective **users** and **stakeholders**
- **Data Requirements** describe what kind of data is needed
- **Functional Requirements** describe the operations performed on the data

Functional Analysis

- Concentrates on describing **high-level** user operations and transactions
  - Does not yet contain implementation details

Towards Schema Design

Conceptual Design → Conceptual Schema

- Transforms Data Requirements to **conceptual schema**
- Describes high-level data entities, relationships, constraints, etc.
  - **Does not contain any implementation details**
  - **Independent of used software and hardware**
- Only loosely depending on the chosen data model!!
  - Should actually be independent…

Logical Design → Logical Schema

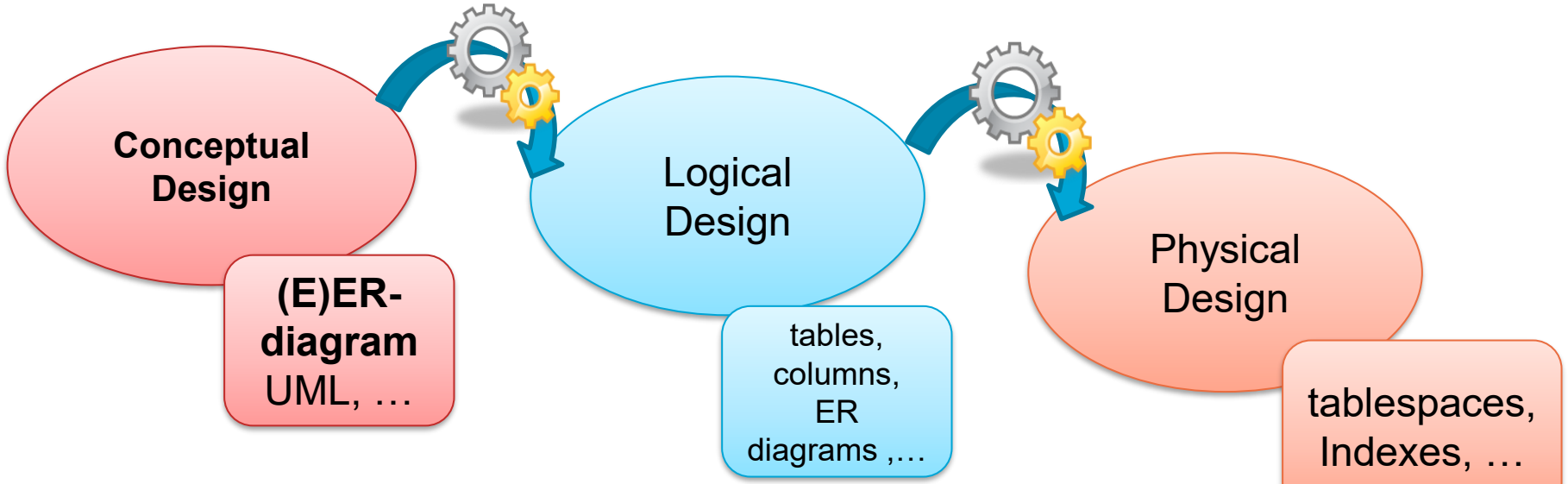
- Maps the conceptual schema to the Data Model used by the DBMS
  - e.g., **relational model**, document model, etc
  - Depends on the **data model** of the chosen DBMS

Physical Design

- Designs the internal structures needed to efficiently store/manage data
  - e.g., table spaces, indexes, access paths
  - Depends on the used hardware and DBMS software

Towards Schema Design

- Data Modeling the data involves three design phases
  - The result of one phase is input of the next phase
  - Often, automatic transition is possible with some additional designer feedback

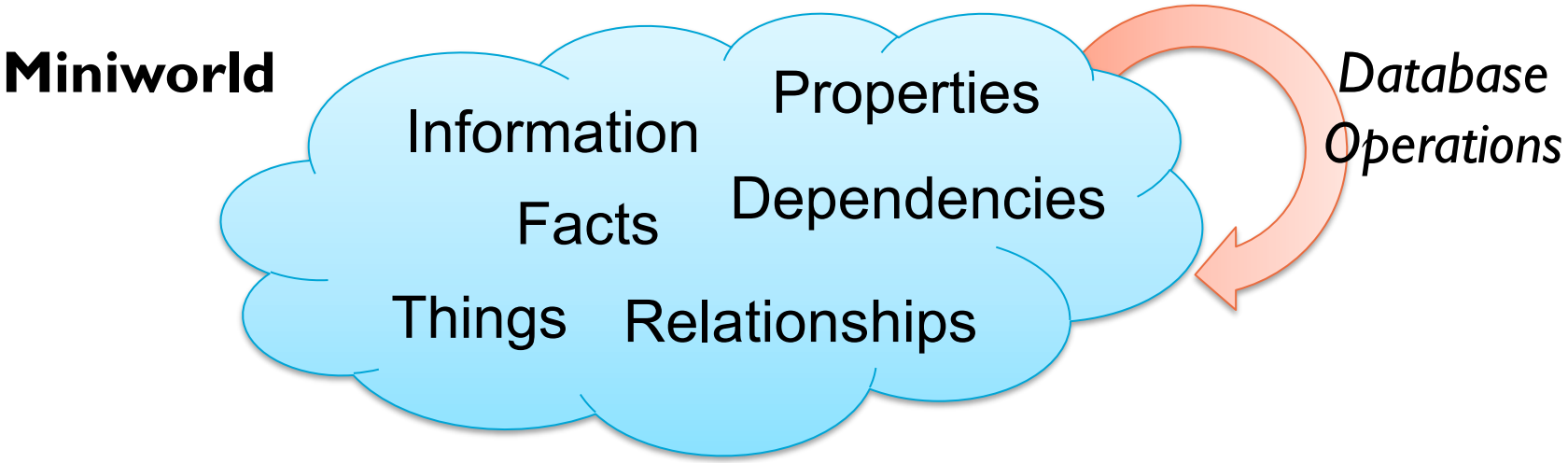


Towards Schema Design

- Conceptual Design
  - Focus on “what do we need to store and how are those things related”
    - Entity (type), Relationship (types), Functional Dependencies, etc.
- Logical Design
  - Adapt conceptual design to the Data Model used by our database
    - i.e., the “Type of Database”: Relational Databases use Tables, Document Databases use documents, Graph databases use graphs, etc.
- Physical Design
  - How to implement the logical design for this particular application/database instance
    - Where to store data, how to encode data, how to index data, etc.

Conceptual Design

- “what do we need to store and how are those things related”



Showcase: STUDYGUIDE

A Web Application we all know

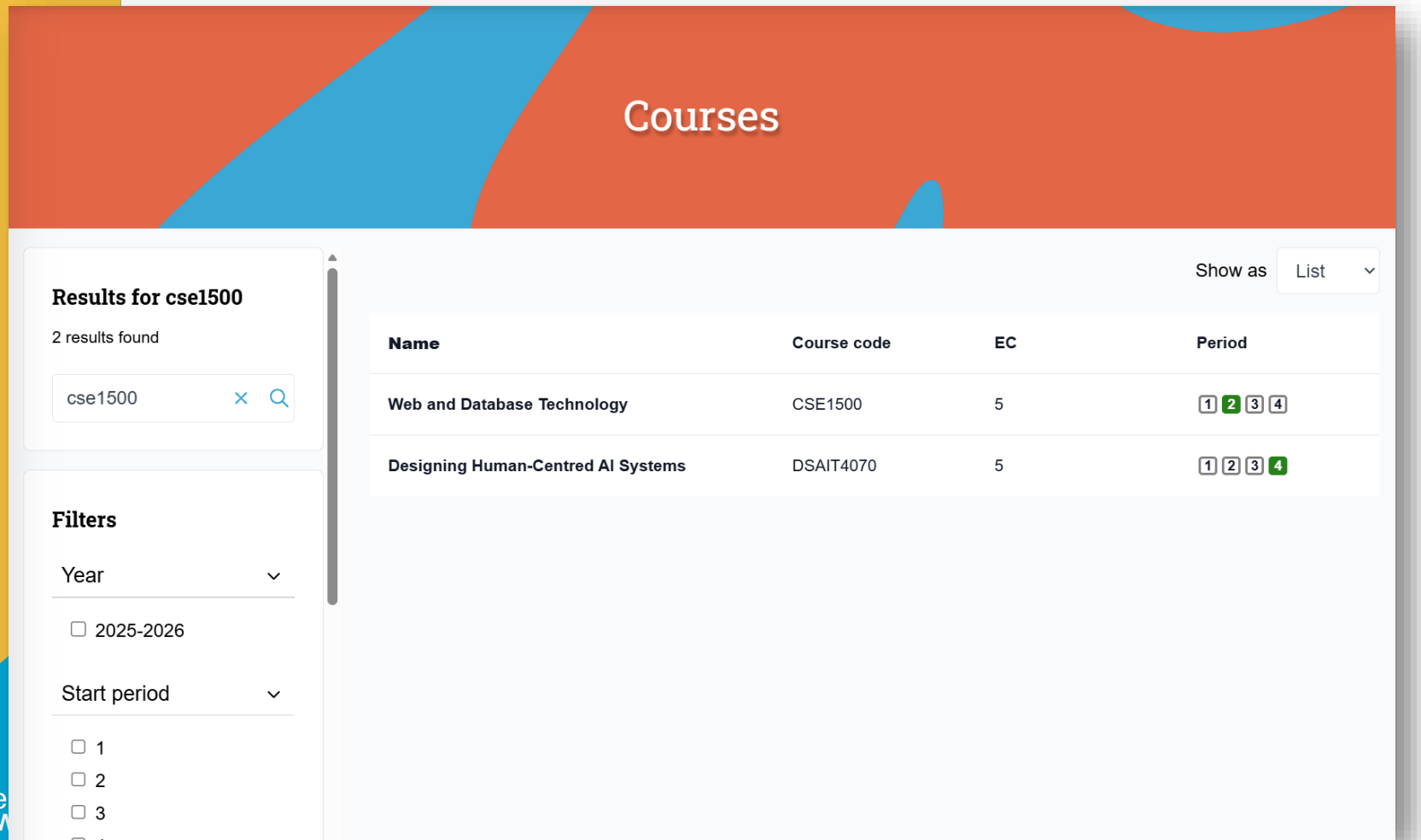
TU Delft WIS Web Information Systems

Studyguide

- https://www.studiegids.tudelft.nl/
  - As of November 2025
  - A Web application: **Web User Interface**, and a **Database** behind it
  - Let's assume we conceive or assess this software

Studyguide

- https://www.studiegids.tudelft.nl/
  - As of November 2025
  - A Web application: **Web User Frontend**, and a **Database backend** behind it



Studyguide

Home > Study Guide > Web and Database Technology

Web and Database Technology

CSE1500

Description

Web en Database Technologies is a first-year course that provides an introduction to web and database technologies and programming.

Topics related to web technology and programming include:

- Introduction to the web and HTTP;
- Introduction to web development and app design;
- Front-end development: HTML, CSS, JavaScript;
- Back-end development: Node.js.

Topics related to database technologies include:

- Introduction to databases;
- Overview of database Languages and architectures;
- The basic relational model; database design theory;
- SQL: data definition, constraints, updates, queries, views, triggers;
- Relational databases and object persistence;
- Introduction to non-relational database models: key-value stores, document stores, graph databases.

Learning objectives

Upon completion of this course, participants will have gained knowledge of web and database system concepts and the ability to:

1. Explain the basic architecture of the Internet and the Web
2. Analyse the requirements for a web application given a description of an application idea
3. Create an application front-end with HTML and CSS based on a given static design and justify design choices
4. Create an interactive web application with client-side JavaScript and justify design choices
5. Explain the elements of the Relational Model and contrast them with other Data Models
6. Design Conceptual and Logical Database Schemas and justify design choices;

Year
2025-2026

Course code
CSE1500

EC
5

Period
1 2 3 4

Level
Bachelor

Programme
BSc Computer Science and Engineering (Bachelor)
WO Bridging Programme MSc DSAIT for TUD BSc TW (Bridging)
WO Bridging Programme MSc DSAIT for TUD BSc EE (Bridging)
WO Bridging Programme MSc DSAIT for TUD BSc NB and LST (Bridging)
WO Bridging Programme MSc CS for TUD BSc TW (Bridging)
WO Bridging Programme MSc CS for TUD BSc EE (Bridging)

Faculty
Electrical Engineering, Mathematics and Computer Science

Responsible lecturer
Dr. C. Lofi, Dr.ir. U.K. Gadiraju

Teacher(s)
Dr. A. L. Power

Results for cse150

2 results found

cse1500

Filters

Year

2025-2026

Start period

1
2
3

Studyguide

1. Purpose and Goals

- What is the primary **purpose** of the application?
- Who are the main **users** (e.g., students, faculty, administrators)?
- What **problems** does it solve for these users?
- Who are the **stakeholders**?
- What are the **success criteria** for the application?

Problem Definition

- **Goal:** Understand the problem or need.
- **Activities:** Requirements gathering, stakeholder interviews, feasibility studies.
- **Output:** Clear problem statement and objectives.

Research and Analysis

- **Goal:** Explore existing solutions, technologies, and constraints.
- **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations.
- **Output:** Informed understanding of context and constraints.

Conceptual Design

- **Goal:** Generate possible solutions.
- **Activities:** Brainstorming, sketching, modeling, trade-off analysis.
- **Output:** Preliminary designs or concepts.

Detailed Design

- **Goal:** Develop the chosen concept into a complete design.
- **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping.
- **Output:** Final design specifications and documentation.

Implementation / Development

- **Goal:** Build or develop the solution.
- **Activities:** Manufacturing, coding, construction, integration.
- **Output:** Working product or system.

Studyguide

2. Core Features and Functionality

- What curriculum-related **data** needs to be managed (courses, modules, credits, prerequisites)?
  - e.g., Should it support multiple academic years or programs?
- What **operations** should users be able to perform (view, search, edit, compare curricula)?
  - Will it include scheduling, enrollment, or grading features?

Problem Definition	Research and Analysis	Conceptual Design	Detailed Design	Implementation / Development
• **Goal:** Understand the problem or need. • **Activities:** Requirements gathering, stakeholder interviews, feasibility studies. • **Output:** Clear problem statement and objectives.	• **Goal:** Explore existing solutions, technologies, and constraints. • **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations. • **Output:** Informed understanding of context and constraints.	• **Goal:** Generate possible solutions. • **Activities:** Brainstorming, sketching, modeling, trade-off analysis. • **Output:** Preliminary designs or concepts.	• **Goal:** Develop the chosen concept into a complete design. • **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping. • **Output:** Final design specifications and documentation.	• **Goal:** Build or develop the solution. • **Activities:** Manufacturing, coding, construction, integration. • **Output:** Working product or system.

Studyguide

🔗 3. Integration and Interoperability

- What **existing systems** (e.g., student information systems, LMS like Brightspace) does it need to integrate with?
- What **export/import data formats** and **APIs** are required for interoperability?

Problem Definition	Research and Analysis	Conceptual Design	Detailed Design	Implementation / Development
• **Goal:** Understand the problem or need. • **Activities:** Requirements gathering, stakeholder interviews, feasibility studies. • **Output:** Clear problem statement and objectives.	• **Goal:** Explore existing solutions, technologies, and constraints. • **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations. • **Output:** Informed understanding of context and constraints.	• **Goal:** Generate possible solutions. • **Activities:** Brainstorming, sketching, modeling, trade-off analysis. • **Output:** Preliminary designs or concepts.	• **Goal:** Develop the chosen concept into a complete design. • **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping. • **Output:** Final design specifications and documentation.	• **Goal:** Build or develop the solution. • **Activities:** Manufacturing, coding, construction, integration. • **Output:** Working product or system.

Studyguide

4. DATA: Information Architecture & Data Modelling

- What are the main **entities** (e.g., course, program, instructor) and their **relationships** relevant for the system?
- How should the **data be structured** and navigated?
- What other properties or metadata are needed (e.g., ECTS credits, course level, language)?

Problem Definition	Research and Analysis	Conceptual Design	Detailed Design	Implementation / Development
• **Goal:** Understand the problem or need. • **Activities:** Requirements gathering, stakeholder interviews, feasibility studies. • **Output:** Clear problem statement and objectives.	• **Goal:** Explore existing solutions, technologies, and constraints. • **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations. • **Output:** Informed understanding of context and constraints.	• **Goal:** Generate possible solutions. • **Activities:** Brainstorming, sketching, modeling, trade-off analysis. • **Output:** Preliminary designs or concepts.	• **Goal:** Develop the chosen concept into a complete design. • **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping. • **Output:** Final design specifications and documentation.	• **Goal:** Build or develop the solution. • **Activities:** Manufacturing, coding, construction, integration. • **Output:** Working product or system.

Web and Database Technology

CSE1500

Description

Web en Database Technologies is a first-year course that provides an introduction to web and database technologies and programming.

Topics related to web technology and programming include:

- Introduction to the web and HTTP;
- Introduction to web development and app design;
- Front-end development: HTML, CSS, JavaScript;
- Back-end development: Node.js.

Topics related to database technologies include:

- Introduction to databases;
- Overview of database Languages and architectures;
- The basic relational model; database design theory;
- SQL: data definition, constraints, updates, queries, views, triggers;
- Relational databases and object persistence;
- Introduction to non-relational database models: key-value stores, document stores, graph databases.

Learning objectives

Upon completion of this course, participants will have gained knowledge of web and database system concepts and the ability to:

1. Explain the basic architecture of the Internet and the Web
2. Analyse the requirements for a web application given a description of an application idea
3. Create an application front-end with HTML and CSS based on a given static design and justify design choices
4. Create an interactive web application with client-side JavaScript and justify design choices
5. Explain the elements of the Relational Model and contrast them with other Data Models
6. Design Conceptual and Logical Database Schemas and justify design choices;
7. Apply DB theory to improve Logical Schemas with focus on functional dependencies, keys, and database normalization, and justify design choices.
8. Design and analyse SQL queries
9. Prototype and deploy database applications using open-source database systems.

Year
2025-2026

Course code
CSE1500

EC
5

Period
1 2 3 4

Level
Bachelor

Programme
BSc Computer Science and Engineering (Bachelor)
WO Bridging Programme MSc DSAIT for TUD BSc TW (Bridging)
WO Bridging Programme MSc DSAIT for TUD BSc EE (Bridging)
WO Bridging Programme MSc DSAIT for TUD BSc NB and LST (Bridging)
WO Bridging Programme MSc CS for TUD BSc TW (Bridging)
WO Bridging Programme MSc CS for TUD BSc EE (Bridging)

Faculty
Electrical Engineering, Mathematics and Computer Science

Responsible lecturer
Dr. C. Lofi, Dr.ir. U.K. Gadiraju

Teacher(s)
Dr. A.J. Power

Location
Delft

Course language

Studyguide

5. WEB: Information Architecture & Data Modelling

- What are the **key user journeys** (e.g., finding a course, comparing programs, teacher assignment overview)?
- What **design**, **accessibility,** and **usability standards** should be followed?
  - Check our courses / the field of "**Human Computer Interaction**" (HCI)
- Should it support mobile devices? How responsive should it be?

Problem Definition	Research and Analysis	Conceptual Design	Detailed Design	Implementation / Development
• **Goal:** Understand the problem or need. • **Activities:** Requirements gathering, stakeholder interviews, feasibility studies. • **Output:** Clear problem statement and objectives.	• **Goal:** Explore existing solutions, technologies, and constraints. • **Activities:** Literature review, benchmarking, data analysis, prototyping, simulations. • **Output:** Informed understanding of context and constraints.	• **Goal:** Generate possible solutions. • **Activities:** Brainstorming, sketching, modeling, trade-off analysis. • **Output:** Preliminary designs or concepts.	• **Goal:** Develop the chosen concept into a complete design. • **Activities:** Data Diagrams, Software Architecture Diagrams, CAD modeling, material selection, calculations, prototyping. • **Output:** Final design specifications and documentation.	• **Goal:** Build or develop the solution. • **Activities:** Manufacturing, coding, construction, integration. • **Output:** Working product or system.

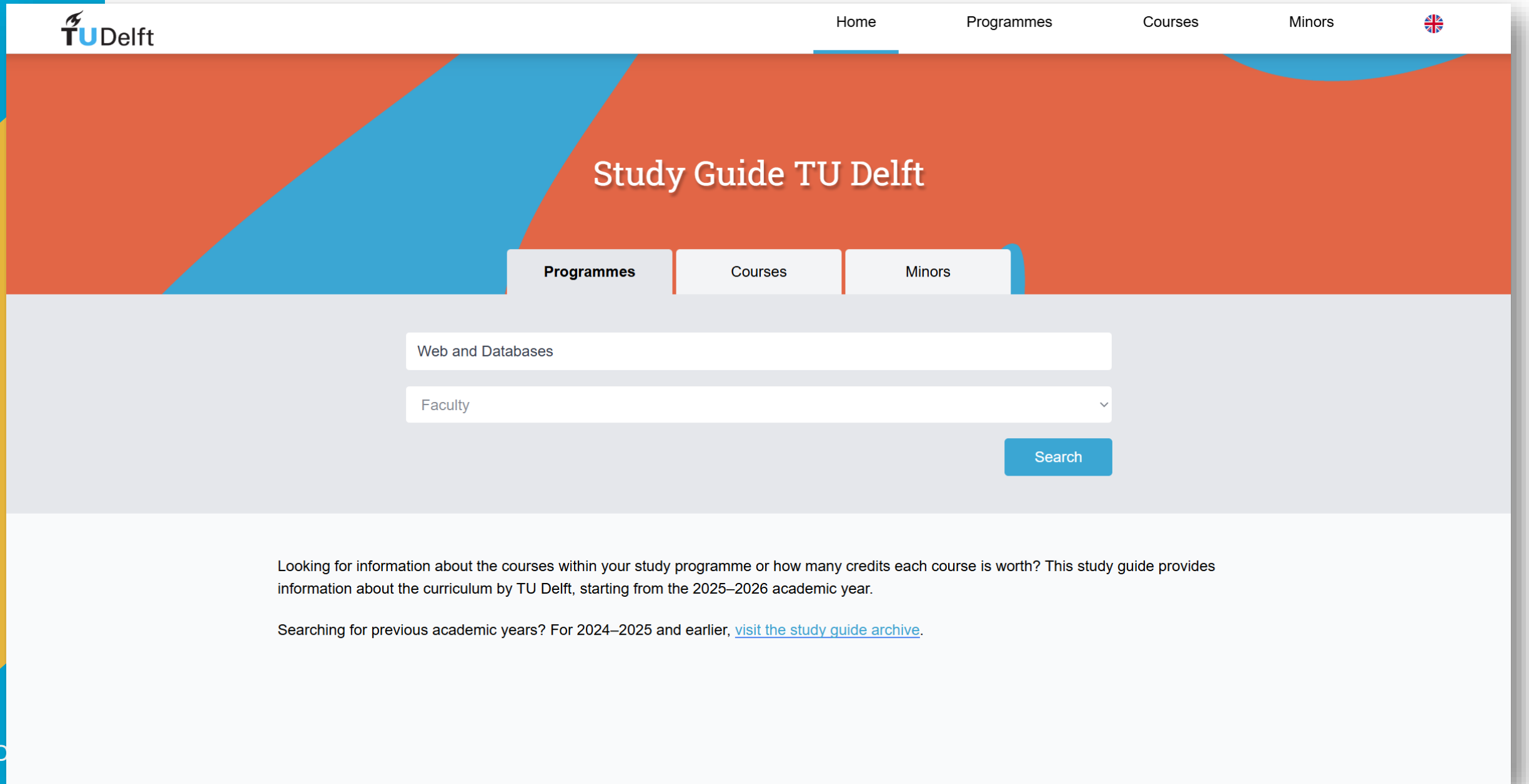
DETOUR

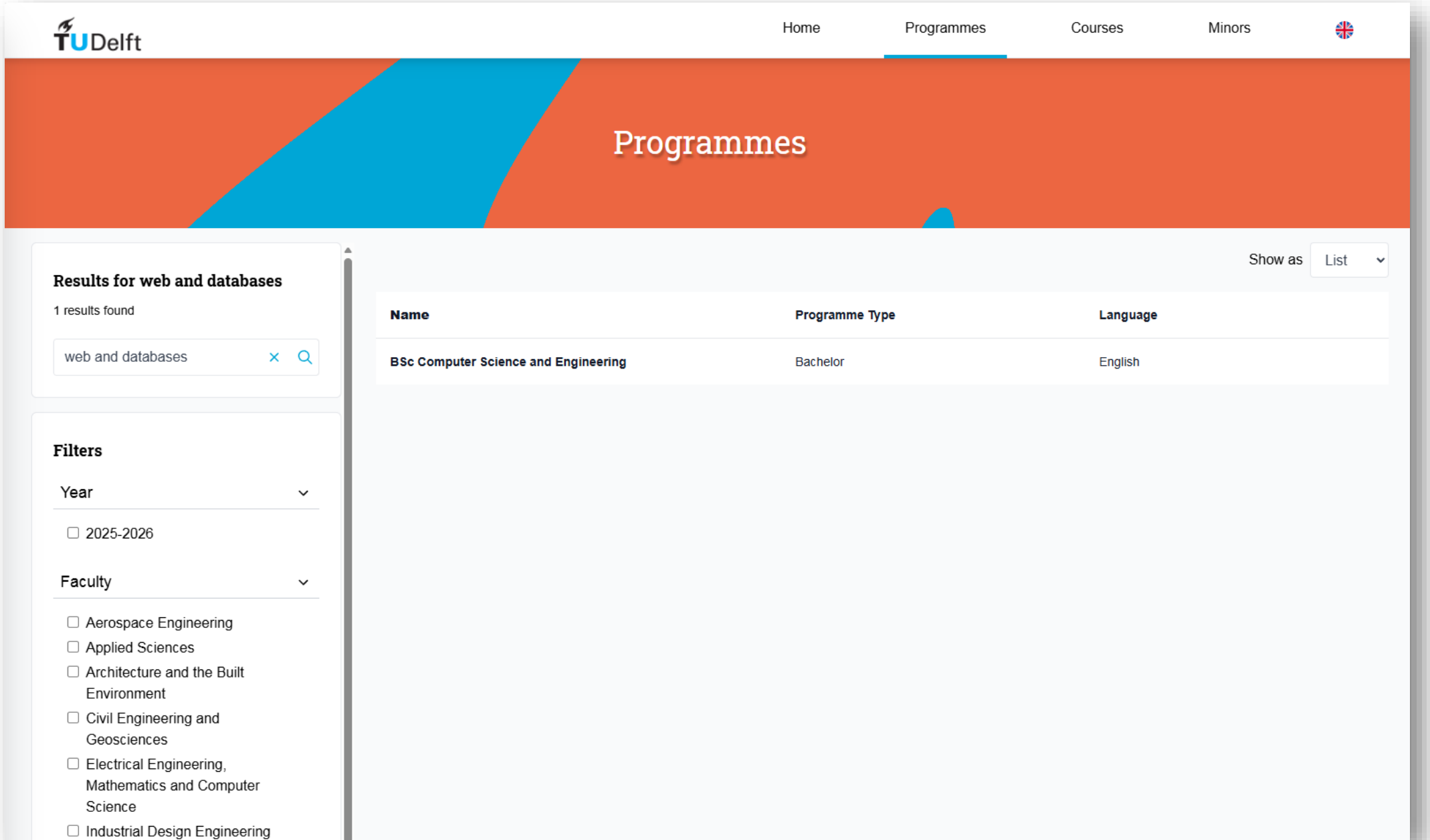
Studyguide

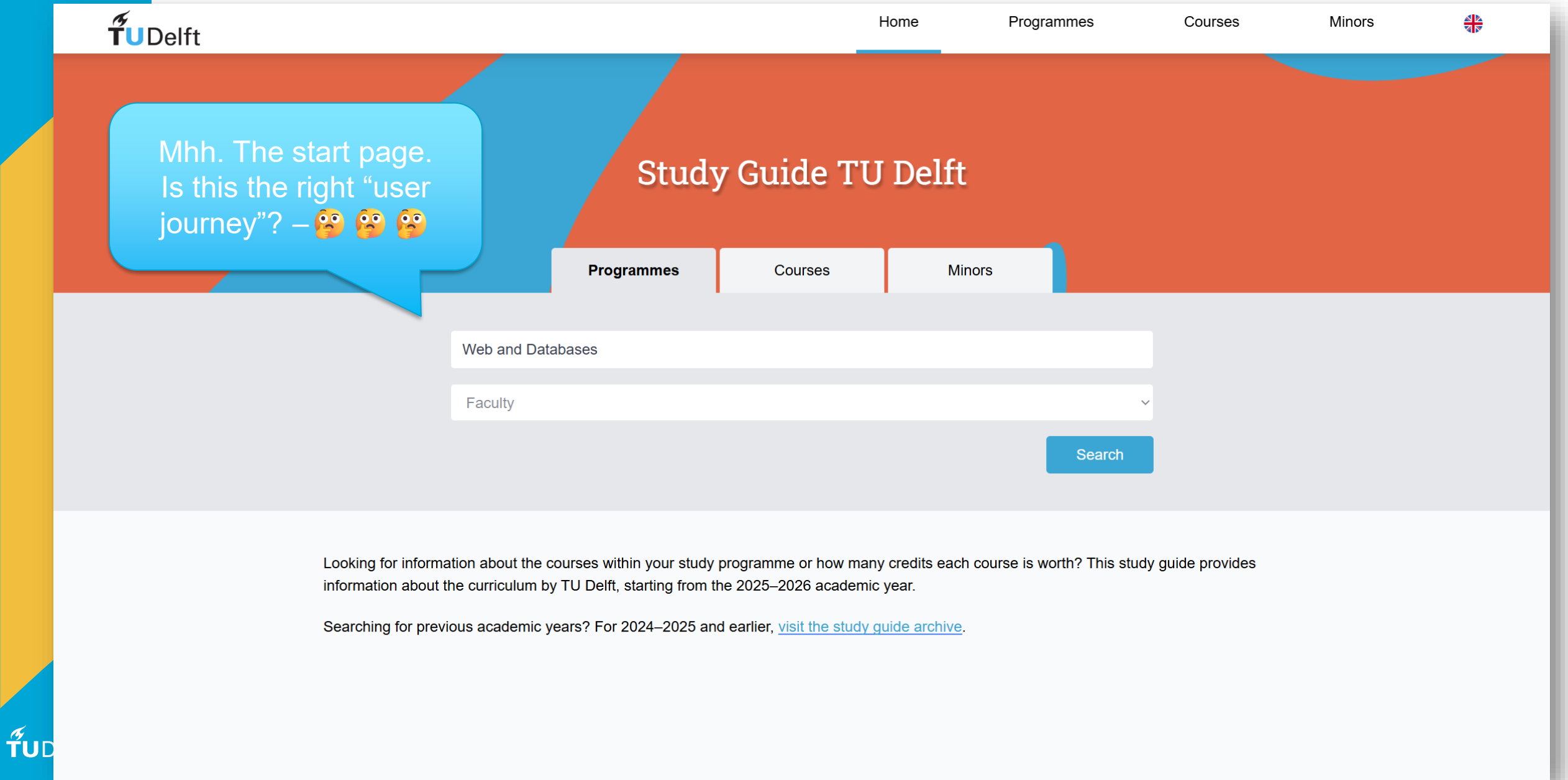
- Accessibility Standards Examples (beyond-the-scope reading)
- **Jakob Nielsen's 10 general principles for interaction design**
  - **Visibility of System Status**
    - Keep users informed about what's happening through timely and clear feedback.
  - **Match Between System and the Real World**
    - Use familiar language, concepts, and metaphors that align with users' expectations and real-world experiences.
  - **User Control and Freedom**
    - Provide easy ways to undo and redo actions. Let users exit unwanted states without hassle.
  - **Consistency and Standards**
    - Follow platform conventions and maintain consistency in terminology, layout, and behavior.
  - **Error Prevention**
    - Design systems to prevent problems before they occur—use constraints, confirmations, and clear instructions.
  - **Recognition Rather Than Recall**
    - Minimize memory load by making options, actions, and information visible and easily accessible.
  - **Flexibility and Efficiency of Use**
    - Support both novice and expert users by offering shortcuts, personalization, and adaptable workflows.
  - **Aesthetic and Minimalist Design**
    - Keep interfaces clean and focused. Avoid unnecessary elements that distract from core tasks.
  - **Help Users Recognize, Diagnose, and Recover from Errors**
    - Use plain language in error messages and provide constructive guidance to resolve issues.
  - **Help and Documentation**
    - Offer accessible, searchable help content—even if the system is easy to use—to support users when needed.

Studyguide

- Accessibility Standards Examples (beyond-the-scope reading)
- **WCAG (Web Content Accessibility Guidelines)**
- **Examples:**
  - **Use captions and transcripts** for audio and video content
    - **Ensure sufficient color contrast** between text and background (e.g., 4.5:1 for normal text)
    - **Avoid relying on color alone** to convey information (e.g., use icons or labels alongside color)
  - **Operable**
    - **Make all functionality available via keyboard** (e.g., tab navigation, skip links)
    - **Avoid content that flashes more than three times per second** to prevent seizures
    - **Provide clear focus indicators** for interactive elements like buttons and links
    - **Allow users to pause, stop, or adjust time-based media and animations**
  - **Understandable**
    - **Use clear and simple language** appropriate for your audience
    - **Provide meaningful link text** (e.g., “Download the report” instead of “Click here”)
    - **Ensure consistent navigation** across pages
    - **Offer error suggestions and instructions** when users input data incorrectly
  - Etc.


TU Delft
Home
Programmes
Courses
Minors
Study Guide TU Delft
Programmes
Courses
Minors
Web and Databases
Faculty
Search
Looking for information about the courses within your study programme or how many credits each course is worth? This study guide provides information about the curriculum by TU Delft, starting from the 2025–2026 academic year.
Searching for previous academic years? For 2024–2025 and earlier, visit the study guide archive.


TUDelft
Home
Programmes
Courses
Minors
Programmes
Results for web and databases
1 results found
web and databases
Filters
Year
2025-2026
Faculty
Aerospace Engineering
Applied Sciences
Architecture and the Built Environment
Civil Engineering and Geosciences
Electrical Engineering, Mathematics and Computer Science
Industrial Design Engineering
Show as
List
Name
Programme Type
Language
BSc Computer Science and Engineering
Bachelor
English


TU Delft
Home
Programmes
Courses
Minors
Mhh. The start page. Is this the right “user journey”? –🤔🤔🤔
Study Guide TU Delft
Programmes
Courses
Minors
Web and Databases
Faculty
Search
Looking for information about the courses within your study programme or how many credits each course is worth? This study guide provides information about the curriculum by TU Delft, starting from the 2025–2026 academic year.
Searching for previous academic years? For 2024–2025 and earlier, visit the study guide archive.

Web and Database Technology

CSE1500

Description

Web en Database Technologies is a first-year course that provides an introduction to web and database technologies and programming.

Topics related to web technology and programming include:

- Introduction to the web and HTTP;
- Introduction to web development and app design;
- Front-end development: HTML, CSS, JavaScript;
- Back-end development: Node.js.

Topics related to database technologies include:

- Introduction to databases;
- Overview of database Languages and architectures;
- The basic relational model; database design theory;
- SQL: data definition, constraints, updates, queries, views, triggers;
- Relational databases and object persistence;
- Introduction to non-relational database models: key-value stores, document stores, graph databases.

Learning objectives

Upon completion of this course, participants will have gained knowledge of web and database system concepts and the ab

1. Explain the basic architecture of the Internet and the Web
2. Analyse the requirements for a web application given a description of an application idea
3. Create an application front-end with HTML and CSS based on a given static design and justify design choices
4. Create an interactive web application with client-side JavaScript and justify design choices
5. Explain the elements of the Relational Model and contrast them with other Data Models
6. Design Conceptual and Logical Database Schemas and justify design choices;
7. Apply DB theory to improve Logical Schemas with focus on functional dependencies, keys, and database normalization, and justify design choices.
8. Design and analyse SQL queries
9. Prototype and deploy database applications using open-source database systems.

Year
2025-2026

Course code
CSE1500

EC
5

Period
1 2 3 4

Level
Bachelor

Programme
BSc Computer Science and Engineering (Bachelor)

WO Bridging Programme MSc DSAIT for TUD BSc TW (Bridging)

WO Bridging Programme MSc DSAIT for TUD BSc EE (Bridging)

WO Bridging Programme MSc DSAIT for TUD BSc NB and LST (Bridging)

WO Bridging Programme MSc CS for TUD BSc TW (Bridging)

ing Programme MSc CS for TUD BSc EE

Engineering, Mathematics and Computer Science

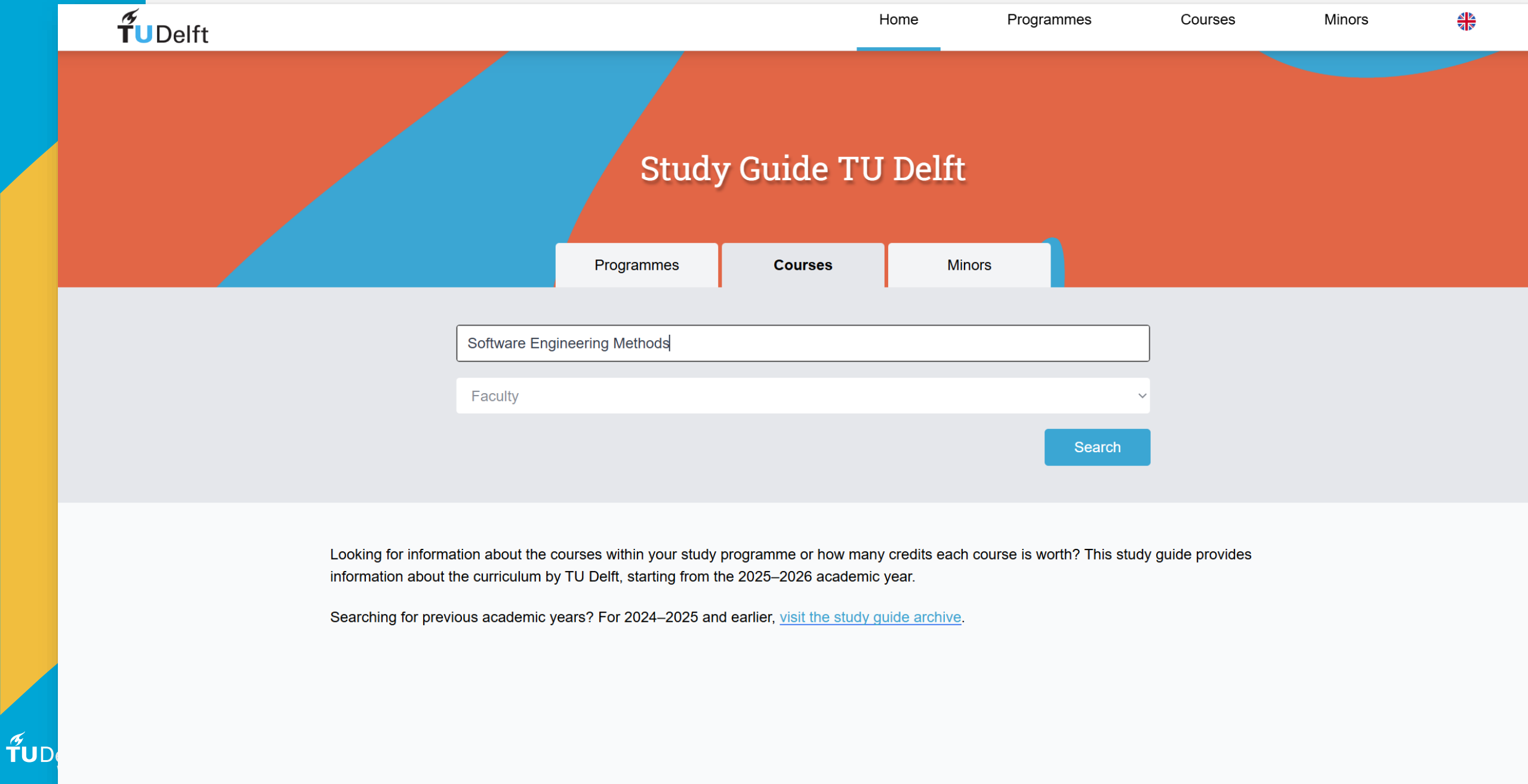
onsible lecturer
Dr. C. Lofi, Dr.ir. U.K. Gadiraju

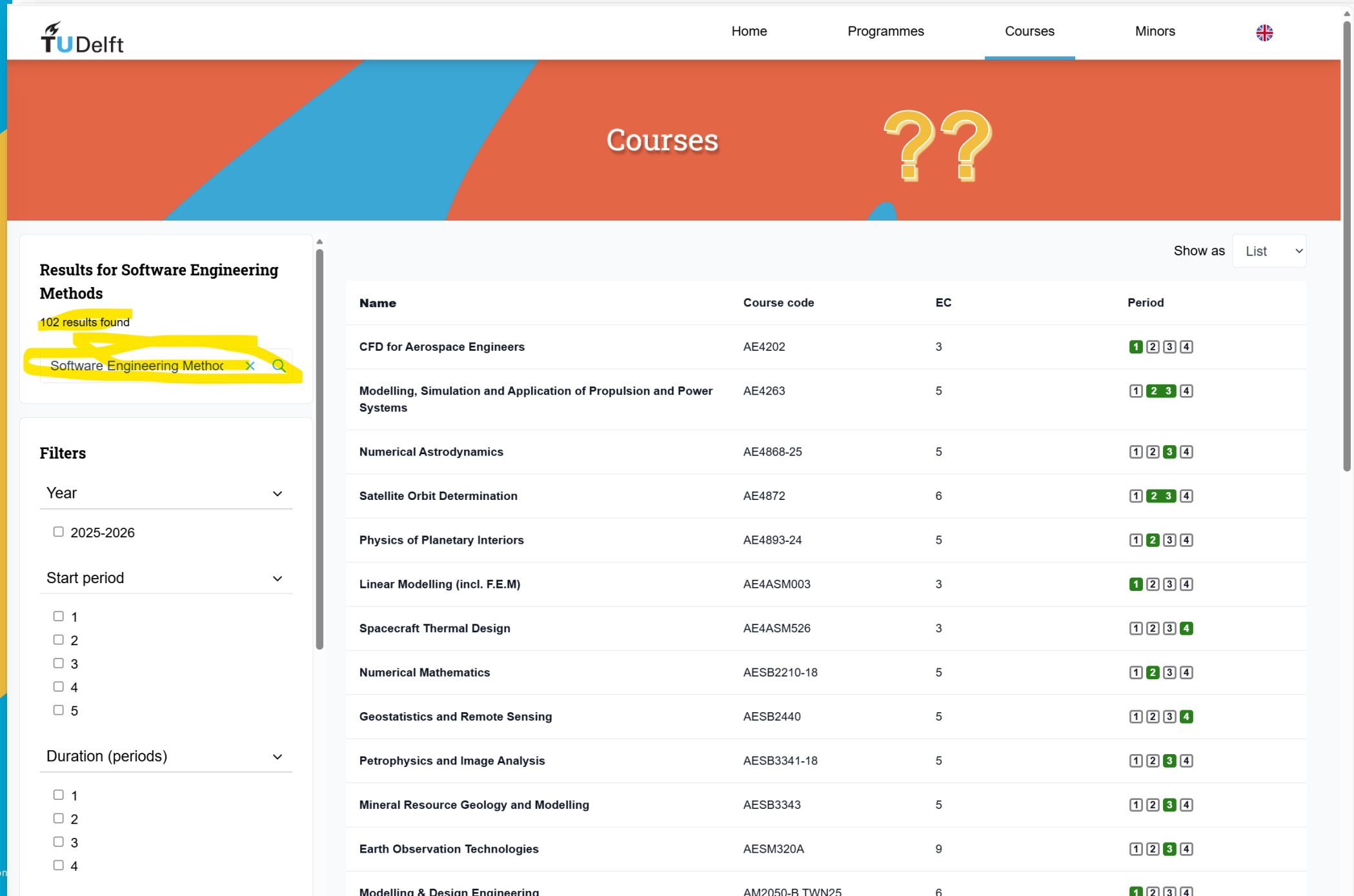
Teacher(s)
Dr. A.J. Power

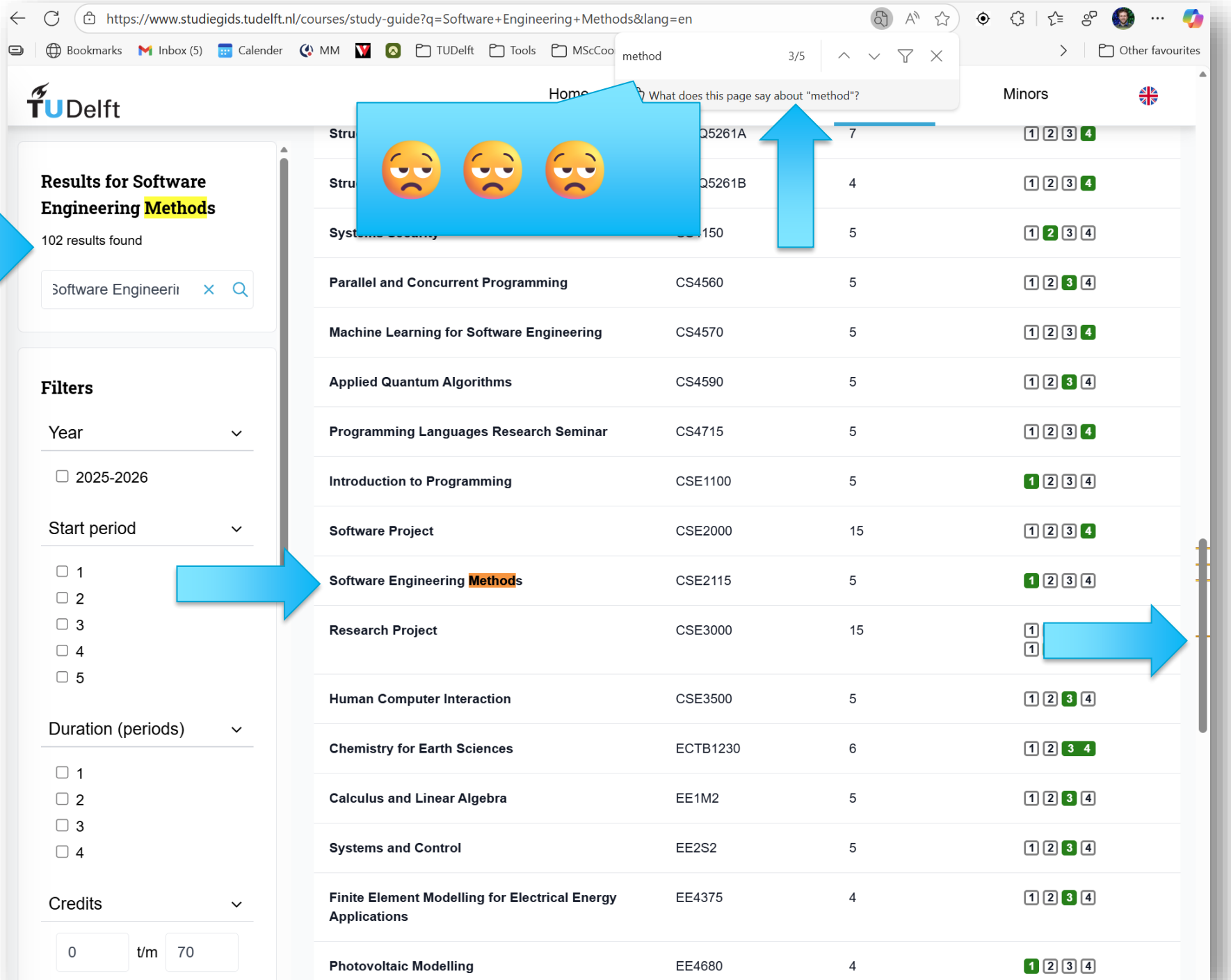
Location
Delft

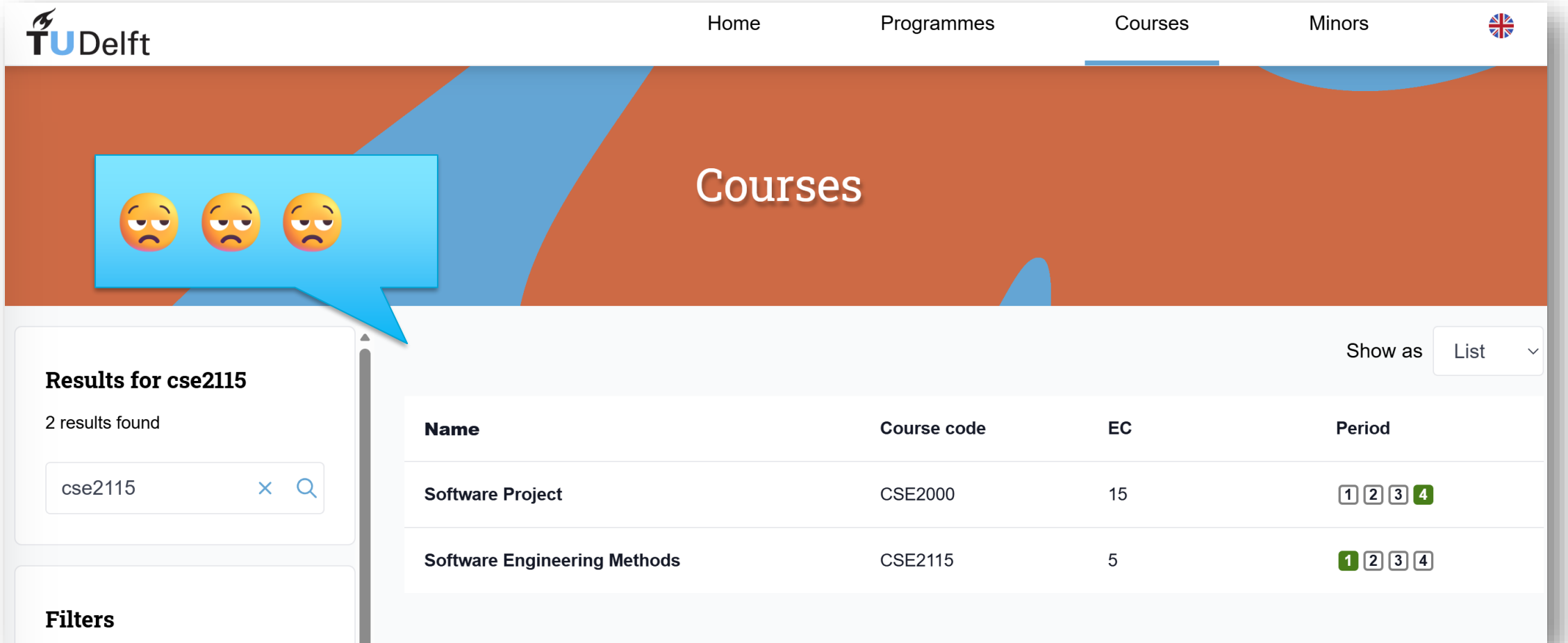
Course language

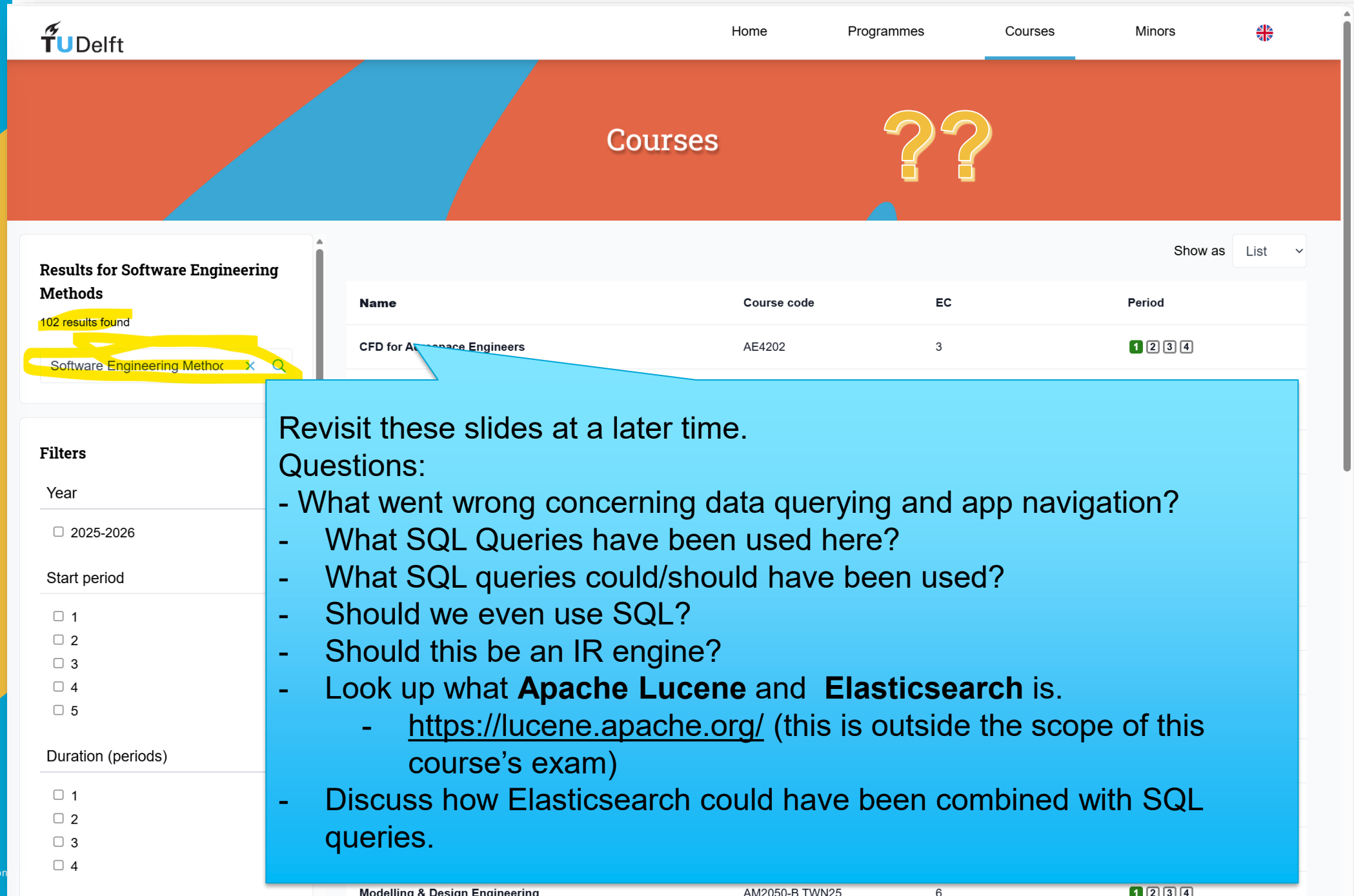
Funny. Clicking on this prepares an email..


TU Delft
Home
Programmes
Courses
Minors
Study Guide TU Delft
Programmes
Courses
Minors
Software Engineering Methods
Faculty
Search
Looking for information about the courses within your study programme or how many credits each course is worth? This study guide provides information about the curriculum by TU Delft, starting from the 2025–2026 academic year.
Searching for previous academic years? For 2024–2025 and earlier, visit the study guide archive.


TU Delft
Home
Programmes
Courses
Minors
Courses
??
Results for Software Engineering Methods
102 results found
Software Engineering Methoc
Filters
Year
2025-2026
Start period
1
2
3
4
5
Duration (periods)
1
2
3
4
Show as
List
Name
Course code
EC
Period
CFD for Aerospace Engineers
AE4202
3
Modelling, Simulation and Application of Propulsion and Power Systems
AE4263
5
Numerical Astrodynamics
AE4868-25
5
Satellite Orbit Determination
AE4872
6
Physics of Planetary Interiors
AE4893-24
5
Linear Modelling (incl. F.E.M)
AE4ASM003
3
Spacecraft Thermal Design
AE4ASM526
3
Numerical Mathematics
AESB2210-18
5
Geostatistics and Remote Sensing
AESB2440
5
Petrophysics and Image Analysis
AESB3341-18
5
Mineral Resource Geology and Modelling
AESB3343
5
Earth Observation Technologies
AESM320A
9


https://www.studiegids.tudelft.nl/courses/study-guide?q=Software+Engineering+Methods&lang=en
Bookmarks
Inbox (5)
Calender
MM
TUDelft
Tools
Other favourites
method
3/5
What does this page say about "method"?
TU Delft
Minors
Results for Software Engineering Methods
102 results found
Software Engineeri
Filters
Year
2025-2026
Start period
1
2
3
4
5
Duration (periods)
1
2
3
4
Credits
0
t/m
70
Parallel and Concurrent Programming CS4560 5
Machine Learning for Software Engineering CS4570 5
Applied Quantum Algorithms CS4590 5
Programming Languages Research Seminar CS4715 5
Introduction to Programming CSE1100 5
Software Project CSE2000 15
Software Engineering Methods CSE2115 5
Research Project CSE3000 15
Human Computer Interaction CSE3500 5
Chemistry for Earth Sciences ECTB1230 6
Calculus and Linear Algebra EE1M2 5
Systems and Control EE2S2 5
Finite Element Modelling for Electrical Energy Applications EE4375 4
Photovoltaic Modelling EE4680 4


TU Delft
Home
Programmes
Courses
Minors
Courses
Results for cse2115
2 results found
cse2115
Filters
Show as
List
Name
Course code
EC
Period
Software Project
CSE2000
15
1 2 3 4
Software Engineering Methods
CSE2115
5
1 2 3 4



Revisit these slides at a later time.
Questions:

- What went wrong concerning data querying and app navigation?
- What SQL Queries have been used here?
- What SQL queries could/should have been used?
- Should we even use SQL?
- Should this be an IR engine?
- Look up what **Apache Lucene** and **Elasticsearch** is.
  - https://lucene.apache.org/ (this is outside the scope of this course's exam)
- Discuss how Elasticsearch could have been combined with SQL queries.

Database Queries and Information Retrieval

- **Database Queries** (this course)
  - Declarative Query Languages like SQL
  - Return a **SET** of **EXACTLY MATCHIG** items
- **Information Retrieval** Queries (see courses on IR)
  - Keyword search, Natural Language Queries, etc
  - Return a **RANKED LIST** of the **MOST RELEVANT** items
  - Look up Apache Lucene, ElasticSearch, and/or pyTerrier

Other Questions….

🔐 7. Security and Privacy
- What data is sensitive (e.g., student records, instructor info)?
- What authentication and authorization mechanisms are needed?
- How will GDPR or other data protection regulations be addressed?

📈 8. Scalability and Performance
- How many users are expected to use the system concurrently?
- What performance benchmarks should be met (e.g., search speed)?
- Will it support internationalization or multi-language content?

🛠️ 9. Technical Constraints and Assumptions
- What technologies or platforms are preferred or required?
- Are there budget or time constraints?
- What legacy systems or data must be supported?

📊 10. Evaluation and Feedback
- How will the system be tested and validated?
- What mechanisms will be in place for user feedback?
- How will updates and improvements be managed post-launch?

Other Questions….

🔐 7. Security and Privacy
- What data is sensitive (e.g., student records, instructor info)?
- What authentication and authorization mechanisms are needed?
- How will GDPR or other data protection regulations be addressed?

📈 8. Scalability and Performance
- How many users are expected to use the system concurrently?
- What performance benchmarks should be met (e.g., search speed)?
- Will it support internationalization or multi-language content?

🛠️ 9. Technical Constraints and Assumptions
- What technologies or platforms are preferred or required?
- Are there budget or time constraints?
- What legacy systems or data must be supported?

📊 10. Evaluation and Feedback
- How will the system be tested and validated?
- What mechanisms will be in place for user fe
- How will updates and improvements be mana

Revisit all these slide at a later time.
Questions:
- Which of the previous questions might have been ignored by the studyguide developers?
- What advice would you give the dev team?

Generic Data Modelling

The Natural Language Approach (which is not often talked about)

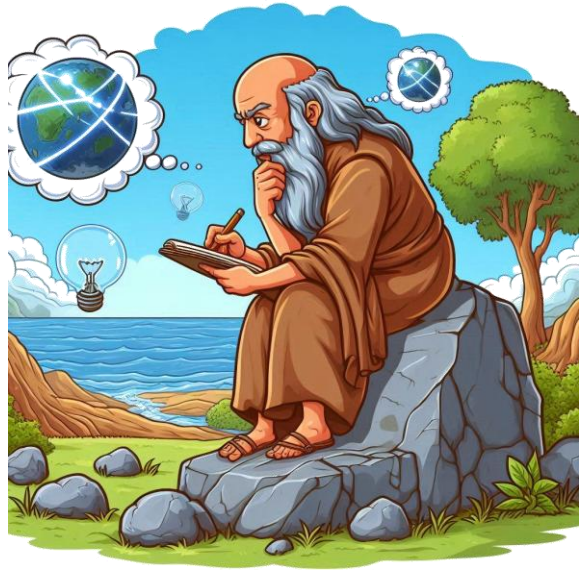
TU Delft WIS Web Information Systems

Generic Data Modelling

- For now: no specific technology limitations – let us just describe the world using normal language
  - Challenge: Let's focus on the **nature** of things – and not the specific semantics of formalisms or diagram types
    - In its early steps, modelling is **not depended on a chosen** technology!
  - Let's use natural language
    - First, let's define which words and concepts we want to use…
    - …and later match them to proper technologies

Entities, Things, Objects, Concepts

- **Thing / Entity**
  - Thing: “a separate and distinct individual quality, fact, idea, or usually entity”
  - Entity “something that has a separate and distinct existence and objective or conceptual reality”
  - That’s basically a thing: we will treat that synonymously
    - Thing==Entity
- **Object**
  - or: **objective entity**, **definite entity**
  - “something material that may be perceived by the senses”
    - “a physical thing”
- **Concept**
  - or: **conceptual entity**
  - “something conceived in the mind: a thought or notion”
    - “a more abstract thing”…



quotes from Merrian-Webster dictionary

Sets

- Sets: “Collection” of entities
  - Entities can be members of multiple sets
  - Sets are **designated** or defined in different ways
    - …**naming**: “the well-known set of natural numbers”
    - …**selection**: the set of all things that are in my bag, the set of all white elephants, etc.
    - …**enumeration**: {this pen, this knife, this lighter}
  - As such, destroying the notion of a set does usually not destroy the set itself
    - e.g., the set of natural numbers will remain even if we cease to recognize it

“Types”, “Classes”, “Kinds”

- A **type** is something that designates a set, members typically share some properties or functions
  - Also, “kind” or “class”
    - Technically, there are small differences between those terms though…
  - e.g., The (Entity) Type “Students” could describe the set of all/some entities which are students
  - e.g., The (Entity) Type “Temporary” could describe all entities I want to temporarily store (and then discard later)

“Relationship”

- “the way in which two or more things are connected, or the state of being connected.”
  - Oxford Dictionary
- “the way in which two things are connected”
  - Cambridge Dictionary
- “a connection, association, or involvement.”
  - Dictionary.com
- “a sexual involvement; affair.”
  - Also Dictionary.com. Ok, this is not really the definition we are looking for in general modelling ☺

Let's Play Through an Example

- Task: Modelling Students, Courses and Exams
- Focus on
  - **Entities** (or technically, the Types of Entities)
    - What attributes or properties do they have? Which attributes have mutable/changeable values? Which ones are relevant to us?
    - What identity do these entities have?
  - **Relationships** between Entities (between Entity Types)

Modelling the World – Example Part 1

- Students

This was a live classroom example. The slides are meaningless without the discussion in class.

Modelling the World – Example Part 1

- Courses

This was a live classroom example. The slides are meaningless without the discussion in class.

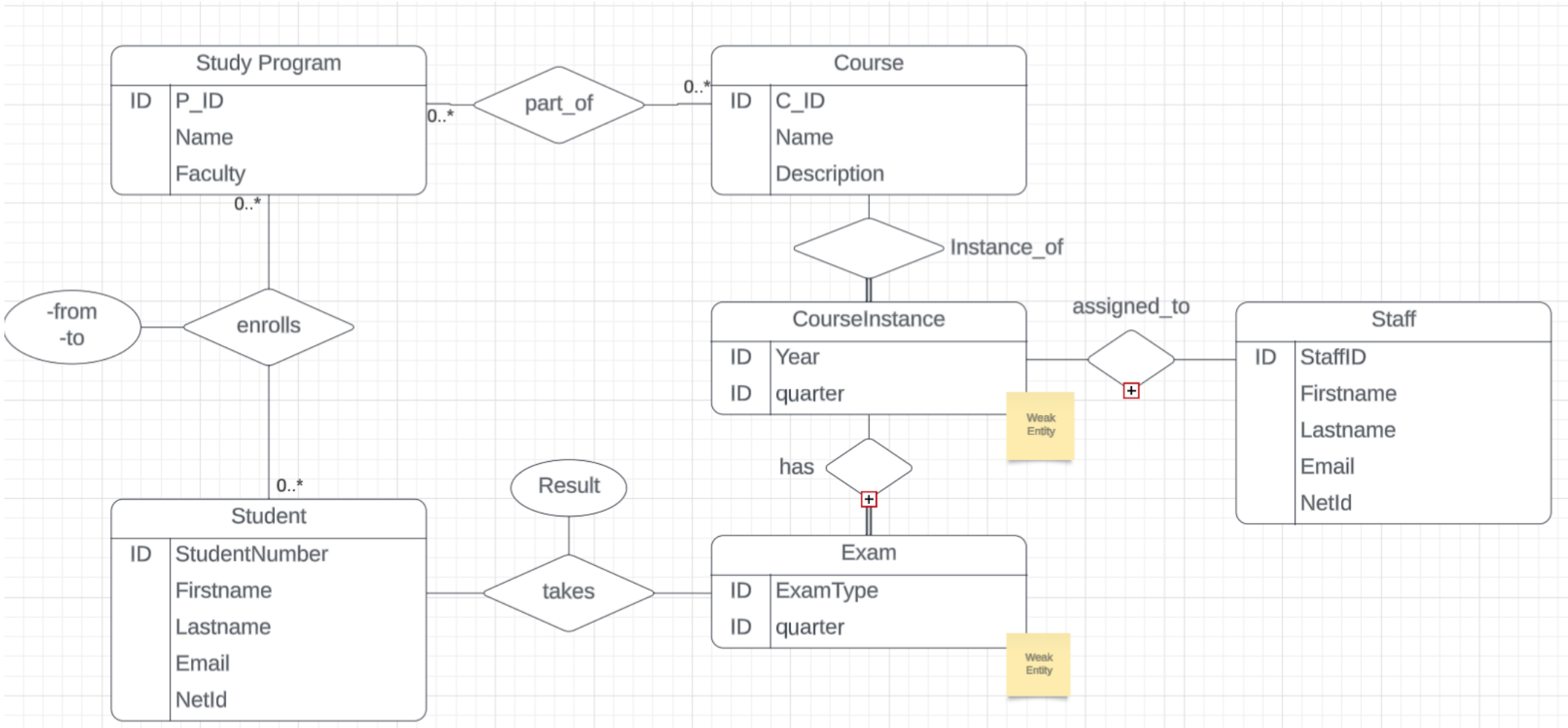
Modelling the World – Example Part 1

- Exams??

This was a live classroom example. The slides are meaningless without the discussion in class.

Modelling the World – Example Part 1

This was a live classroom example. The slides are meaningless without the discussion in class.



@TODO, UNFINISHED!!!!!! Something like this...

Preview Discussion: Patient Identifiers

- “You design a management system for an Emergency Clinic.
  How should you identify patients?
  Is the Dutch BSN a good identifier?
  Discuss, argue, and decide.”

Towards Relational DBs

TU Delft WIS Web Information Systems

What are Data Model Theories?

- A **Data Model (Theory)** is an abstract model that describes how data is represented, accessed, and reasoned about
  - **Warning:** The term "**data model**" is ambiguous
    - A data model **theory** is a formal description of how data may be structured and accessed and is independent of a specific software or hardware.
    - A data model **instance** or **schema** applies a data model theory to create an instance for some application (e.g., data models in MySQL Workbench designer refer to a logical model adapted to the MySQL database).
    - This will haunt us a lot…and it's nobody's fault really.
    - I try to use **data model (theory)** and **schema** during this course to lower the confusion…
  - We re-visit this later!

Data Model Theories

- There are several common data models
  - **Relational Model**
  - Key-Value Model
  - Document-centered Model
  - Graph Model
  - Etc.

Relational Model Example

- Core Idea: We store data in Tables!!

movie

m_id	title	year
1	Terminator 2 – Judgement Day	1991
2	Twins	1988

Primary key: m_id

actor

a_id	f_name	l_name
1	Arnold	Schwarzenegger
2	Linda	Hamilton
3	Danny	DeVito

Primary key: a_id

cast

m_id	a_id	role
1	1	T-800
1	2	Sarah Connor
2	1	Julius Benedict
2	3	Vincent Benedict

Primary key: (m_id, a_id)
- Note: This allows each actor to only have one role per movie. Primary key should be (m_id, a_id, role) to allow multiple roles per movie.

Foreign key constraints: (m_id-> movie.m_id, a_id-> actor.a_id)

Relational Model Example

Columns (or also “attribute”)

movie

m_id	title	year
1	Terminator 2 – Judgement Day	1991
2	Twins	1988

actor

a_id	f_name	l_name
1	Arnold	Schwarzenegger
2	Linda	Hamilton
3	Danny	DeVito

cast

m_id	a_id	role
1	1	T-800
1	2	Sarah Connor
2	1	Julius Benedict
2	3	Vincent Benedict

Rows
(or also “tuples”)

Table (or also: “Relation”)

Relational Model Example

movie

m_id	title	year
1	Terminator 2 – Judgement Day	1991
2	Twins	1988

Primary key: m_id

actor

a_id	f_name	l_name
1	Arnold	Schwarzenegger
2	Linda	Hamilton
3	Danny	DeVito

Primary key: a_id

Foreign Key Constraint
(just the graphical representation of it)

cast

m_id	a_id	role
1	1	T-800
1	2	Sarah Connor
2	1	Julius Benedict
2	3	Vincent Benedict

Primary key: (m_id, a_id)
- Note for later: This allows each actor to only have one role per movie. Primary key should be (m_id, a_id, role) to allow multiple roles per movie.
- Foreign key constraints: (m_id-> movie.m_id, a_id-> actor.a_id)

Document Model Example

Movie-Centered Documents

```
"terminator2":

{ "title" : "Terminator 2 : Judgement Day",
  "year"  : 1991,
  "director" : {
    "first_name" : "James",
    "last_name"  : "Cameron" }
  "actors": [
    {"first_name" : "Arnold", "last_name" : "Schwarzenegger" },
    {"first_name" : "Linda", "last_name" : "Hamilton" },
    {"first_name" : "Edward", "last_name" : "Furlong" }
  ]
}
```

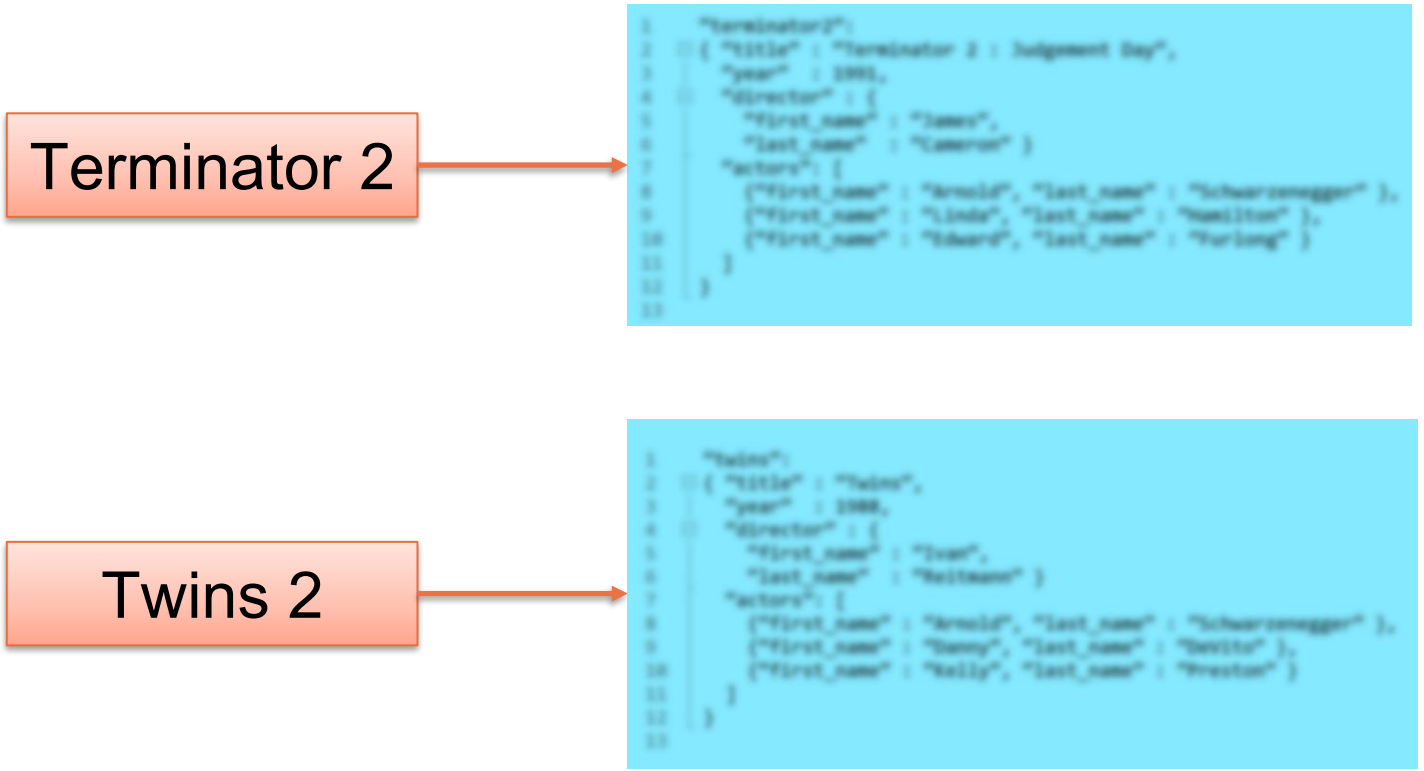
```
"twins":
{ "title" : "Twins",
  "year"  : 1988,
  "director" : {
    "first_name" : "Ivan",
    "last_name"  : "Reitmann" }
  "actors": [
    {"first_name" : "Arnold", "last_name" : "Schwarzenegger" },
    {"first_name" : "Danny", "last_name" : "DeVito" },
    {"first_name" : "Kelly", "last_name" : "Preston" }
  ]
}
```

Actor-Centered Documents

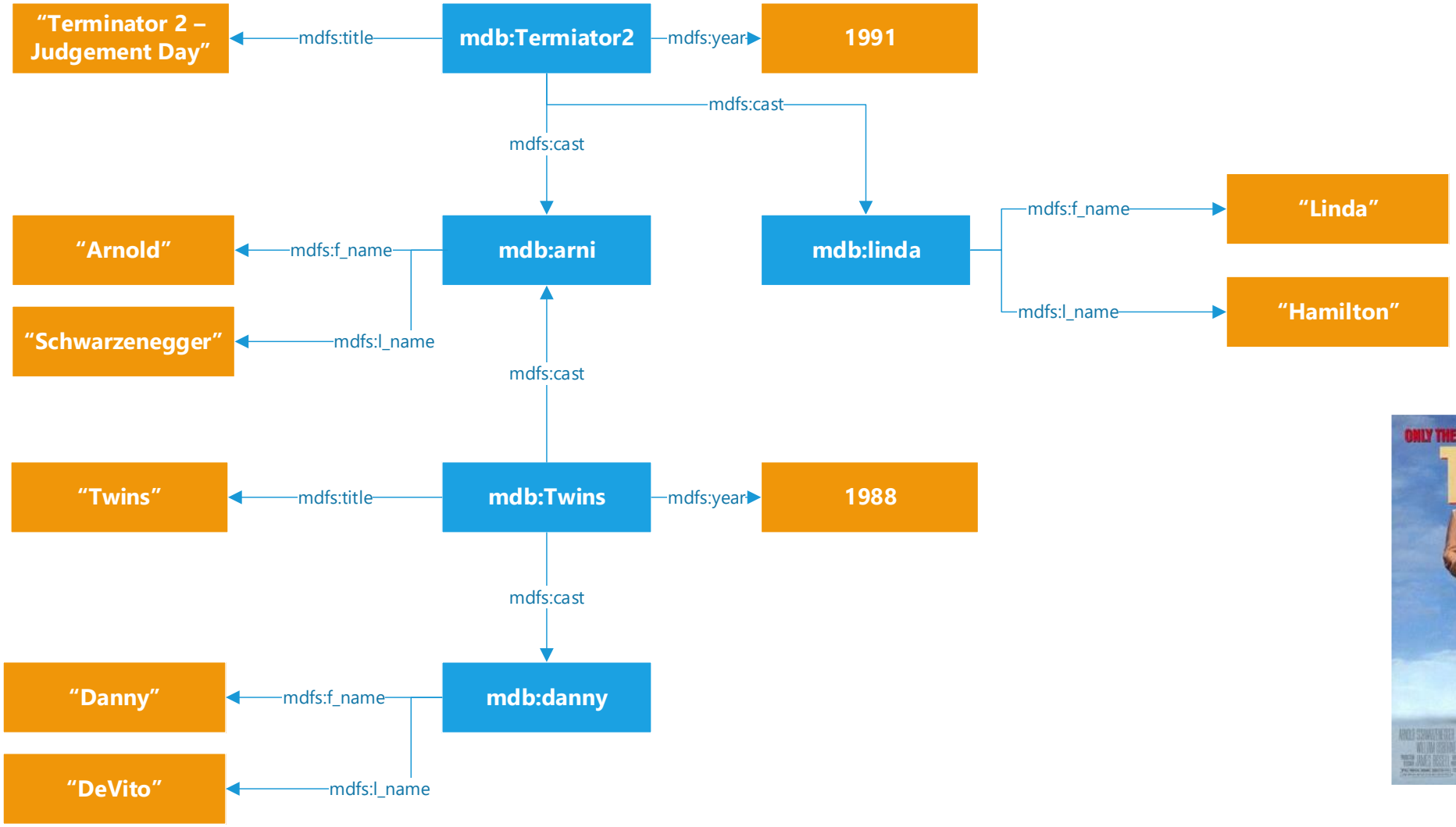
```
"arni":
{ "first_name" : "Arnold",
  "last_name" : "Schwarzenegger",
  "movies" : [
    {
       "title" : "The Terminator 2",
       "year"  : "1991" }
    {  "title" : "Twins",
       "year"  : "1988" }
  ]
}
```

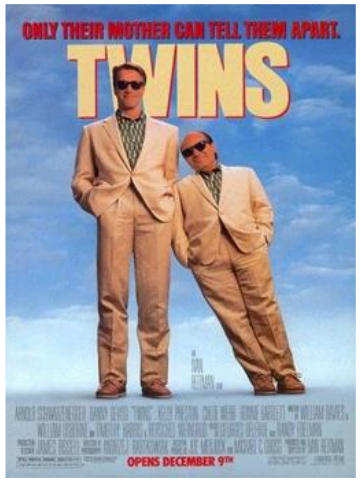
```
"linda":
{ "first_name" : "Linda",
  "last_name" : "Hamilton",
  "movies" : [
    {
       "title" : "The Terminator 2",
       "year"  : "1991" }
  ]
}
```

Key-Value Store Example



Graph Model Example





Data Models

- A **Data Model** describes data objects, operations and their effects
- Data Definition Language (**DDL**)
  - How to structure data
- Data Manipulation Language (**DML**)
  - How to add, edit, delete data
  - **DML** and **DDL** are usually clearly separated, since they handle **data** and **meta-data**, respectively

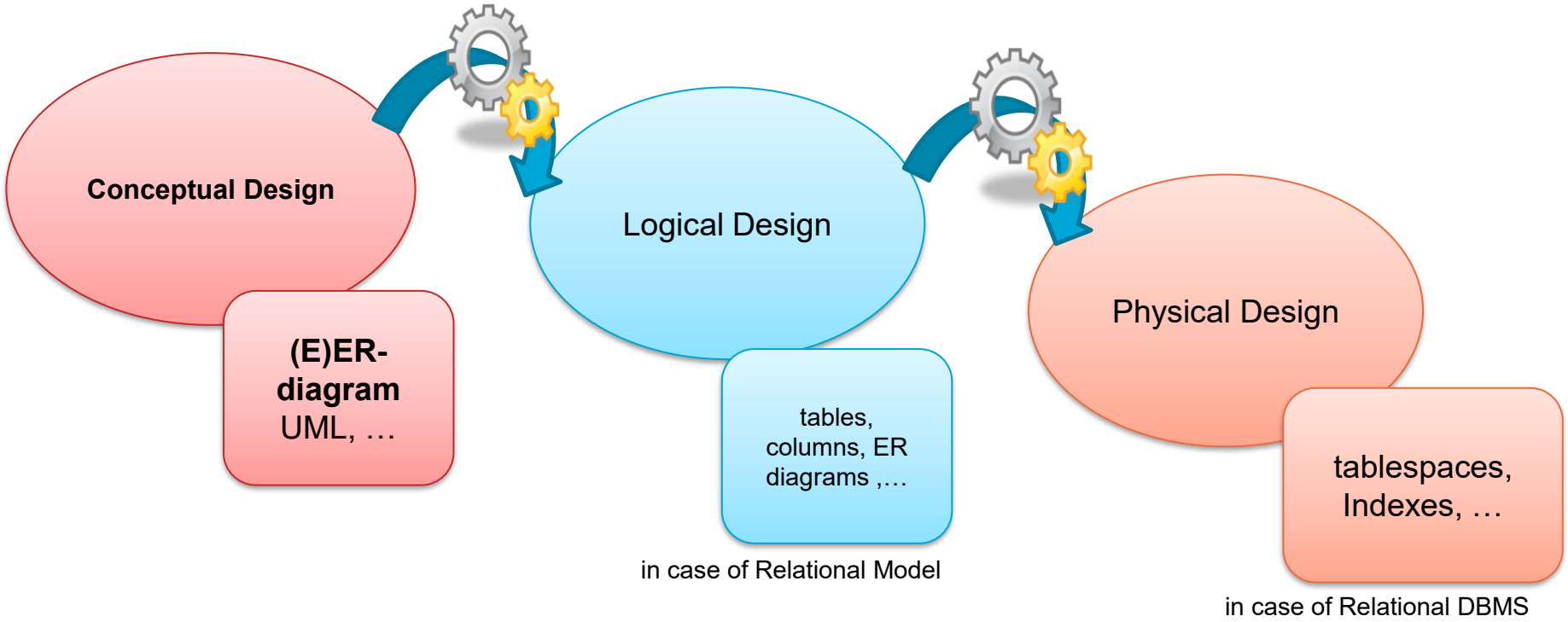
Schemas and Instances

- Later in modeling, we will again introduce 3 layers of schemas
  - Schemas abstract the data to be stored about the real world
  - **Conceptual Schema**
    - What entities are important, and what are their relationships?
      - "I want to store movies and actors. Each movie can have multiple actors. "
  - **Logical Schema**
    - Transforms the conceptual schema with respect to the chosen Data Model
    - Relational Logical Schema: Describes tables and their attributes, keys and constraints
      - "Actor(id, firstname, lastname);
        Movie(id, title, year);
        a2m(a_id->Actor(id), m_id->Movie(id), role)"
  - **Physical Schema**
    - Describes HOW data is organized on disks

Informal Summary

- **Database**: “A collection of related data represented (using a data model and a defined data schema).”
- **Database Management System** (DBMS): “A software system managing and maintaining a database.”
- **Data Model**: “A formal definition on how to represent data (in general) and the available data operations.”
- **Data Schema**: “A definition of the structure of a specific database.”

Summary


Conceptual Design
(E)ER-diagram
UML, …
Logical Design
tables, columns, ER diagrams ,…
in case of Relational Model
Physical Design
tablespaces, Indexes, …
in case of Relational DBMS

Naming Conventions

Data Models

- Repeat: A **data model (theory)** is an abstract model that describes how data is represented, accessed, and reasoned about
  - e.g., network model, relational model, object-oriented model, document-centric model
  - **Warning:** The term "**data model**" is ambiguous
    - A data model **theory** is a formal description of how data may be structured and accessed and is independent of a specific software or hardware.
    - A data model **instance** or **schema** applies a data model theory to create an instance for some application (e.g., data models in MySQL Workbench designer refer to a logical model adapted to the MySQL database).

Data Models

Synonym or Homonym??

- A word of warning:
  - When data modelling is concerned, **nomenclature** is one of your core enemies!
    - Many terms are re-used to describe completely different concepts
      - Data Model == Data Model?
      - Entity == Entity Type?
      - Logical Schema == Physical Schema?
      - etc.
    - The same concept can be known under different terms
      - Especially across different modelling techniques like ER, UML, etc.

Data Models / Schema

- Data Models can be used to design **schemas**
  - A schema describes a particular scenario / instance / miniworld using the capabilities and expressiveness of the chosen data model
  - Many people call schemas also “data models”
    - … which is not a bad name per se, a schema “models” the data in a given database
    - But for us, let’s use “data model == a data model theory”, and “schema == a description of the structure of a given database using a specific data model”

“data modelling”: Naming Confusion

- Careful with “Data Model” vs data model… !
  - I repeat again: This is not an issue of we define terms in this course, this is an issue of how people use these terms in the world!
  - Data Models as in “real” Data Models
    - “How can data be organized, stored, modified, and retrieved in systems from general point of view”
    - **Relational Model**
    - Network Model
    - Document Model
    - Graph Model
    - etc.
  - Data Models aka “schemas”
    - How is data organized in this particular system/application
    - Conceptual Schemas
    - Logical Schemas (which people sometimes call physical schemas)
    - Physical Schemas (which the same people who call logical schemas physical schemas still call physical schemas…)

See you in the Tutorial!

ER Diagrams!

This chapter on ER modelling is intended for home study AND will be covered in the Tutorial!

Please read this at home, maybe also with some sections from the DB book and other resources.

TU Delft WIS Web Information Systems

Recap “Relationship”

- “the way in which two or more things are connected, or the state of being connected.”
  - Oxford Dictionary
- “the way in which two things are connected”
  - Cambridge Dictionary
- “a connection, association, or involvement.”
  - Dictionary.com
- “a sexual involvement; affair.”
  - Also Dictionary.com. Ok, this is not really the definition we are looking for in general modelling ☺

Summary ER Modelling

- Traditional approach to **Conceptual Modeling**
  - **Entity-Relationship Models** (ER-Models)
    - Also known as Entity-Relationship Diagrams (ERD)
    - Introduced in 1976 by **Peter Chen**
    - Graphical representation
- Top-Down-Approach for modeling
  - Entities and Attributes
  - Relationships
  - Constraints
- Some derivates became popular
  - ER Crow's Foot Notation (Bachman Notation)
  - ER Baker Notation
  - Later: Unified Modeling Language (UML)

The Entity-Relationship Diagrams

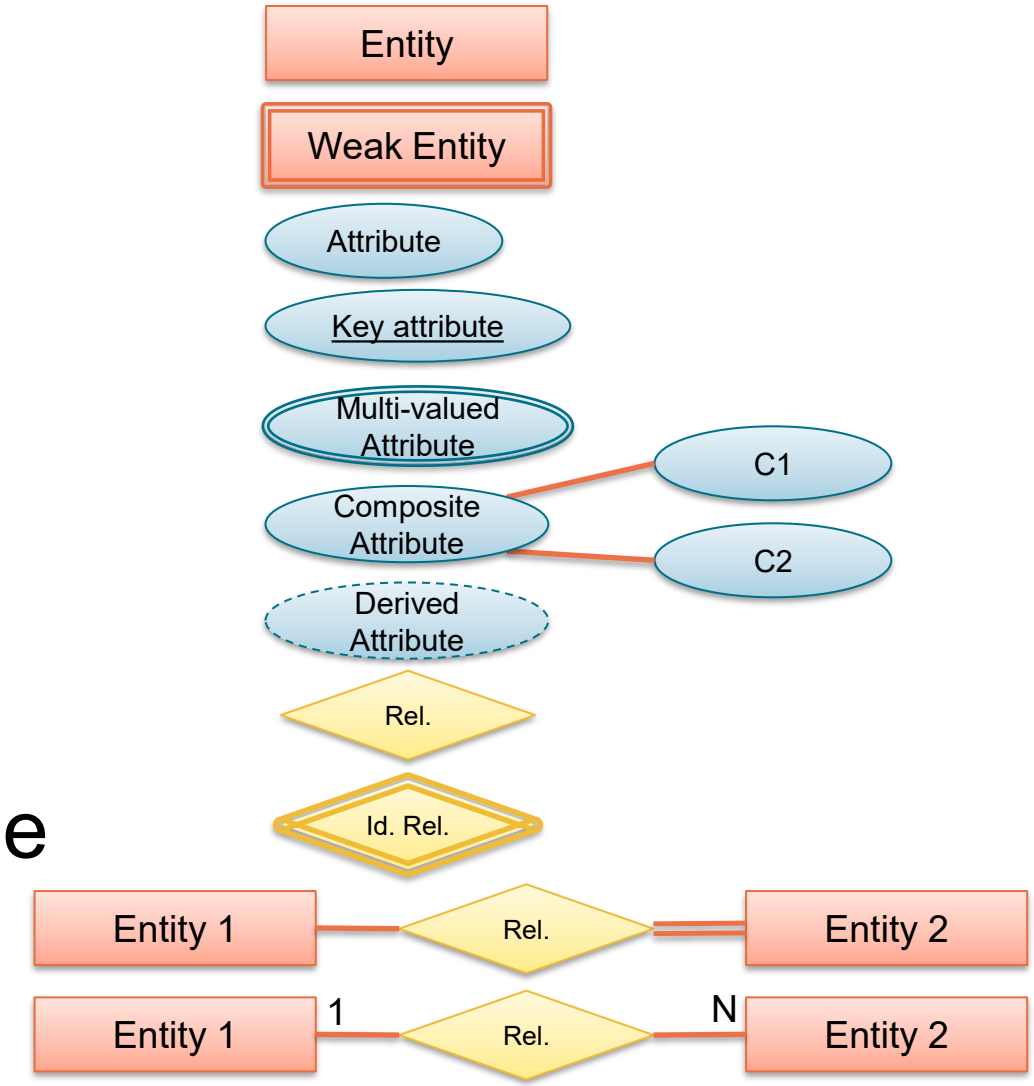
- The most used conceptual data diagramming style
- Provides a series of constructs capable of describing the data requirements of an application:
  - In a way that is easy to understand
  - Using a graphical formalisms
  - Independently from the database system of choice
- For every construct, there is a corresponding graphical representation
  - This representation allows us to define an ER schema diagrammatically

Summary ER Modelling

- ER Models “entities” and their “relationships”
- Entities:
  - Well, this is a bad name. It represents entity types
    - In ontology, an entity is a “thing”
  - A logical record type:
    - A type designating the set of possible tuples of a relation which is given by the Cartesian product of the attribute types
  - A relation
    - Mathematical representation of a table
  - The data stored according to the logical record type:
    - The set of actual tuples stored in a DB
  - A real-world entity type:
    - For example, the box with “person” implicitly relates to all real-world persons in our miniworlds
  - In the following, we will call these boxes entity types!
    - This is NOT how ER models usually do it …, but usually an “entity” should denote an instance of the set denoted by a type!!

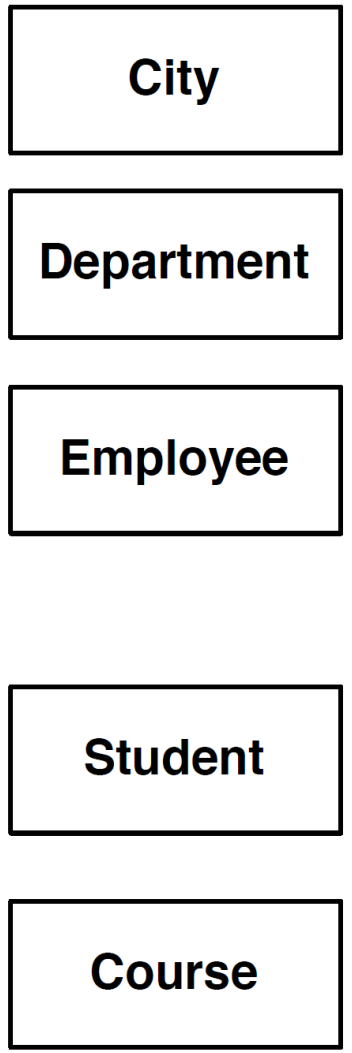
Summary ER Modelling

- Entity (Type)
- Weak Entity (Type)
- Attribute
- Key Attribute
- Multi-valued Attribute
- Composite Attribute
- Derived Attribute
- Relationship Type
- Identifying Relationship Type
- Total Participation
- 1:N Cardinality



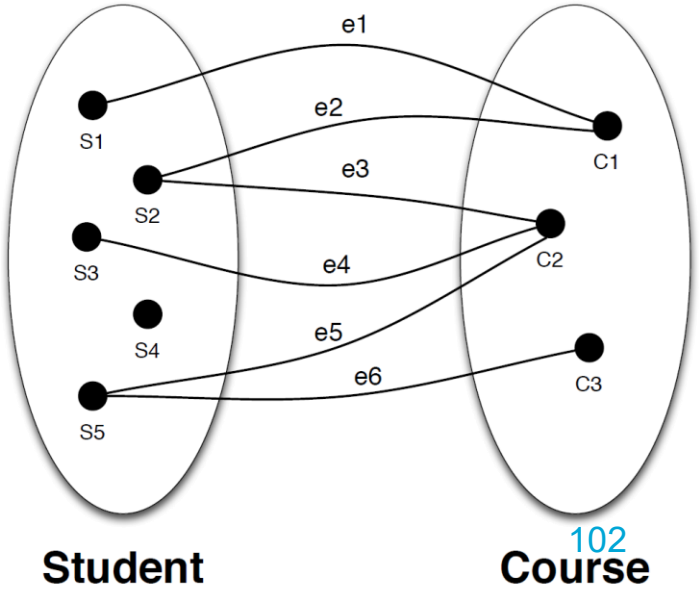
Entity Types

- Classes of objects (e.g., facts, things, people) having:
  - Common properties
  - Autonomous existence
- Examples:
  - Commercial organization: CITY, DEPARTMENT, EMPLOYEE, PURCHASE, and SALE
  - University: STUDENT, COURSE
- An occurrence of an entity type is an object (or an entity) of the class that the entity type represents.

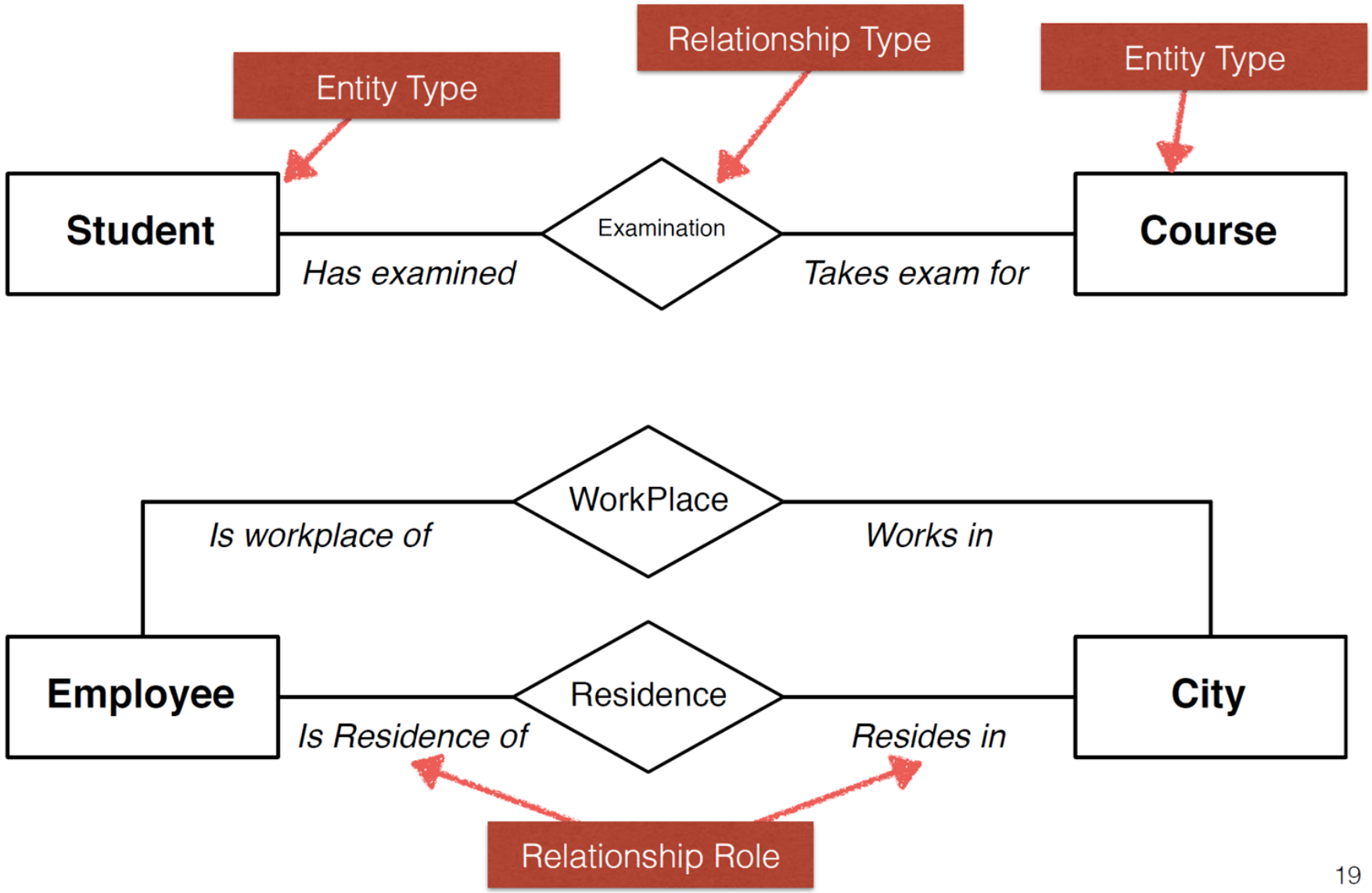


Relationship Types

- Logical links between two or more entity types
  - Defines a set of associations among occurrences from these entity types
  - An entity type is said to participate in a relationship
- Examples:
  - RESIDENCE is an example of a relationship that can exist between the entity types CITY and EMPLOYEE
  - EXAM is an example of a relationship that can exist between the entity types STUDENT and COURSE



Relationship types in the E-R model

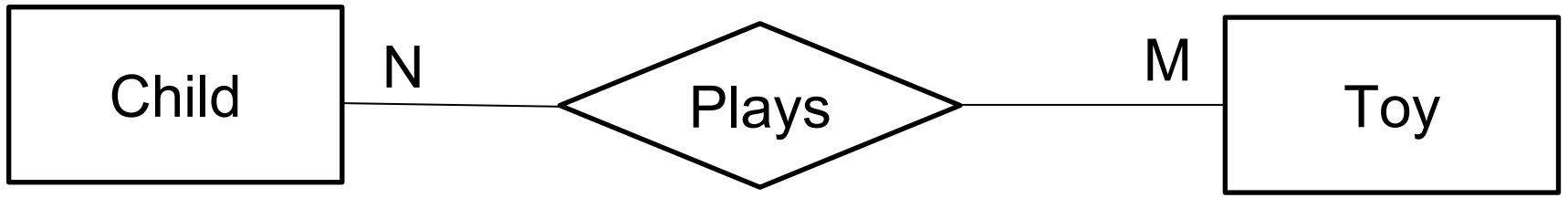


About Relationship Types

- An occurrence of a relationship type is an n-tuple made up of occurrences of entity types, one for each of the entity types involved
- Degree of a relationship type
  - Number of participating entity types (Binary, Ternary, Recursive)
- No identical occurrences!

Relationship Types

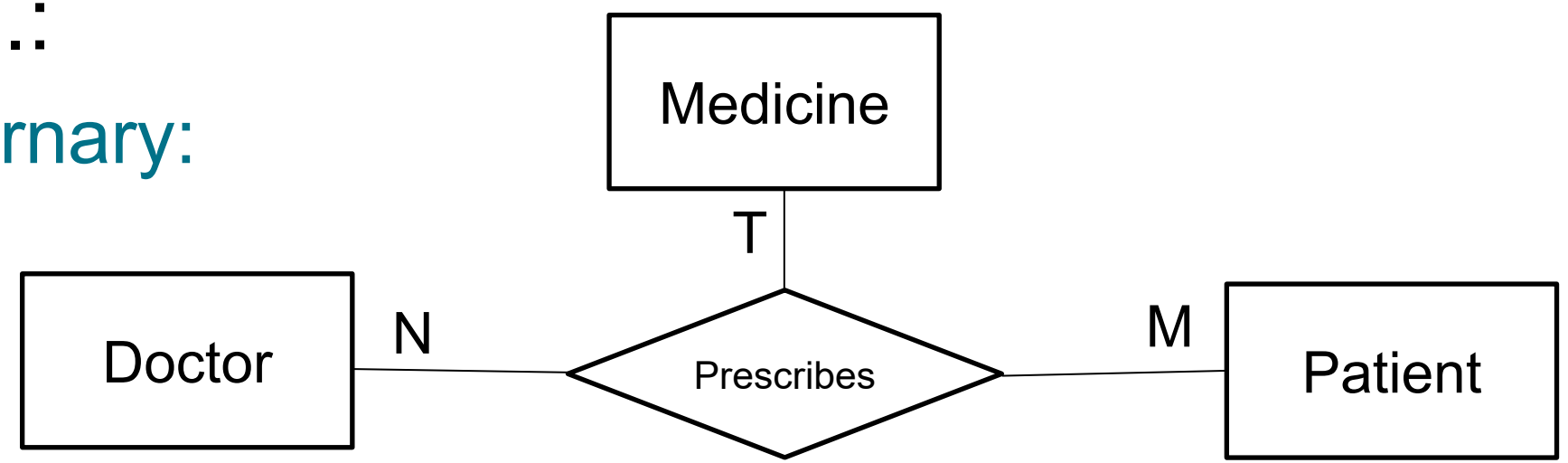
- E.g.:
  - Binary:



- A **Child** *Plays* with M **Toys**, and a **Toy** can be *Played* with by N **Children**

Relationship Types

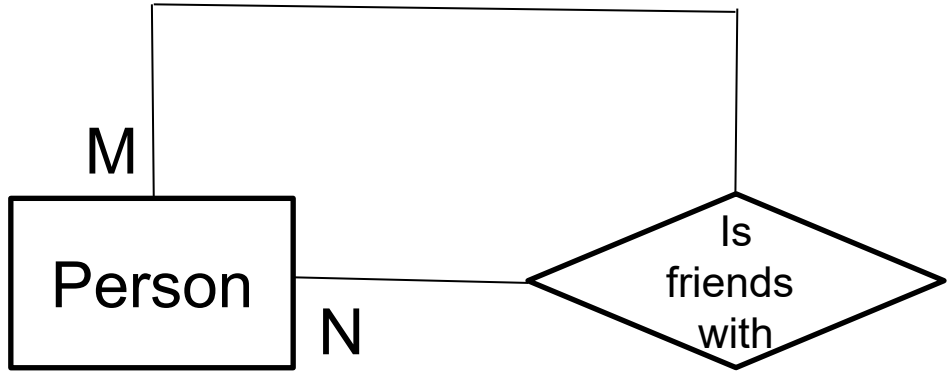
- E.g.:
  - Ternary:



- A **Doctor** can *Prescribe* T **Medicine** to M **Patients**, a **Medicine** can be *Prescribed* by N **Doctors** to M **Patients**, a **Patient** can be *Prescribed* T **Medicine** by N **Doctors**

Relationship Types

- E.g.:
  - Recursive:



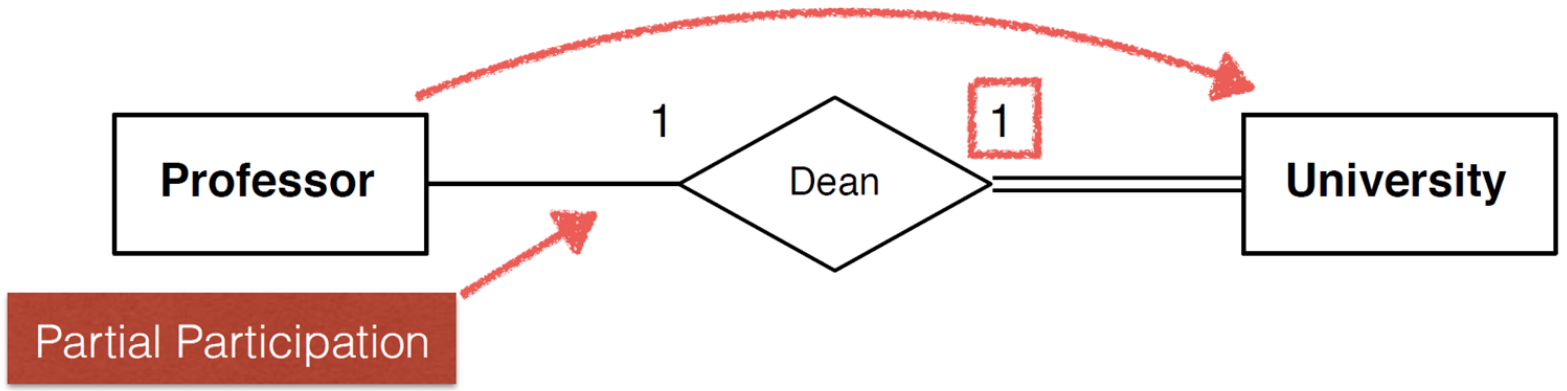
- A **Person** can *Be friends with* M **Persons**
- Can a Person Be friends with themselves?

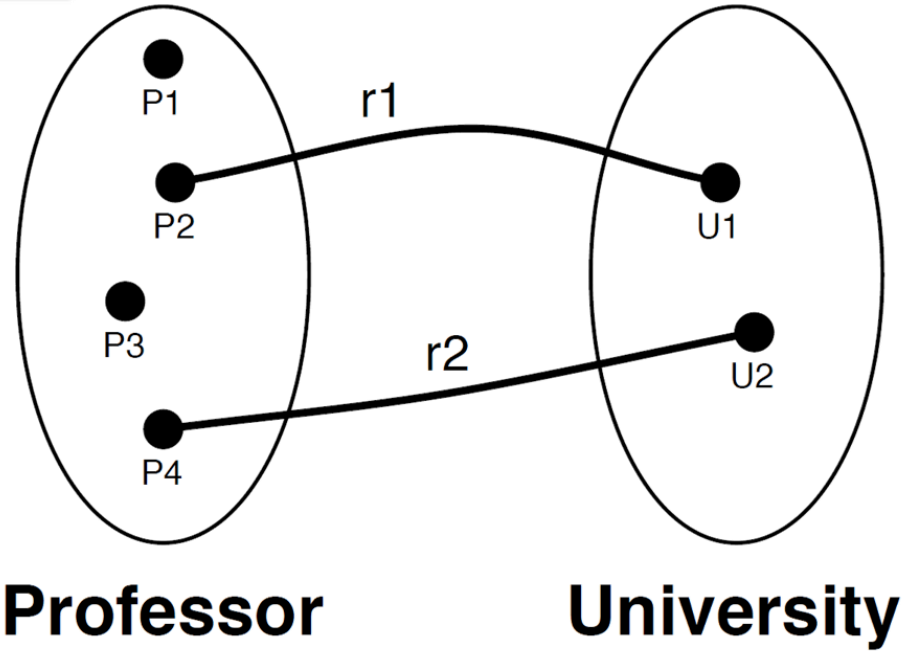
Structural Constraints

- **Cardinality**: describes the maximum and minimum number of relationship occurrences in which an entity occurrence can participate
  - Specified for each entity participating in a relationship
- **Maximum** (cardinality ratio) can be
  - 1: each occurrence of the entity is associated at most with a single occurrence of the relationship
  - N: each occurrence of the entity is associated with an arbitrary number of occurrences of the relationship
- **Minimum** (participation constraint) can be
  - 0: the participation in the relationship is optional or partial
  - 1: the participation is mandatory or total (existence dependency)

Example of 1:1 relationship type /1

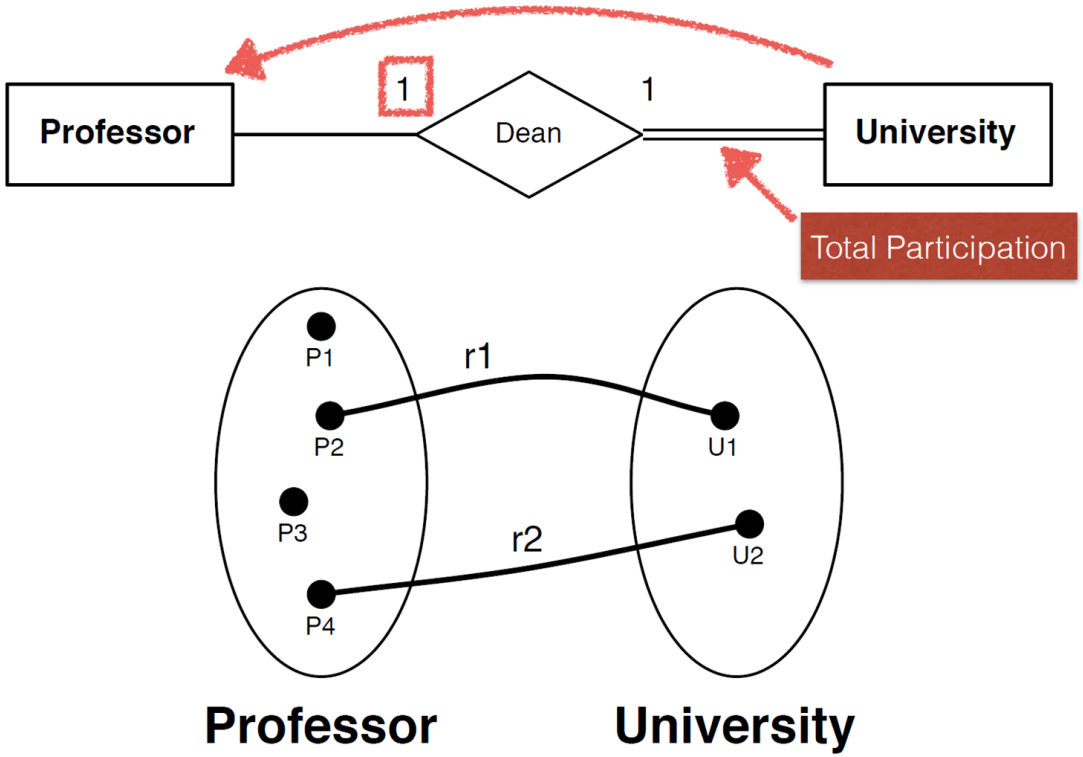
A professor may be the dean of a single university





Example of 1:1 relationship type /2

A university must have one professor acting as a dean

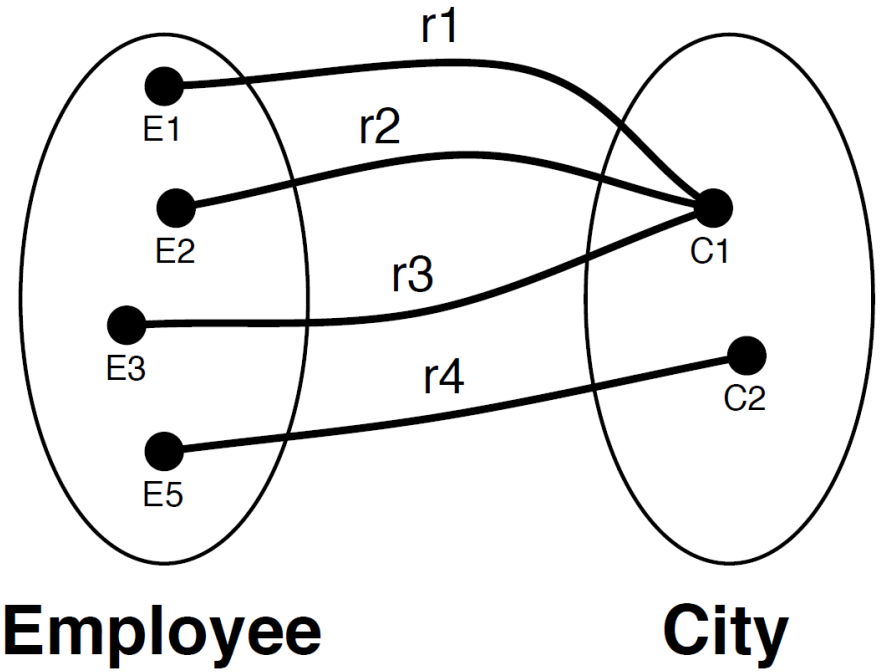


- Minimum cardinalities are rarely 1 for all the entities participating in a relationship

Example of 1:N relationship type

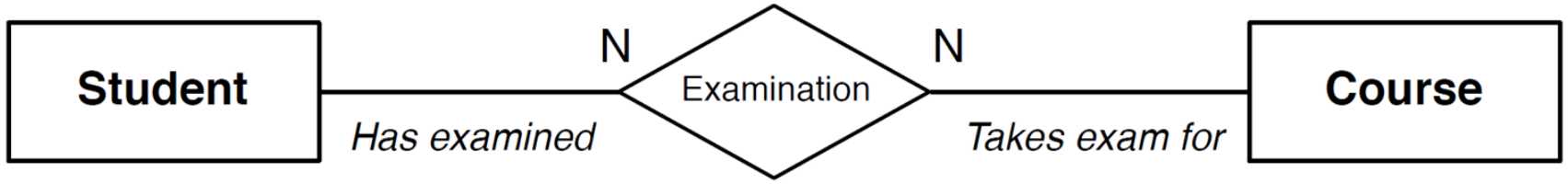
An employee must reside in a city. A city must be the residence of at least one employee

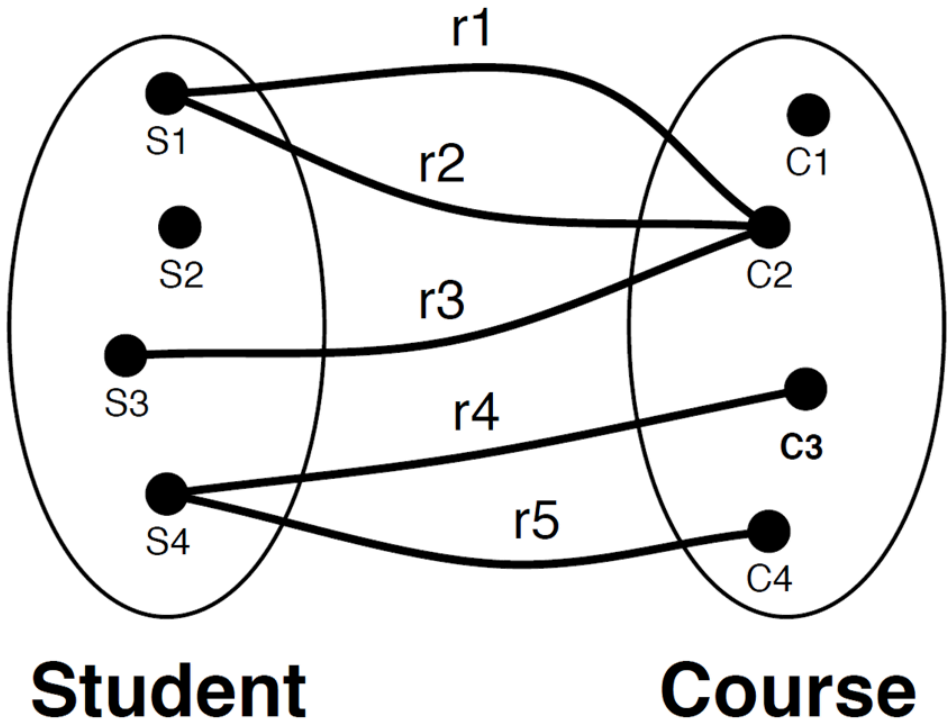
Relationship Role



Example of N:N relationship type

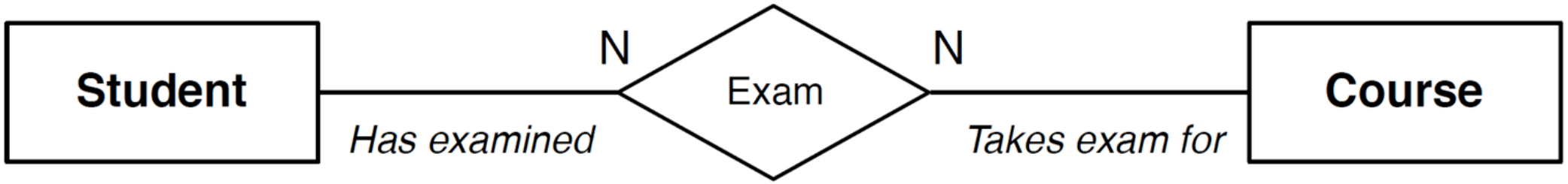
A student can take exams for many courses. A course might examine several students.





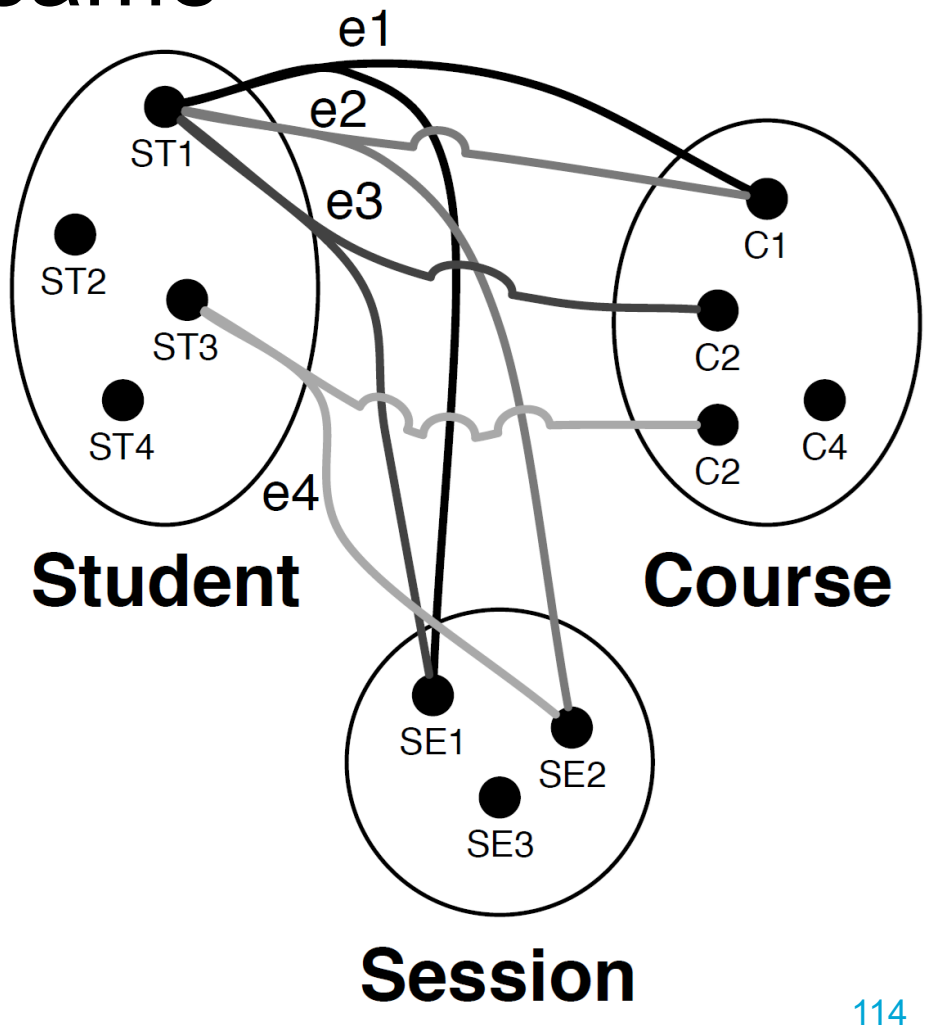
Limits of binary relationships

- Can we represent a student taking the same exam multiple times/sessions?

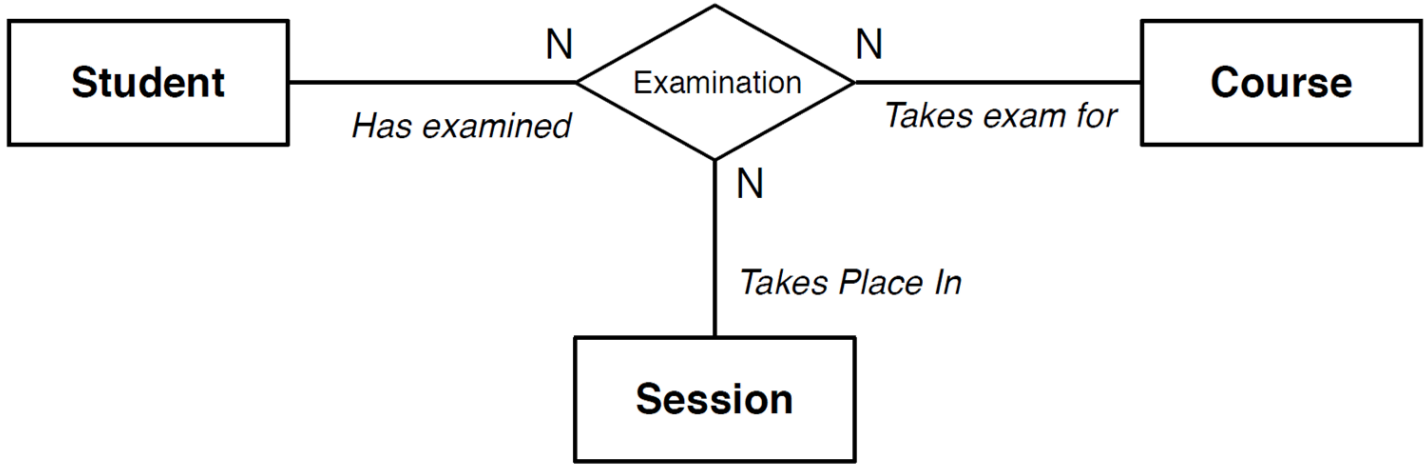


Example of ternary relationship

- A student can repeat the same exam in multiple sessions
- Example:
  - ST1 C1 SE1
  - ST1 C1 SE2
  - ST1 C2 SE1



Example of ternary relationship type

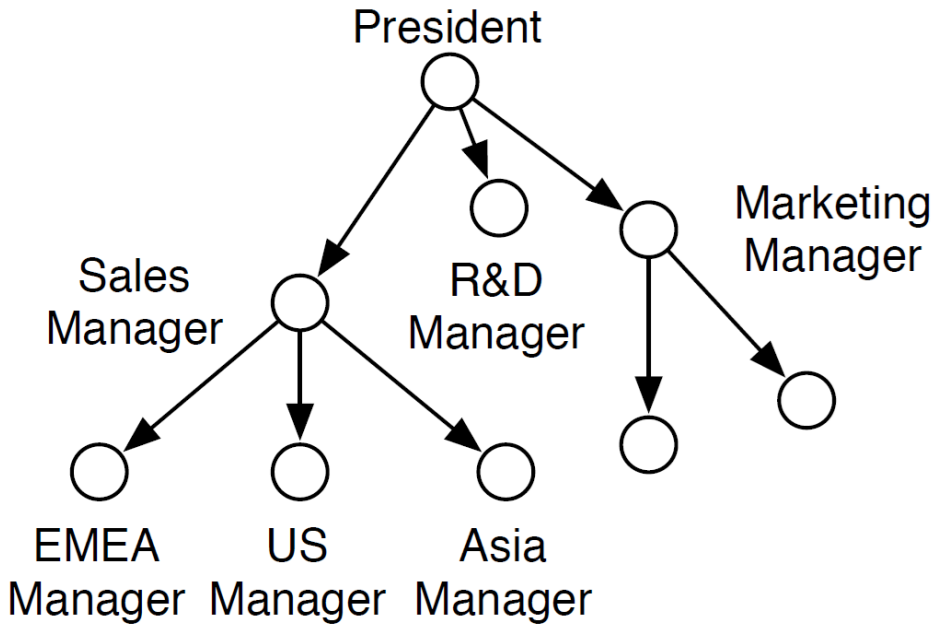


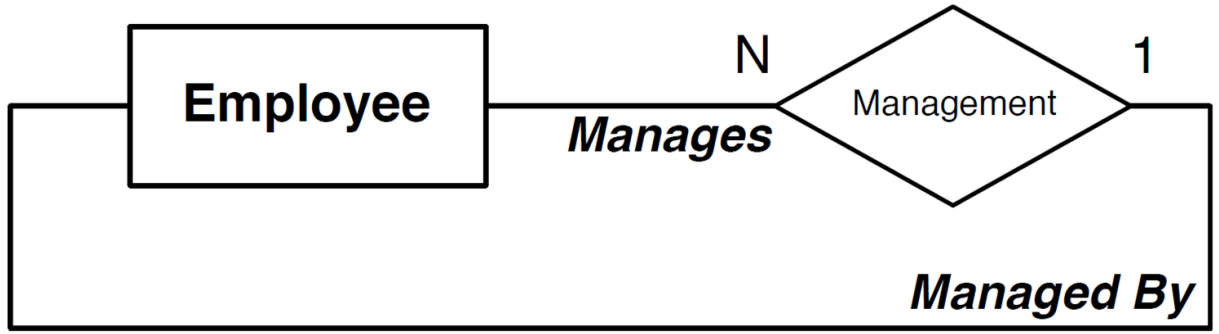
- Maximum cardinalities in an n-ary relationships are almost always N
- If an entity E participates with maximum cardinality 1, it is possible to remove the n-ary relationship and relate E with one of the other entities with a binary relationship

More in Section 7.9 of the Elmasri-Navathe Book

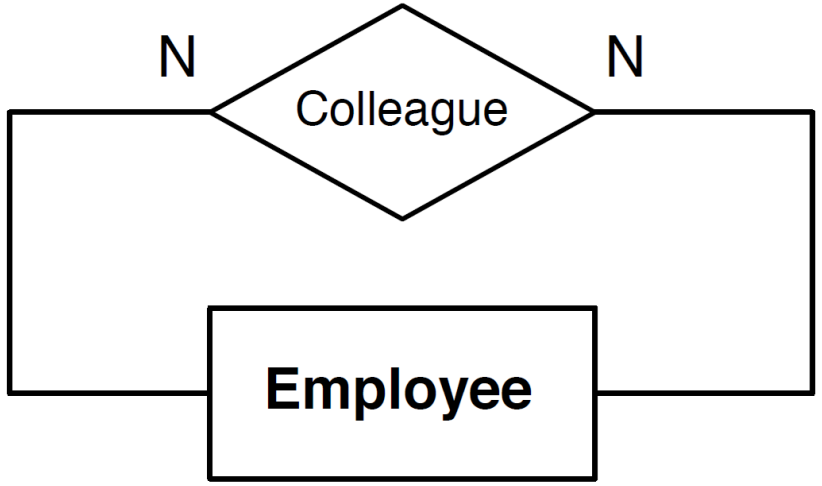
Recursive Relationship Types

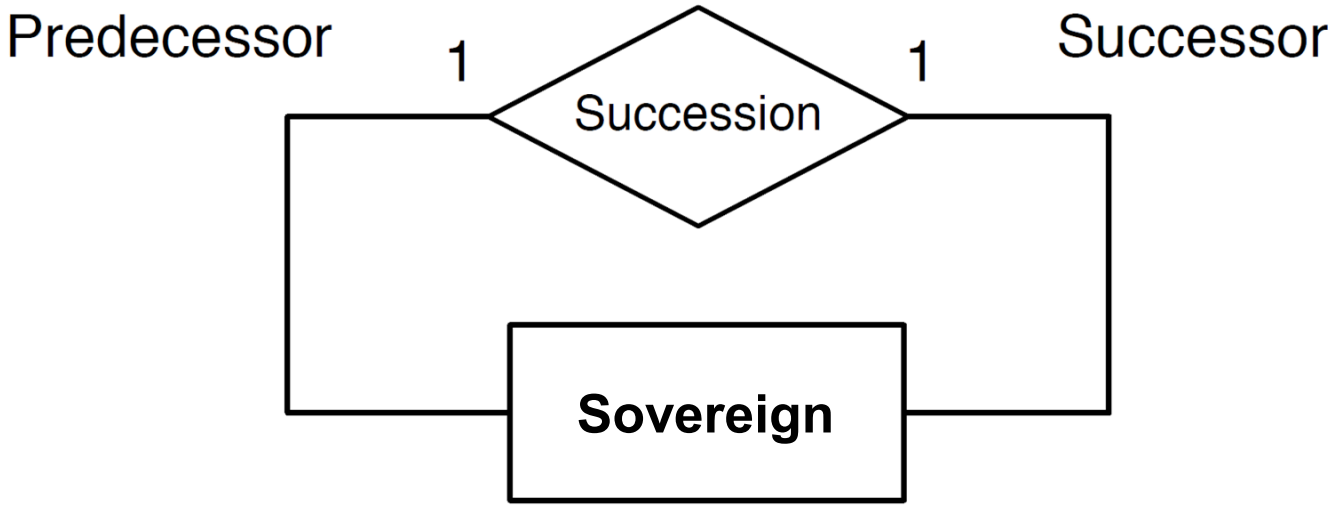
- Relationship where the same entity participates with different roles



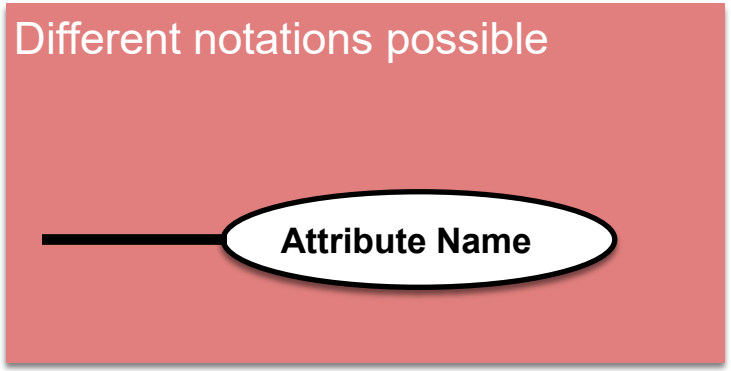


Recursive Relationships Types



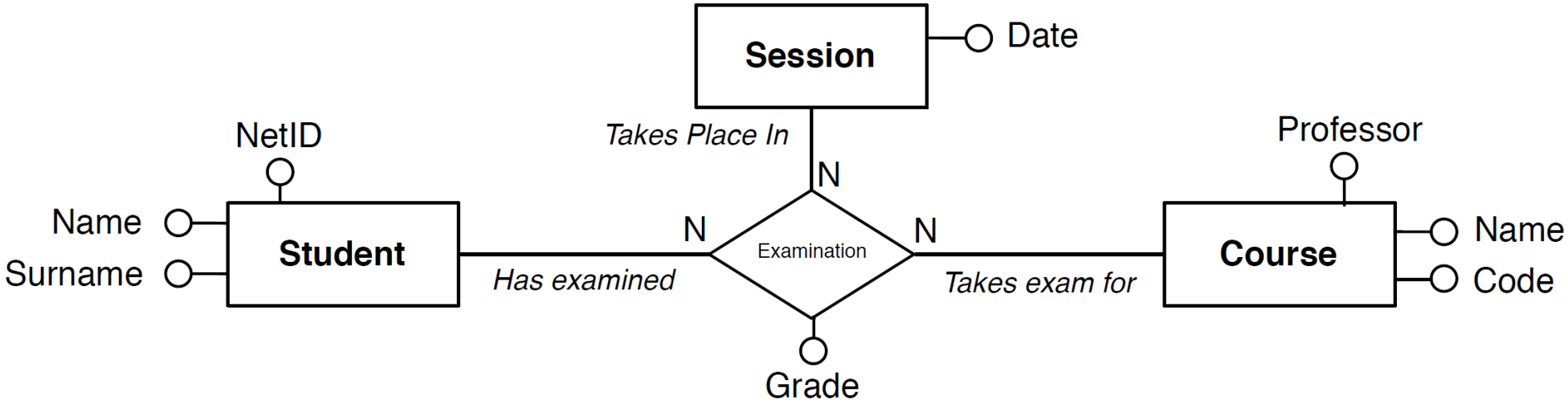


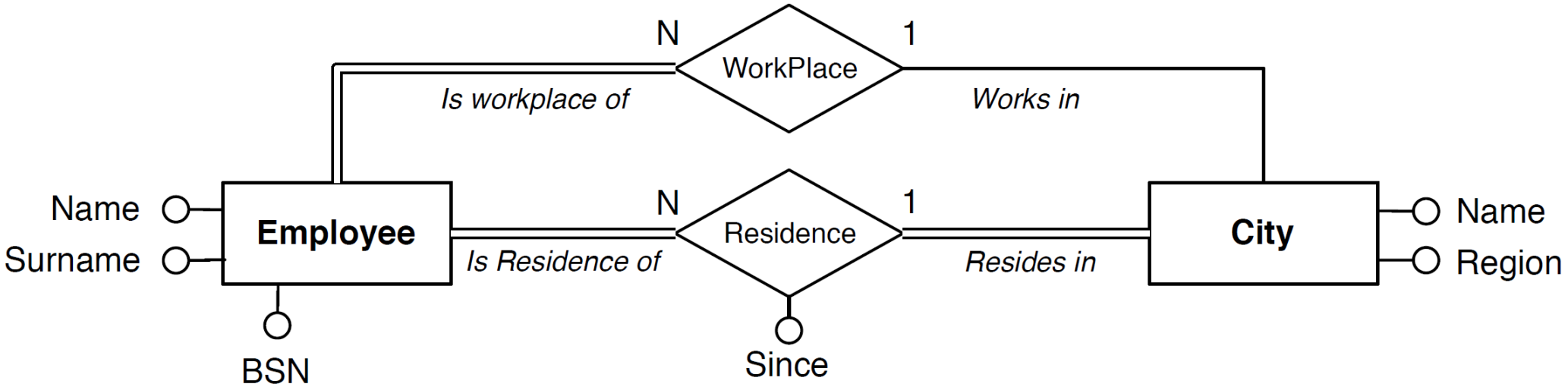
Simple Attributes



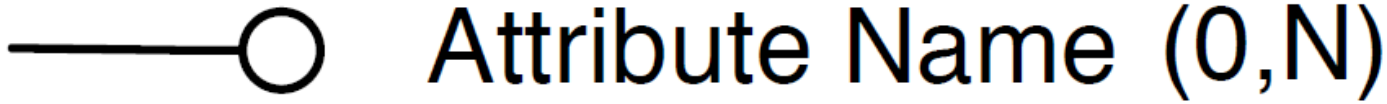
- Describe the elementary properties of entities or relationships
  - Surname, Salary and Age are possible attributes of the EMPLOYEE entity
  - Score is a possible attribute for the relationship EXAMINATION between STUDENT and COURSE
- Each attribute is characterized by a domain (not visually represented)

Examples With Attributes





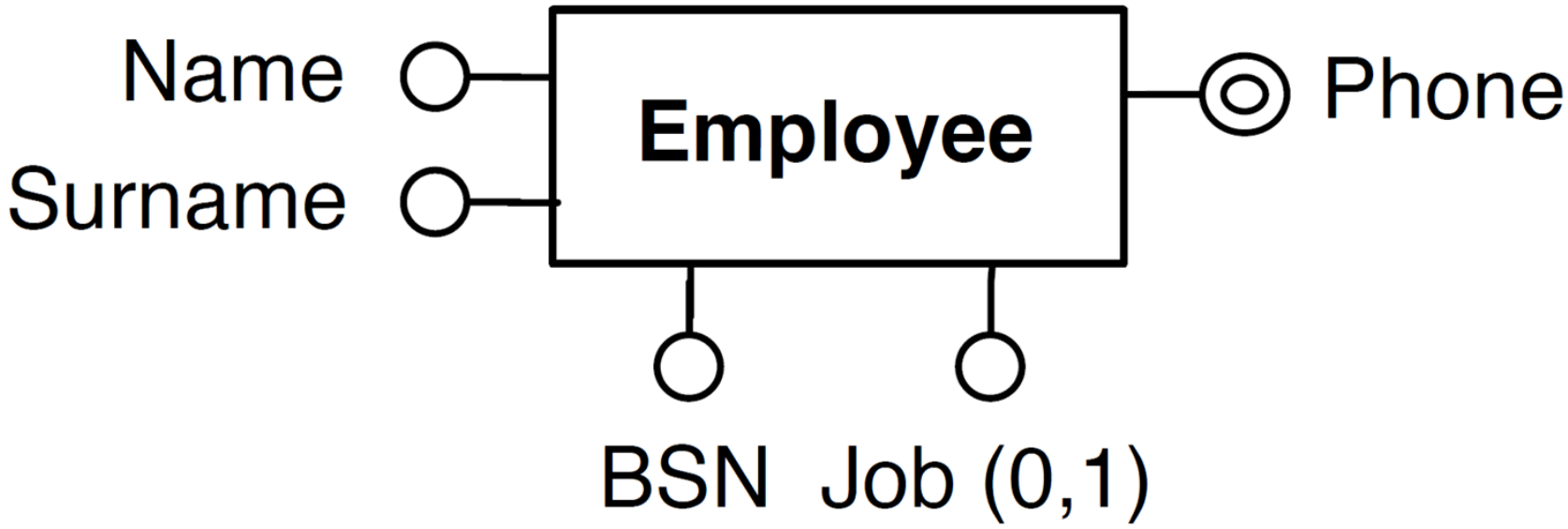
Attribute Cardinality



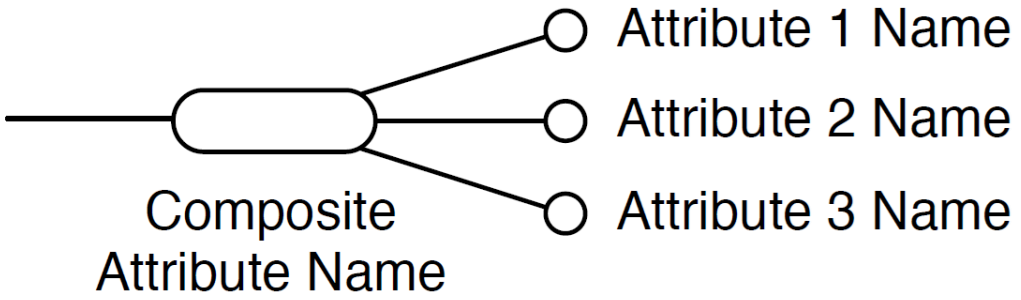
- Describes the minimum and maximum number of values of the attribute associated with each occurrence of an entity or a relationship
  - Typically, cardinality is equal to (1,1) and is omitted
- Minimum cardinality 0 means that the value can be NULL
- Maximum cardinality N means that the attribute may assume more than one value in the same instance
  - In this case we talk about multi-valued attributes, and they can be represented as follows:

Example

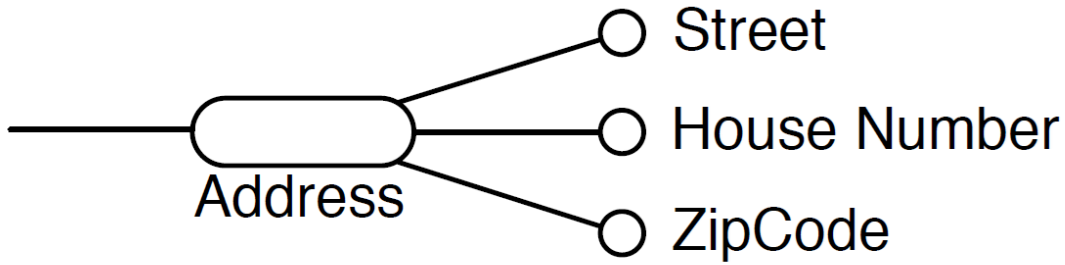
- Beware of multi-valued attributes!
  - They might represent situations that could be modelled with additional entity and relationship types
  - E.g., Job as multivalued attribute vs. entity



Composite Attributes



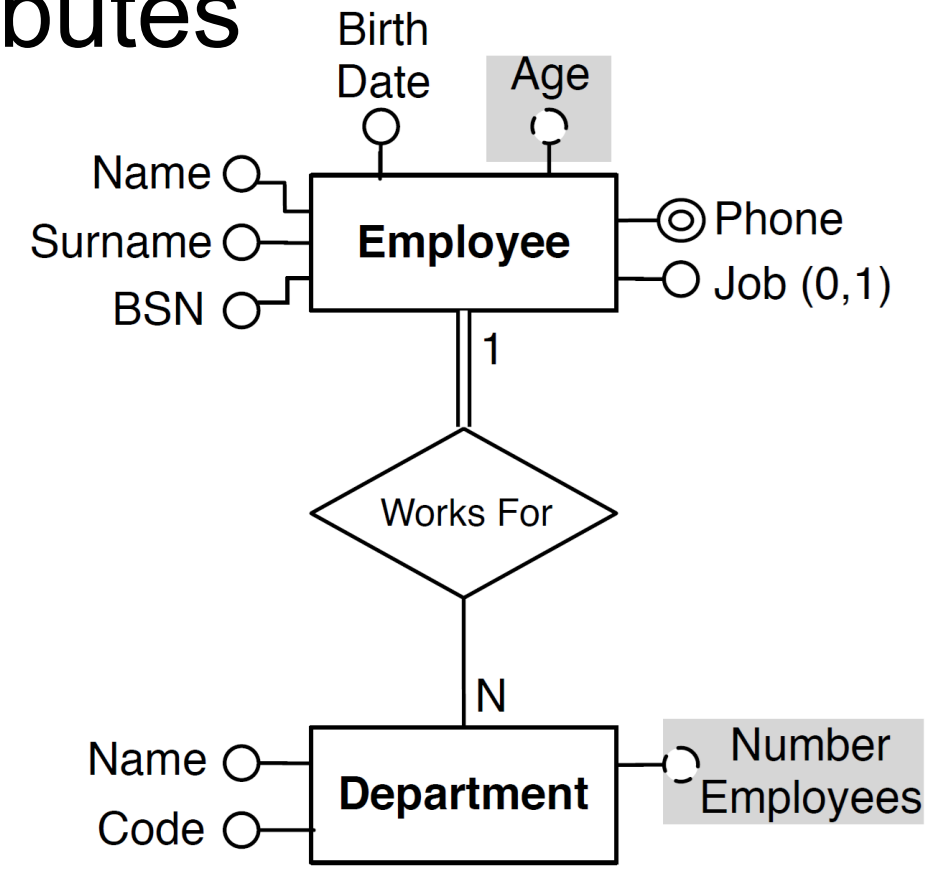
- Achieved by concatenating simpler attribute types
- Pictured by trees of atomic attributes



- Composite attributes can form hierarchies

Derived Attributes

- Attributes having values generated from other attributes
  - Calculations, algorithms or procedures
- Examples
  - Age from BirthDate
  - Number of Employees in a Department

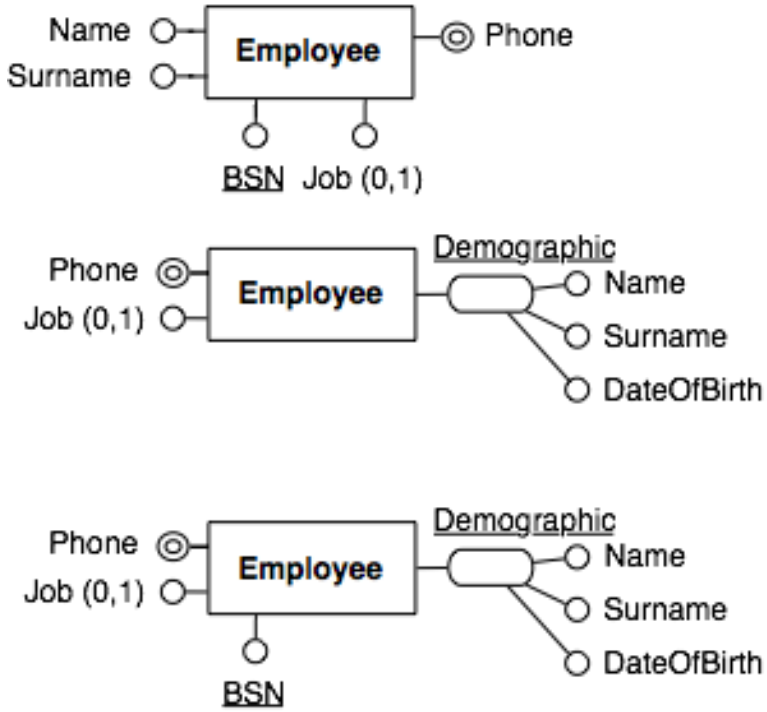


Identifiers

- Describe the concepts of the schema that allow the unambiguous identification of the entity instances
  - Attributes and/or entities
- They are specified for each entity of a model
- Relationships have no identifier!

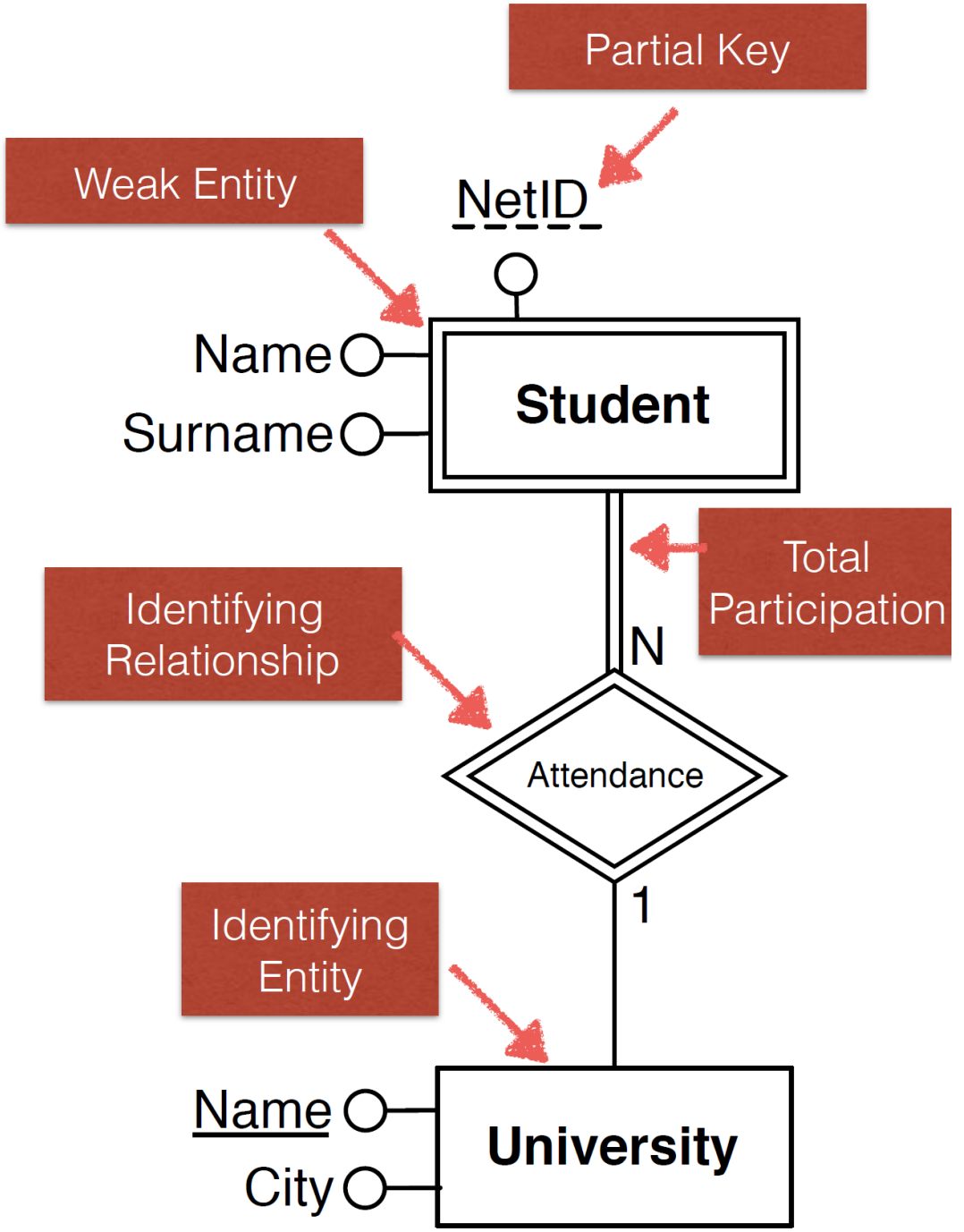
Internal Identifiers

- An identifier is often formed by one or more
attributes of the entity itself
- In this case we talk about an internal identifier
  - Also known as a key
- An identifier can involve one or more attributes
  - Each of them has (1,1) cardinality
  - A new composite attribute becomes key
- Each entity must have one identifier, but can have more than one
  - Then each distinct underlined attribute is a key
  - There is no primary key in ER (only keys)



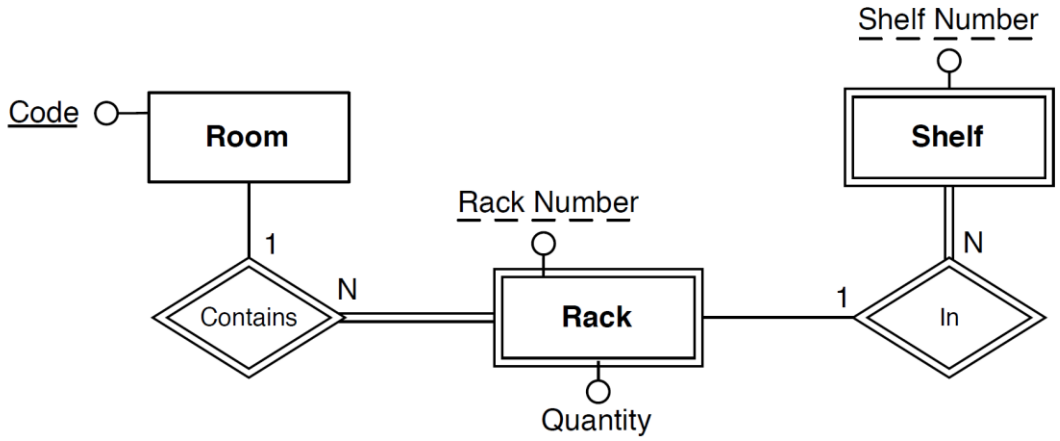
Weak Entities

- Sometimes the attributes of an entity are not sufficient to identify its occurrences unambiguously
  - Weak Entities
  - Entities may not have internal identifiers
  - Or partial identifiers (Partial keys)
- Other entities need to be involved in the identification
  - This is called an external identifier
- The relationship that relates a weak entity to its owner is called identifying relationship



General Observations

- Weak entities can sometimes be represented as complex attributes
- An external identifier can involve one or more entities, provided that each of them is member of a relationship to which the entity to identify participates with cardinality equal to (1,1)
- An external identifier can involve an entity that is in its turn identified externally, as long as cycles are not generated



Enhanced Entity-Relationship Diagrams (EER)

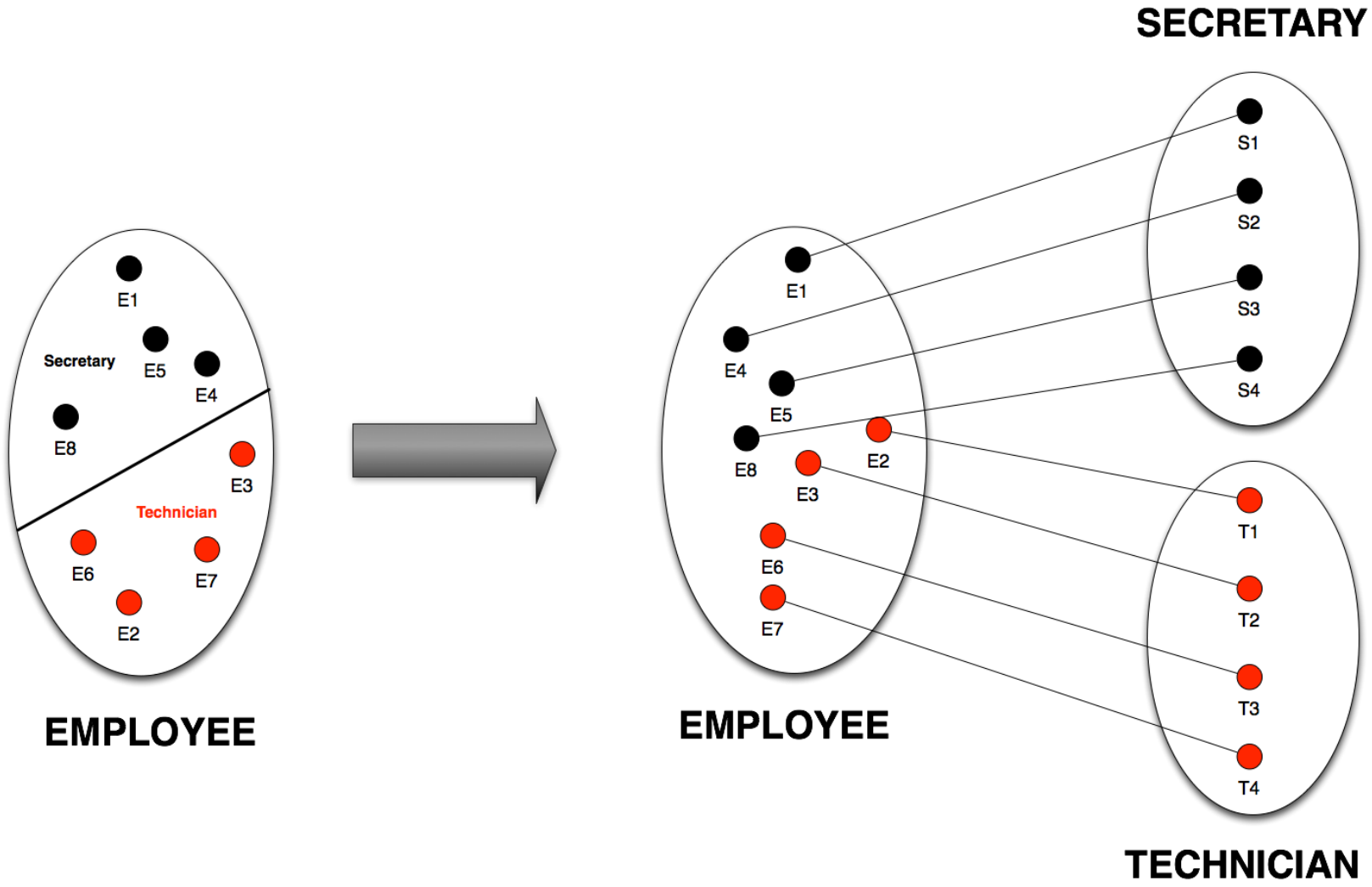
- Created to design more accurate database schemas reflecting the data properties and constraints more precisely
- More complex requirements than traditional applications
- EER model includes all modelling concepts of the ER model

- In addition, EER includes:
  - Subclasses and superclasses
  - ~~Specialisation and generalisation~~ (not covered in this class)
  - ~~Category or union type~~ (not covered in this class)
  - ~~Attribute and relationship inheritance~~ (not covered in this class)

Class / Subclass (Type / Subtype)

- An entity type is used to represent a type of instance (attribute and relationships)
  - `EMPLOYEE` (name, BSN, BirthDate, etc.)
- But also, the *collection of entities (entity set) of that type* that exist in the database
  - E.g., entities that represent specific types of employees
  - `SECRETARY`, `TECHNICIANs`, `MANAGERs`, etc.
- The relationship that exists between a type of entity and its entity set is called a **class/ subclass** relationship
  - `EMPLOYEE` is the **super type** or **super class**
  - `SECRETARY`, `TECHNICIAN` etc. are **subtypes** or **subclasses** of `EMPLOYEE`

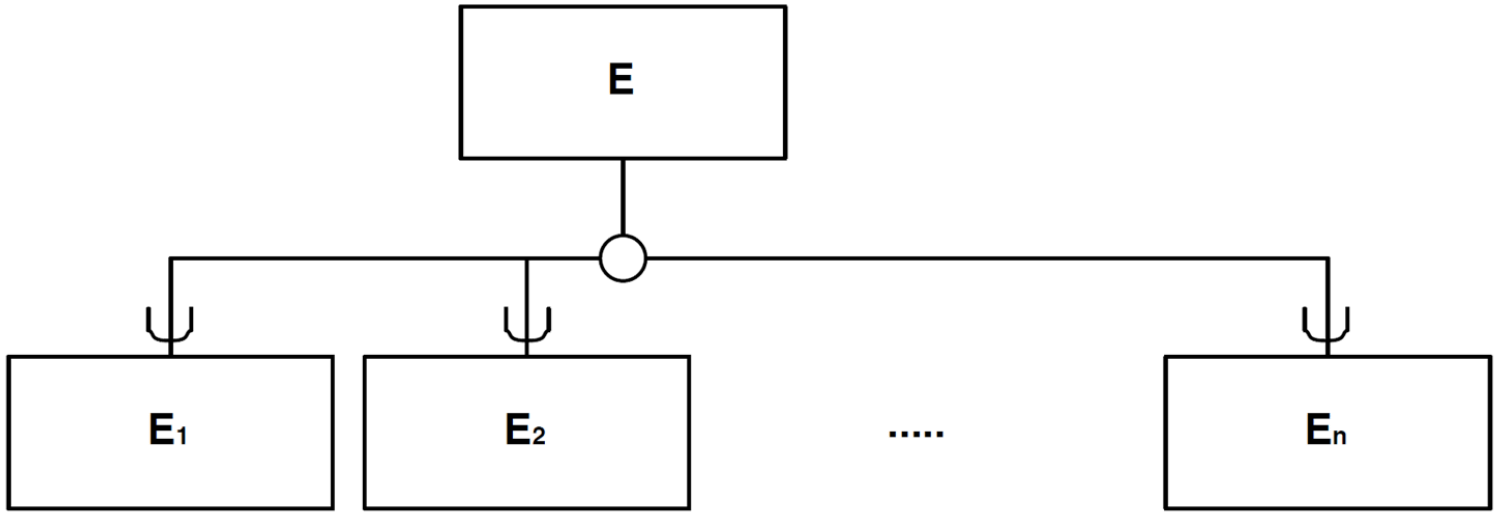
Example Class-Subclass /1



NOT a 1:1 relationship type, but a relationship between OCCURRENCES

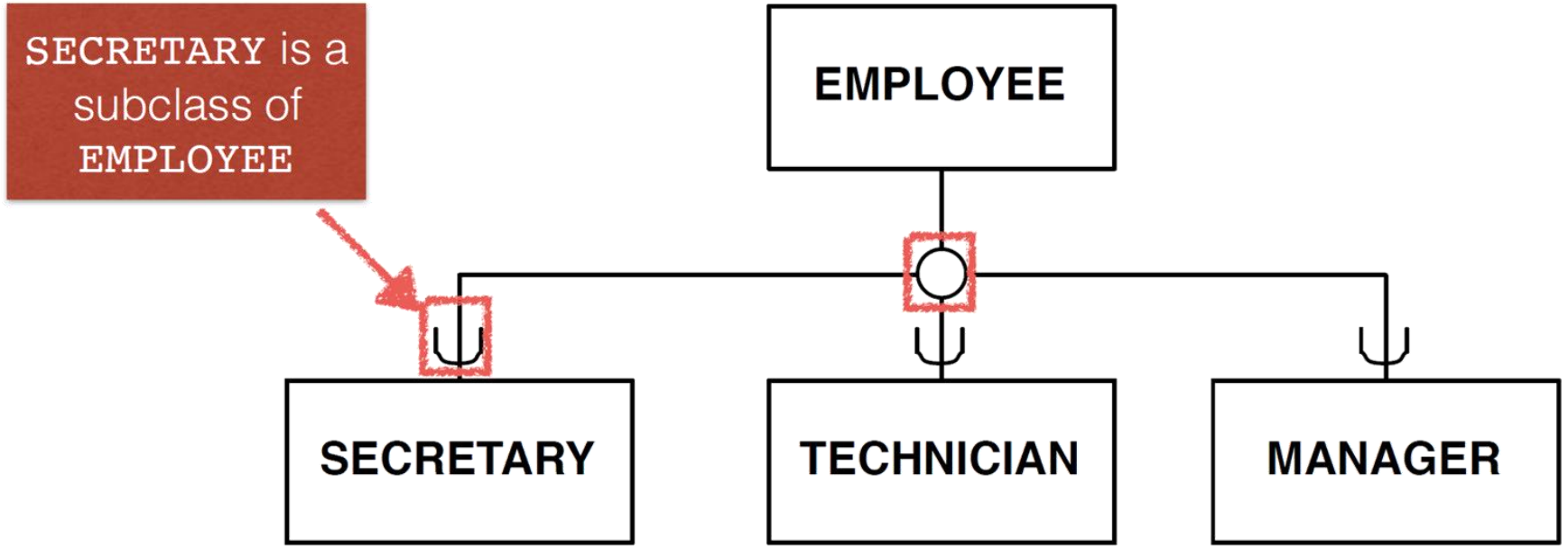
Class/ Subclass relationship

- Also called **generalisation** or **specialisation** relationship
- Represent logical links between an entity E, known as parent entity, and one or more entities $E_1, \ldots, E_n$ called child entities, of which E is more general, in the sense that it comprises them as a particular case.
- In this situation we say that E is a **generalisation** of $E_1, \ldots, E_n$ and that the entities $E_1, \ldots, E_n$ are a specialisation of the E entity



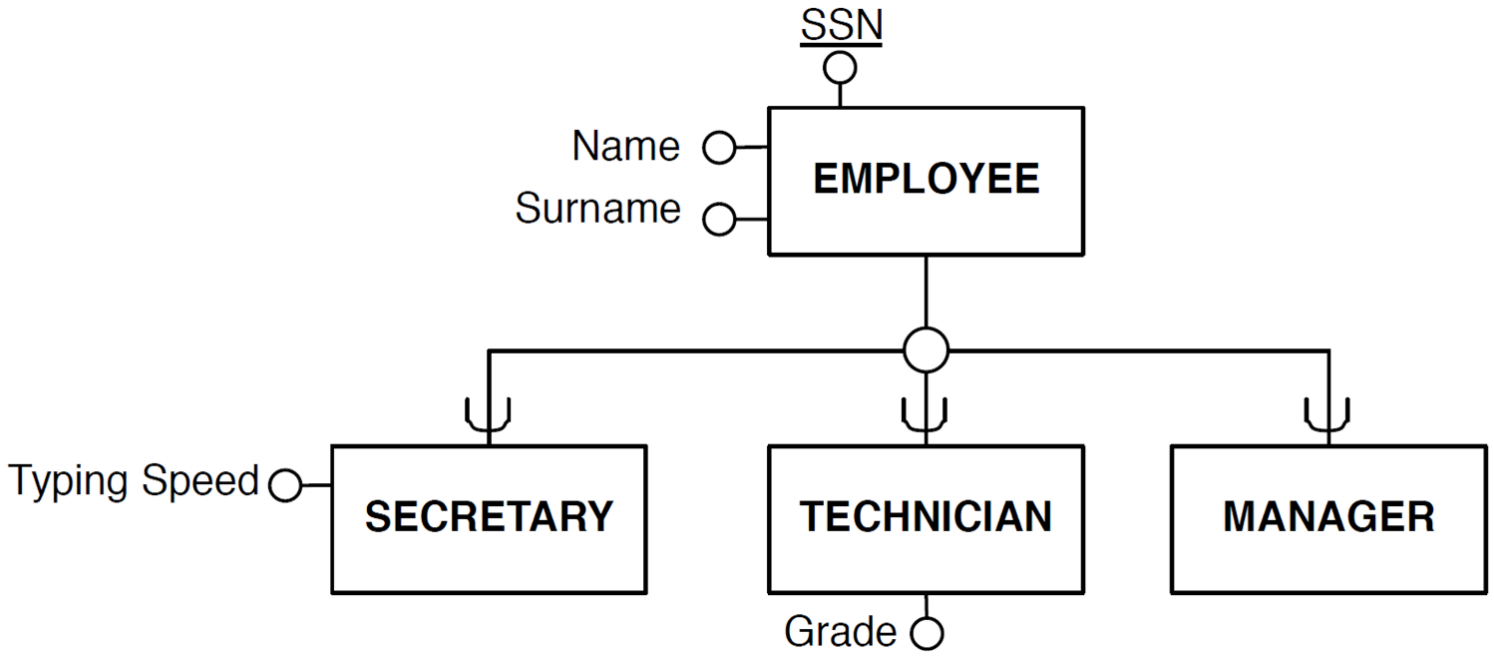
Example Class-Subclass /2

- The subset symbol indicates the *direction* of the **specialisation**
- The circle defines the entities involved in the specialisation



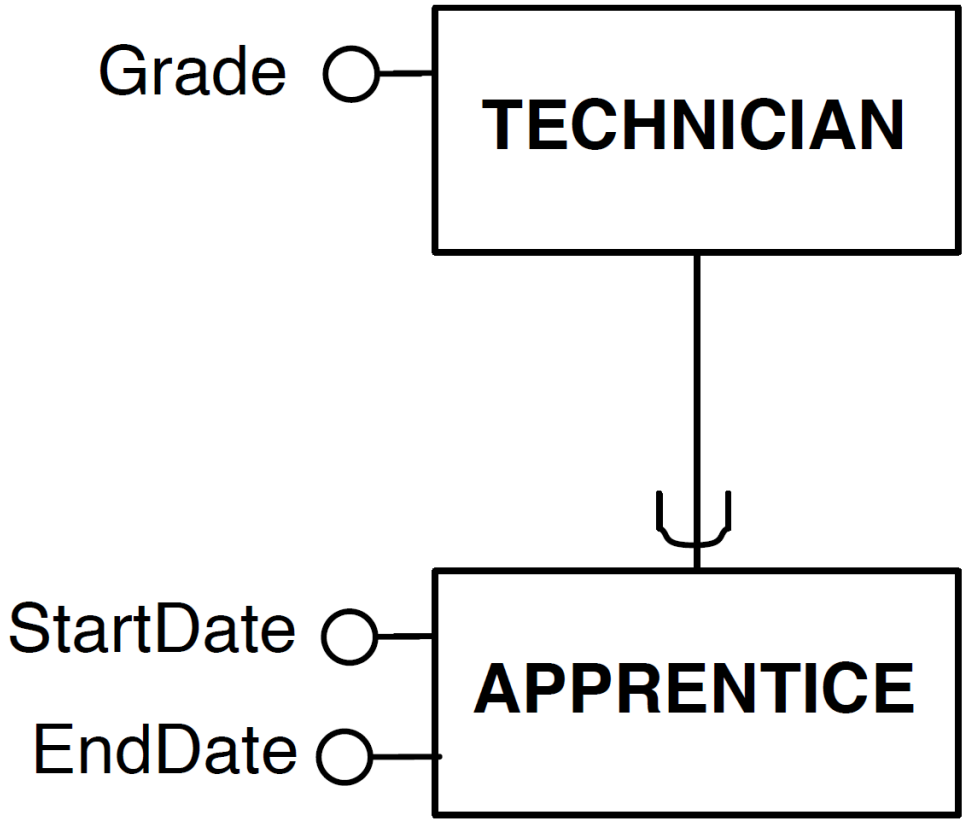
Properties of Specialisation

- Every occurrence of a child entity is also an occurrence of the parent entity
- Inheritance
  - Every property of the parent entity (attributes, identifiers, relationships and other generalisations) is also a property of a child entity.
- Specialisation can define specific attributes and/or specific relationship types



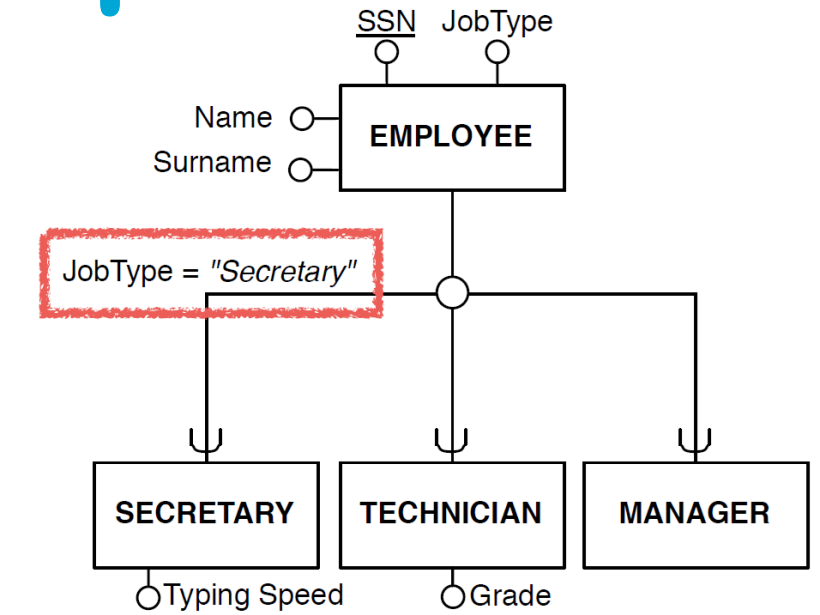
Subset

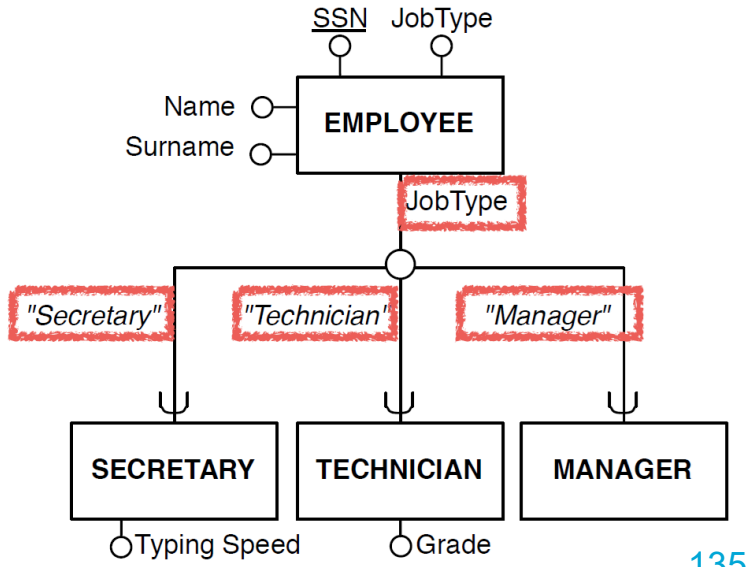
- Special class of specialisation with a single child entity



Constraints: Membership

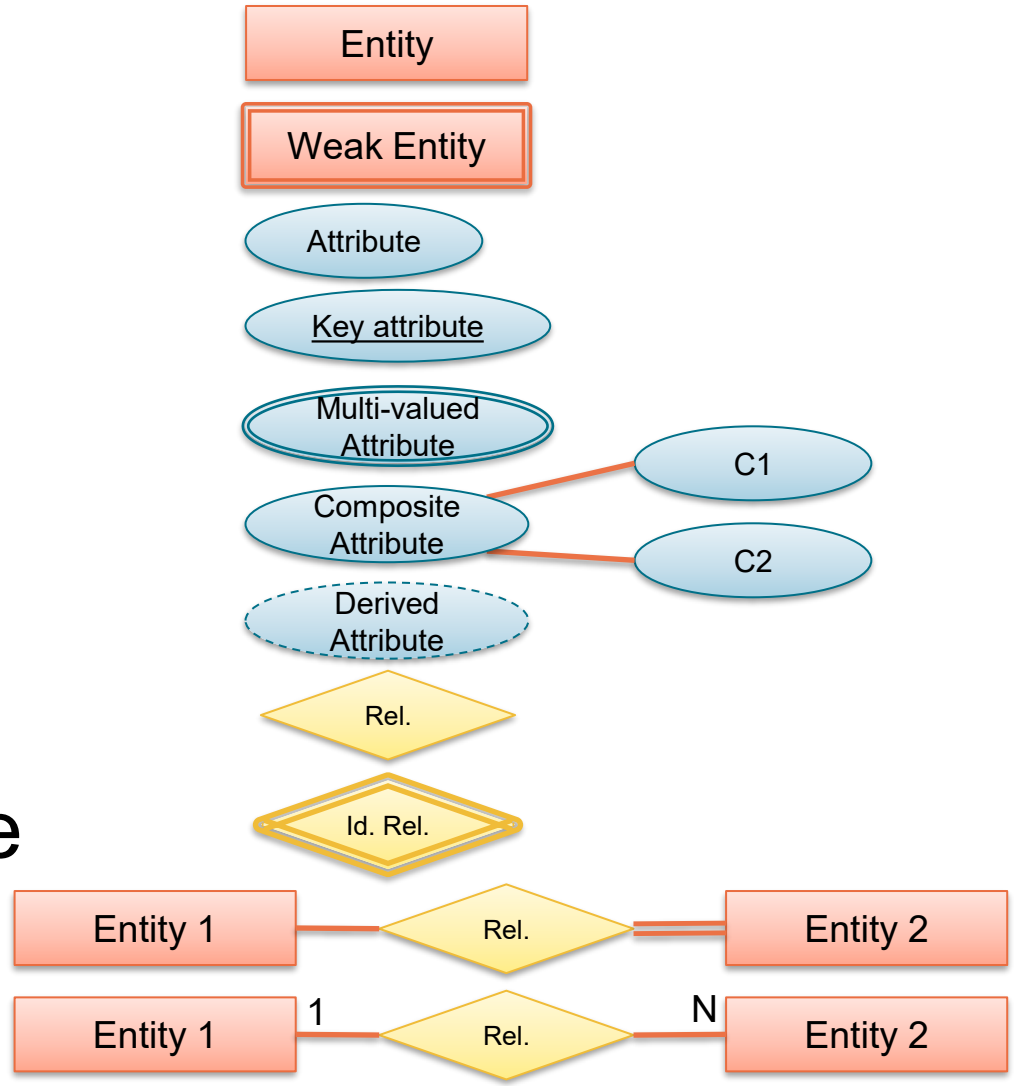
- In some specialisations we can determine exactly the entities that will become members of each subclass
  - Predicate-defined (or condition-defined) subclasses
- All the subclasses in a specialisation have their membership condition on the same attribute
  - Attribute-defined specialisation (Defining attribute of the specialisation)
- No specific condition
  - User-defined subclass
  - Specialisation defined by the database user, instance by instance





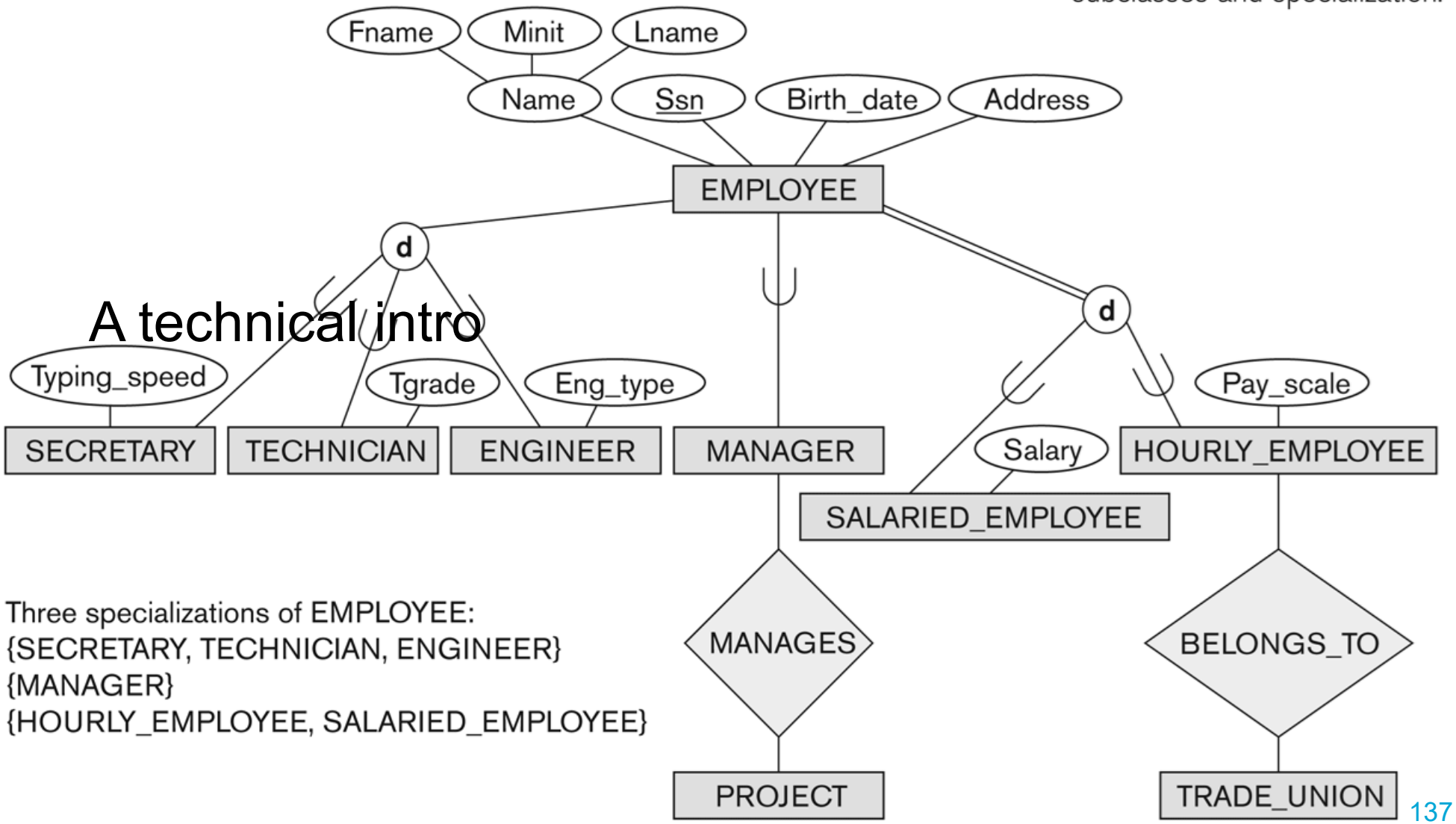
Summary ER Modelling

- Entity (Type)
- Weak Entity (Type)
- Attribute
- Key Attribute
- Multi-valued Attribute
- Composite Attribute
- Derived Attribute
- Relationship Type
- Identifying Relationship Type
- Total Participation
- 1:N Cardinality



Summary ER

Figure 4.1
EER diagram notation to represent subclasses and specialization.



A technical intro

Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Web And Database Technology

See you next time!

TU Delft WIS Web Information Systems