Databases Questions
Fragen zu Datenbanken

Thomas Weise (汤卫思)

February 20, 2026

Abstract

Here you find questions for practice for the course *Databases*. The course book and all teaching material are provided at `https://thomasweise.github.io/databases`. The questions are provided in both English and German language.

Hier finden Sie Fragen zum Üben für den Kurs *Databases*. Das Kursbuch und alles Lehrmaterial wird auf `https://thomasweise.github.io/databases` zur Verfügung gestellt. Die Fragen sind in Englisch und Deutsch bereitgestellt.

Contents

Preface

In this document, we provide questions to support your understanding and self-studying for the subject of databases. These questions accompany the book *Databases* [90], which is freely available at `https://thomasweise.github.io/databases`. The questions are provided both in English and German.

In diesem Dokument stellen wir Fragen zur Verfügung, um Ihr Verstehen und Selbst-Lernen des Themas Datenbanken zu unterstützen. Die Fragen begleiten das Buch *Databases* [90], welches unter `https://thomasweise.github.io/databases` frei zur Verfügung steht. Die Fragen stehen in Englisch und Deutsch bereit.

Visit the course website / Besuchen Sie die Kurswebseite:

Chapter 1

Basics / Grundlagen

1.1 Three-Schema-Architecture / Drei-Schema-Architektur (Q1)

EN Name the components of the three-schema-architecture and their different purposes.

DE Nennen Sie die Komponenten der Drei-Schema-Architektur und ihre verschiedenen Zwecke.

1.2 ACID (Q2)

EN Name the four ACID properties that a DBMS must enforce for transactions. For each of these four properties, describe its meaning and purpose.

DE Nennen Sie die vier ACID-Eigenschaften, die ein DBMS für Transaktionen sicherstellen muss. Beschreiben Sie die Bedeutung und den Zweck jeder dieser vier Eigenschaften.

1.3 Atomicity / Atomizität (Q3)

EN Explain what the property *atomicity* is that databases should have.

DE Erklären Sie, was die Eigenschaft *Atomizität* ist, die Datenbanken haben sollten.

1.4 Consistency / Konsistenz (Q4)

EN Explain what the property *consistency* is that databases should have.

DE Erklären Sie, was die Eigenschaft *Konsistenz* ist, die Datenbanken haben sollten.

1.5 Isolation (Q5)

EN Explain what the property *isolation* is that databases should have.

DE Erklären Sie, was die Eigenschaft *Isolation* ist, die Datenbanken haben sollten.

1.6 Durability / Dauerhaftigkeit (Q6)

EN Explain what the property *durability* is that databases should have.

DE Erklären Sie, was die Eigenschaft *Dauerhaftigkeit* ist, die Datenbanken haben sollten.

1.7 Security / Sicherheit (Q7)

EN Describe the importance of *security* in the context of databases. Give two examples of a scenarios where security aspects need to be considered and possible measures that could be taken in them.

DE Beschreiben Sie die Wichtigkeit von Datenschutz und Datensicherheit im Kontext von Datenbanken. Geben Sie zwei Beispiele von Szenarien, wo Datenschutz und Datensicherheit beachtet werden muss und die Maßnahmen, die in ihnen getroffen werden könnten.

1.8 Concurrency / Gleichzeitigkeit (Q8)

EN Give an example for concurrent access to a database. Outline one possible error that could occur if the ACID principles are not enforced.

DE Geben Sie ein Beispiel für gleichzeitigen Zugriff auf eine Datenbank. Geben Sie einen möglichen Fehler an, der entstehen kann, wenn die ACID-Prinzipien nicht durchgesetzt werden.

1.9 Networks / Netzwerke (Q9)

EN What role do computer networks play in the context of today's database applications?

DE Welche Rolle spielen Computer-Netzwerke im Kontext heutiger Datenbankapplikationen.

1.10 Software (Q10)

EN Which DBMS are we using in our course?

DE Welches DBMS benutzen wir in unserem Kurs?

1.11 Alternatives / Alternativen (Q11)

EN Why are spreadsheets offered by programs such as Microsoft `Excel` or LibreOffice `Calc` not good solutions for managing the operational data of a bank? Give at least four reasons.

DE Warum sind Tabellen (EN: *spreadsheets*), wie sie von Microsoft `Excel` und LibreOffice `Calc` angeboten werden, keine guten Lösungen zum Managen der operationalen Daten einer Bank?

1.12 Lifetime / Lebensdauer (Q12)

EN Are databases short-lived or long-time assets of an organization? What does this imply regarding the value of databases and the software engineering used to develop them?

DE Sind Datenbanken kurzlebige oder langlebige Besitztümer einer Organisation? Was bedeutet das für ihren Wert und auch die softwareingenieurstechnischen Methoden, die zu ihrer Entwicklung angewandt werden?

Chapter 2

Tools / Werkzeuge

2.1 PostgreSQL (Q13)

EN What is PostgreSQL? What is it used for?

DE Was ist PostgreSQL? Wofür benutzt man es?

2.2 PostgreSQL (Q14)

EN What is the client program shipping with PostgreSQL? How can it be used?

DE Was ist das Klientenprogramm, das mit PostgreSQL ausgeliefert wird? Wie kann man es benutzen?

2.3 Open Source DBMS (Q15)

EN Name three open source relational database management systems (DBMSes).

DE Nennen Sie drei Open Source relationale Datenbankmanagementsysteme (DBMSe).

2.4 Commercial DBMS / Kommerzielle DBMS (Q16)

EN Name two commercial / proprietary / closed source relational database management systems (DBMSes).

DE Nennen Sie zwei kommerzielle / proprietäre / Closed-Source Datenbankmanagementsysteme (DBMSe).

2.5 Forms / Formulare (Q17)

EN What are forms in the context of databases? What is their purpose and how are they used? Name one program that can be used to create forms for databases.

DE Was sind Formulare im Kontext von Datenbanken? Was ist ihr Zweck und wie können Sie benutzt werden? Nennen Sie ein Programm, das benutzt werden kann, um Formulare für Datenbanken zu erstellen.

2.6 Reports / Berichte (Q18)

EN What are reports in the context of databases? What is their purpose and how are they used? Name one program that can be used to create reports for databases.

DE Was sind Berichte im Kontext von Datenbanken? Was ist ihr Zweck und wie können Sie benutzt werden? Nennen Sie ein Programm, das benutzt werden kann, um Berichte für Datenbanken zu erstellen.

2.7 External Access / Zugriff von Außen (Q19)

EN Name one library that allows you to access a database from a programming language. Provide both the library and the programming language name.

DE Nennen Sie eine Bibliothek / ein Paket, mit dem Sie auf eine Datenbank aus einer Programmiersprache heraus zugreifen können. Geben Sie sowohl den Name der Bibliothek als auch der Programmiersprache an.

2.8 yEd (Q20)

EN What is `yEd`? What can it be used for in the context of databases (DBs) design?

DE Was ist `yEd`? Wofür kann man es im Kontext des Datenbankdesigns verwenden?

2.9 pgModeler (Q21)

EN What is `PgModeler`? What can it be used for in the context of databases (DBs) design?

DE Was ist `PgModeler`? Wofür kann man es im Kontext des Datenbankdesigns verwenden?

2.10 pgModeler vs. yEd (Q22)

EN Both the `PgModeler` and `yEd` can be used during the design of databases (DBs). Explain how the tools are different. Explain the difference between the modeling stages for which they are used.

DE Sowohl der `PgModeler` als auch `yEd` können für das Design von Datenbanken verwendet werden. Beschreiben Sie, wie diese beiden Werkzeuge sich unterscheiden. Erklären Sie die Unterschiede zwischen den Modellierungsschritten für die sie verwendet werden.

Chapter 3

Development / Entwicklung

3.1 Models / Modelle (Q23)

EN The design of databases is often understood as the sequential development of three models (conceptual model, logical model, and physical model). Explain the purpose of each of these models.

DE Das Design von Datenbanken wird oft als Abfolge der Entwicklung dreier Modelle (konzeptuelles Modell, logisches Modell und physisches Modell) verstanden. Erklären Sie den Zweck jedes dieser Modelle.

3.2 Models / Modelle (Q24)

EN Explain the difference between the conceptual and the logical schema in DB design.

DE Erklären Sie den Unterschied zwischen konzeptuellem und logischem Schema in Datenbank-Design.

3.3 Conceptual Model / Konzeptuelles Modell (Q25)

EN Why do we often have separate conceptual and logical models in the database design? What is the conceptual model good for? Why do we not just directly with the logical model?

DE Warum haben wir oft ein separates konzeptuelles und logisches Modell während der Datenbankentwicklung? Wofür braucht man ein konzeptuelles Modell? Warum fängt man nicht direkt mit dem logischen Modell an?

3.4 Requirements / Anforderungen (Q26)

EN Explain the purpose of and differences between the four types of requirements (business requirements, functional requirements, non-functional requirements, and constraints).

DE Erklären Sie den Zweck von und die Unterschiede der vier Arten von Anforderungen (Fachanforderungen, funktionale Anforderungen, nicht-funktionale Anforderungen und Projektgrenzen).

3.5 Requirements / Anforderungen (Q27)

EN What is the difference between non-functional requirements and project constraints?

DE Was ist der Unterschied zwischen nicht-funktionalen Anforderungen und den Projektgrenzen?

3.6 Requirements / Anforderungen (Q28)

EN What is the difference between business requirements and functional requirements?

DE Was ist der Unterschied zwischen Fachanforderungen und funktionalen Anforderungen?

3.7 Requirements / Anforderungen (Q29)

EN Assume that you are supposed to design a DB for managing the operations of a restaurant. Give three examples for business requirements of such an application.

DE Nehmen Sie an, dass Sie eine Datenbank zum Managen des Betriebs eines Restaurants entwickeln sollen. Geben Sie drei Beispiele für Fachanforderungen einer solchen Applikation.

3.8 Requirements / Anforderungen (Q30)

EN Assume that you are supposed to design a DB for managing the operations of a naval port. Give vier examples for functional requirements of such an application.

DE Nehmen Sie an, dass Sie eine Datenbank zum Managen des Betriebs eines Seehafens entwickeln sollen. Geben Sie four Beispiele für funktionalen Anforderungen einer solchen Applikation.

3.9 Requirements / Anforderungen (Q31)

EN Assume that you are supposed to design a DB for managing the operations of a history museum. Give three examples for non-functional requirements of such an application.

DE Nehmen Sie an, dass Sie eine Datenbank zum Managen des Betriebs eines geschichtlichen Museums entwickeln sollen. Geben Sie drei Beispiele für nicht-funktionale Anforderungen einer solchen Applikation.

3.10 Requirements / Anforderungen (Q32)

EN Assume that you are supposed to design a DB for managing the operations of a high school. Give three examples for project constraints of such an application.

DE Nehmen Sie an, dass Sie eine Datenbank zum Managen des Betriebs einer Schule der Sekundarstufe II, z. B. eines Gymansiums, entwickeln sollen. Geben Sie drei Beispiele für Projektgrenzen einer solchen Applikation.

3.11 Stakeholders (Q33)

EN A stakeholder is anybody that is concerned with or, in any form, touched by or related to a project, is a user of or has a professional interest in a project. Assume that you got the task to develop a database for managing the day-to-day processes in a big hospital with several thousands of patients per day. The goal is the complete digitalization of all processes in the hospital. Name and discuss six groups of stakeholders in this project. For each of them, explain why they are stakeholders and why their opinions and suggestions should be considered during the development process.

DE Als *Stakeholder* wird jede Person bezeichnet, die irgendwie mit einem Projekt in Berührung kommt oder damit in Zusammenhang steht; also jeder, der damit etwas zu tun hat, es verwendet, oder einen professionellen Bezug zum Projekt hat. Nehmen Sie an, dass Sie die Aufgabe bekommen haben, eine Datenbank zum Managen aller täglicher Prozesse eines großen Krankenhauses mit mehreren tausend Patienten pro Tag zu entwickeln. Das Ziel ist die komplette Digitalisierung von allen Projekten des Krankenhauses. Nennen und diskutieren Sie sechs Gruppen von Stakeholders in dem Projekt. Für jede Gruppe, erklären Sie warum sie Stakeholders sind und warum ihre Meinungen und Vorschläge während des Entwicklungsprozesses berücksichtigt werden sollten.

3.12 Stakeholders (Q34)

EN A stakeholder is anybody that is concerned with or, in any form, touched by or related to a project, is a user of or has a professional interest in a project. Assume that you got the task to develop a database for managing the day-to-day processes in a big supermarket with several thousands of customers per day. The goal is the complete digitalization of all processes in the supermarket. Name and discuss five groups of stakeholders in this project. For each of them, explain why they are stakeholders and why their opinions and suggestions should be considered during the development process.

DE Als *Stakeholder* wird jede Person bezeichnet, die irgendwie mit einem Projekt in Berührung kommt oder damit in Zusammenhang steht; also jeder, der damit etwas zu tun hat, es verwendet, oder einen professionellen Bezug zum Projekt hat. Nehmen Sie an, dass Sie die Aufgabe bekommen haben, eine Datenbank zum Managen aller täglicher Prozesse eines großen Supermarkts mit mehreren tausend Kunden pro Tag zu entwickeln. Das Ziel ist die komplette Digitalisierung von allen Projekten des Supermarkts. Nennen und diskutieren Sie fünf Gruppen von Stakeholders in dem Projekt. Für jede Gruppe, erklären Sie warum sie Stakeholders sind und warum ihre Meinungen und Vorschläge während des Entwicklungsprozesses berücksichtigt werden sollten.

3.13 Requirements / Anforderungen (Q35)

EN Explain four methods that can be used to gather and understand the requirements of a DB project.

DE Erklären Sie vier Methoden, die benutzt werden können, um die Anforderungen eines Datenbankprojekts zu sammeln und zu verstehen.

Chapter 4

Conceptual Modeling / Konzeptuelle Modellierung

4.1 Basics / Grundlagen (Q36)

EN What is an *entity* and what is an *entity type* in the conceptual modeling stage? What is the difference between them? What real-world things do they represent?

DE Was ist eine *Entität* und was ist ein *Entitätstyp* in der konzeptuellen Modellierung? Was ist der Unterschied zwischen ihnen? Welche Dinge der realen Welt repräsentieren sie?

4.2 Basics / Grundlagen (Q37)

EN What are *attributes* in the conceptual modeling stage? What real-world concepts do they represent?

DE Was sind *Attribute* in der konzeptuellen Modellierung? Welche Konzepte aus der realen Wert repräsentieren sie?

4.3 Keys / Schlüssel (Q38)

EN What are *keys* in the conceptual model?

DE Was sind *Schlüssel* im konzeptuellen Modell?

4.4 Keys / Schlüssel (Q39)

EN What are *(candidate) keys* and *primary keys* in the conceptual model? What is the difference between them?

DE Was sind *Schlüssel(kandidaten)* und *Primärschlüssel* im konzeptuellen Modell? Was ist der Unterschied zwischen beiden?

4.5 Entities and Attributes / Entitäten und Attribute (Q40)

EN List the entities, entity types, and attributes in the following text:
"The province Anhui has an area of 140 100 km^2 and a population of over 61 million people. The city Hefei has a GDP of over 180 billion USD an is one of the science centers of China. Hefei has several good universities, including USTC, HFUT, Hefei University, and Anhui University."

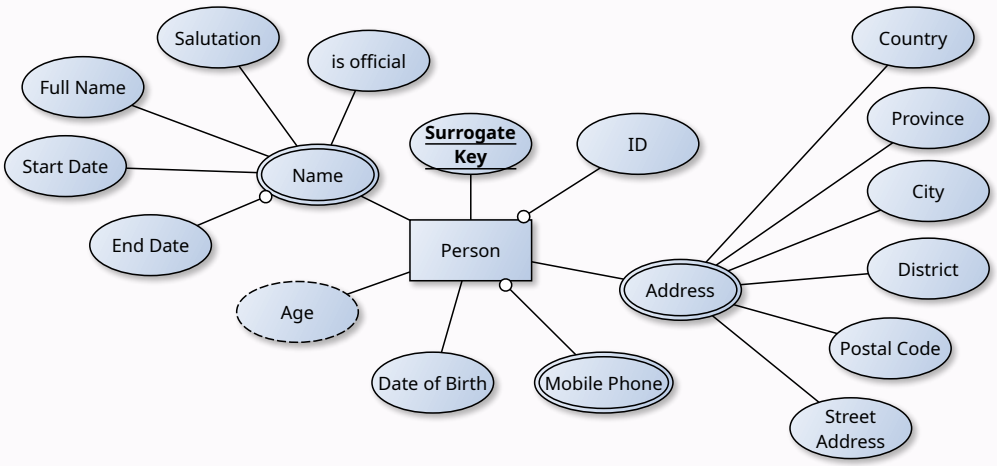
DE Geben Sie die Entitäten, Entitätstypen und Attribute aus dem folgenden Text an:
"Die Provinz Anhui hat eine Fläche von 140 100 km^2 und eine Bevölkerung von über 61 Million Menschen. Die Stadt Hefei hat ein Bruttosozialprodukt von über 180 Milliarden USD und ist eines der Wissenschaftszentren von China. Hefei hat mehrere gute Universitäten, wie z. B. die USTC, HFUT, die Universität Hefei und die Anhui Universität."

4.6 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q41)

EN Express the following relationship with Crow's Foot notation: "Every person has at least one address. An arbitrary number (including 0) of people can live at an address."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jede Person hat mindestens eine Adresse. Eine beliebige Anzahl (auch 0) Leute können unter einer Adresse wohnen."

4.7 ERDs (Q42)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

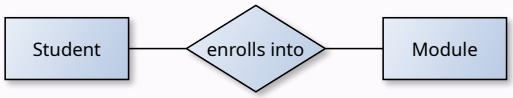
DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.8 Simple Single-Valued Attribute / Einfaches Einwertiges Attribut (Q43)

EN Explain what a simple single-valued attribute is in the entity-relationship model. Give an example.

DE Erklären Sie, was ein einfaches einwertiges Attribute in einem Entity-Relationship-Modell ist. Geben Sie ein Beispiel.

4.9 ERDs (Q44)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.10 Office ERD / Büros ERD (Q45)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: Every office has a room number. Every employee has a name, title, and worker's number. Every office offers some work stations, each of which having a station number. Every employee occupies one of the work stations, starting from a certain date. Every office has one or multiple telephones, each with a certain phone number.

DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Jedes Büro hat eine Zimmernummer. Jeder Mitarbeiter hat einen Name, Titel und Arbeiternummer. Jedes Büro bietet eine Menge von Arbeitsplätzen, die jeweils eine Platznummer haben. Jeder Mitarbeiter sitzt ab einem gewissen Datum an einem der Arbeitsplätze. In jedem Büro gibt es ein or mehrere Telefone mit jeweils einer festen Telefonnummer.

4.11 Bad Practices / Schleche Praxis (Q46)

EN Why is it often a bad idea to model addresses as single text strings? Give one idea on how to model them better.

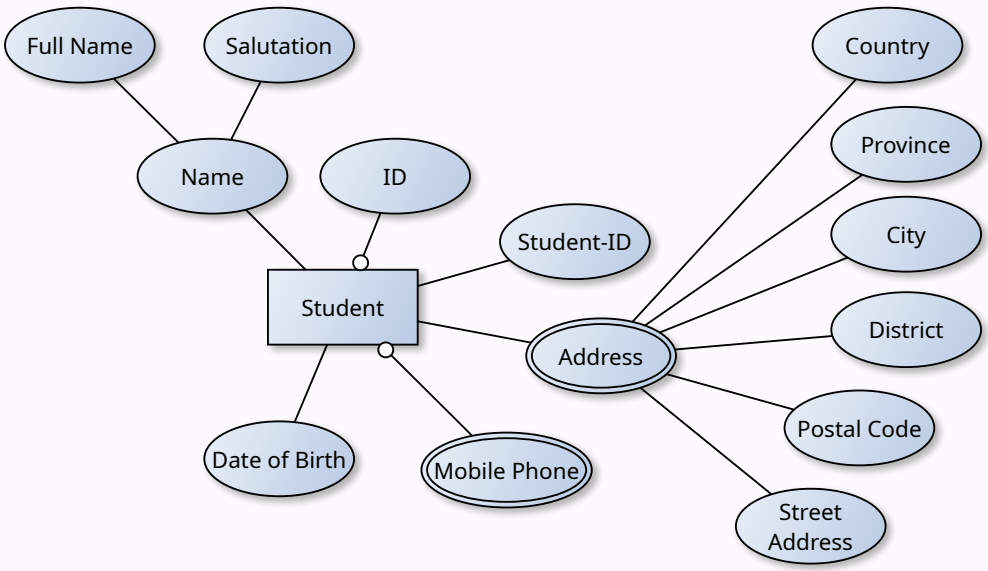
DE Warum ist es oft eine schlechte Idee, Adressen als einfache Zeichenketten zu modellieren? Machen Sie einen Vorschlag, wie man sie besser modellieren kann.

4.12 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q47)

EN Express the following relationship with Crow's Foot notation: "Every bank account belongs to exactly one person. Each person in the bank database has at least one bank account."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jedes Bankkonto gehört zu genau einer Person. Jede Person in der Bank-Datenbank hat mindestens ein Bankkonto."

4.13 ERDs (Q48)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.14 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q49)

EN Explain the meaning of the A ⟊—⟊ B relationship pattern. Give one example.

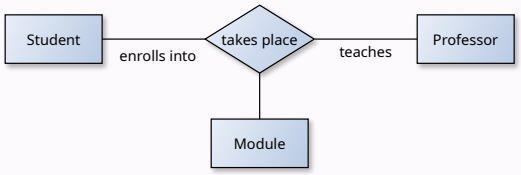
DE Erklären Sie die Bedeutung des Beziehungsformats A ⟊—⟊ B. Geben Sie ein Beispiel.

4.15 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q50)

EN Explain the meaning of the S ⩾—|< T relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats S ⩾—|< T. Geben Sie ein Beispiel.

4.16 ERDs (Q51)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.17 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q52)

EN Explain the meaning of the M ||—|< N relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats M ||—|< N. Geben Sie ein Beispiel.

4.18 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q53)

EN Explain the meaning of the O ⩾o—o< P relationship pattern. Give one example.

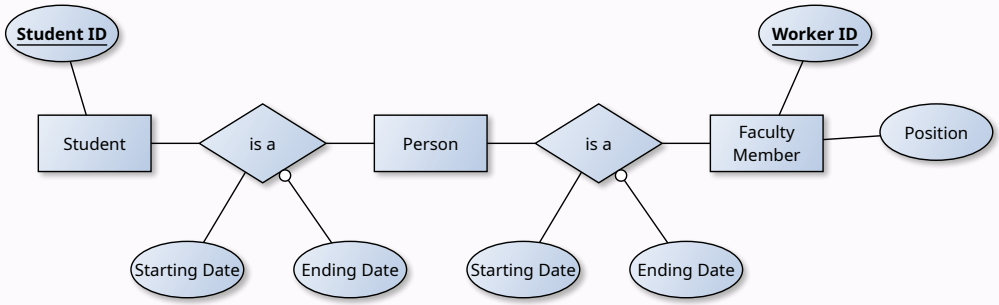
DE Erklären Sie die Bedeutung des Beziehungsformats O ⩾o—o< P. Geben Sie ein Beispiel.

4.19 Compositie Multi-Valued Attribute / Zusammengesetztes Mehrwertiges Attribut (Q54)

EN Explain what a compositve multi-valued attribute is in the entity-relationship model. Give an example.

DE Erklären Sie, was ein zusammengesetztes mehrwertiges Attribute in einem Entity-Relationship-Modell ist. Geben Sie ein Beispiel.

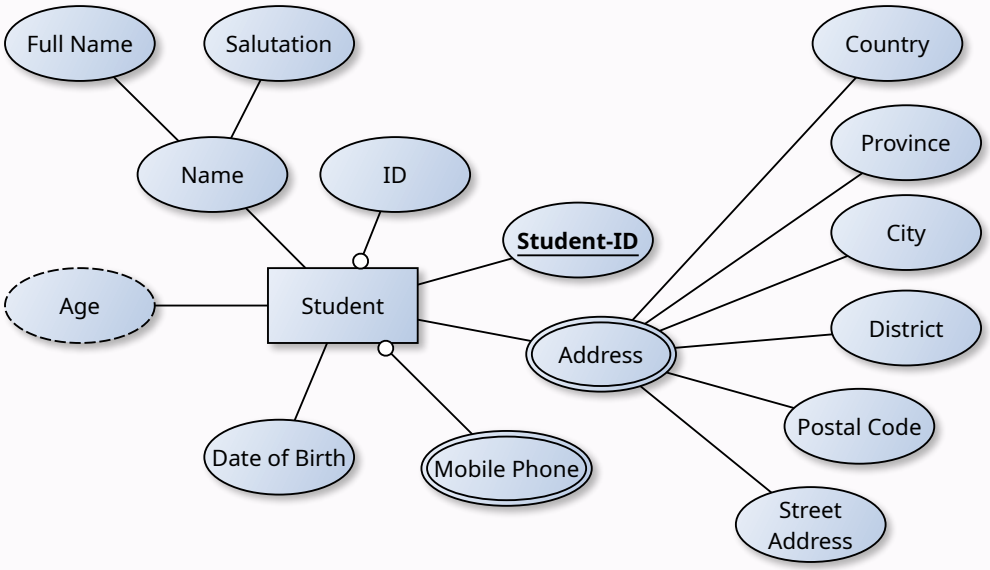
4.20 ERDs (Q55)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.21 ERDs (Q56)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.22 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q57)

EN Express the following relationship with Crow's Foot notation: "A primary school student can either be subscribed to a meal plan or not. To a meal plan, an arbitrary number of primary school student can be subscribed."

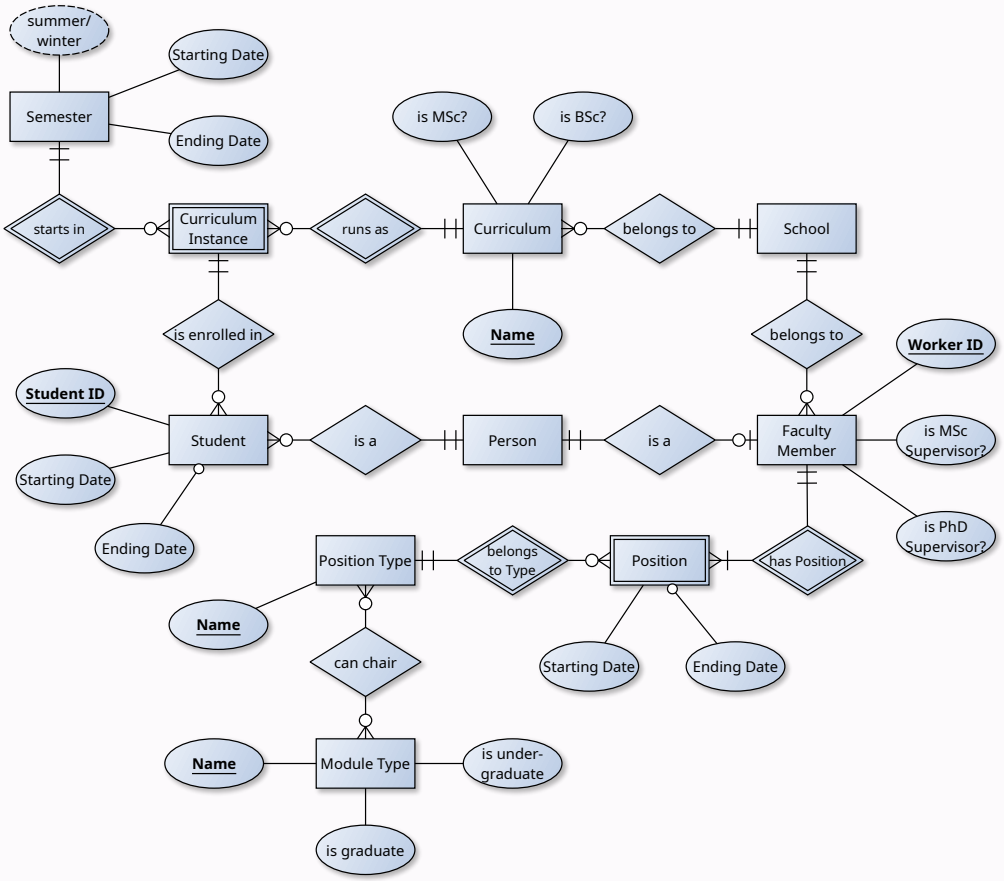
DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Ein Grundschüler kann entweder einen Essensplan aboniert haben oder nicht. Jeder Essensplan kann von einer beliebigen Anzahl von Grundschülern aboniert werden."

4.23 Furniture ERD / Möbel ERD (Q58)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: A furniture store sells furniture. Each type of furniture has a type-ID, a name, and a price. It can be purchased and sold in variants, which are characterized by a name, color, and wood pattern. It is produced by a vendor, which is characterized by a vendor-ID, a name, and an address. The store has several warehouses, each of which has a name and stores amounts for different variant of different furniture types. The company has salespeople (employee-ID, name, date of birth, date of hiring, salary). A salesperson can acquire and be responsible for arbitrarily many customers (customer-ID, name, address). The customers can issue purchase orders, characterized by an order-ID, furniture variant, amount, and deadline.

DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Ein Möbelladen verkauft Möbel. Jeder Möbel-Typ hat eine Typ-ID, einen Namen und einen Preis. Möbel eines Types kann in verschiedenen Varianten gekauft und verkauft werden, welche jeweils einen Namen, eine Farbe und eine Holzmaserung aufweisen. Ein Möbeltyp wird von einem Anbieter hergestellt, der eine Anbieter-ID, einen Namen und eine Adresse hat. Jede der Lagerhalle des Möbelladens hat einen Namen und kann Anzahlen verschiedener Varianten verschiedener Möbeltypen speichern. Der Laden hat Verkäufer (Verkäufer-ID, Name, Geburtsdatum, Einstellungsdatum, Gehalt). Ein Verkäufer kann beliebig viele Kunden (Kunden-ID, Name, Adresse) anwerben und für diese verantwortlich sein. Ein Kunde kann Bestellungen aufgeben, die jeweils eine Bestellungs-ID, Variante eines Möbeltyps, Anzahl und Deadline haben.

4.24 ERDs (Q59)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.25 Bad Practices / Schleche Praxis (Q60)

EN Explain an example szenario where using the telephone number as a primary key for identifying people in a database can be a problem.

DE Geben Sie ein Beispielszenario an, wo das Verwenden der Telefonnummer als Primärschlüssel um Personen zu identifizieren problematisch werden kann.

4.26 Weak Entities / Schwache Entitäten (Q61)

EN Explain what a weak entity is in the entity-relationship model. Why do we need weak entities? Give an example.

DE Erklären Sie, was eine schwache Entität in einem Entity-Relationship-Modell ist. Wofür brauchen wir schwachte Entitäten? Geben Sie ein Beispiel.

4.27 Soccer ERD / Fußball ERD (Q62)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: We develop a soccer DB. Every player has a name and age. Ever game takes place on a certain game day. Every team has a name and a hometown. In each game, one team participates as home team and one as guest team. We store which player plays in which game, together with the starting and ending minute of their participation. Every team has a trainer, of whom we store name and age.

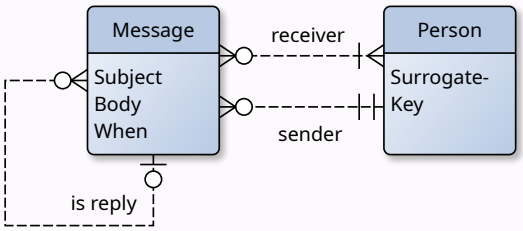
DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Es wird eine Fußballdatenbank entwickelt. Jeder Spieler hat einen Namen und ein Alter. Jedes Spiel findet an einem Spieltag statt. Jede Mannschaft hat einen Namen und einen Heimatort. An jedem Spiel nimmt eine Mannschaft als Heimteam und eine Mannschaft als Gast teil. Es wird gespeichert, an welchem Spiel welcher Spieler teilgenommen hat und zwar jeweils die Anfangs- und die Ende-Minute seines Einsatzes. Jede Mannschaft hat einen Trainer, dessen Name und Alter gespeichert werden.

4.28 ERDs (Q63)

EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

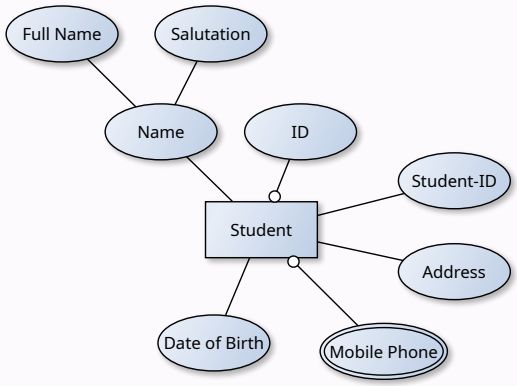
4.29 ERDs (Q64)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.30 ERDs (Q65)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.31 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q66)

EN Explain the meaning of the C ⟊—‖ D relationship pattern. Give one example.

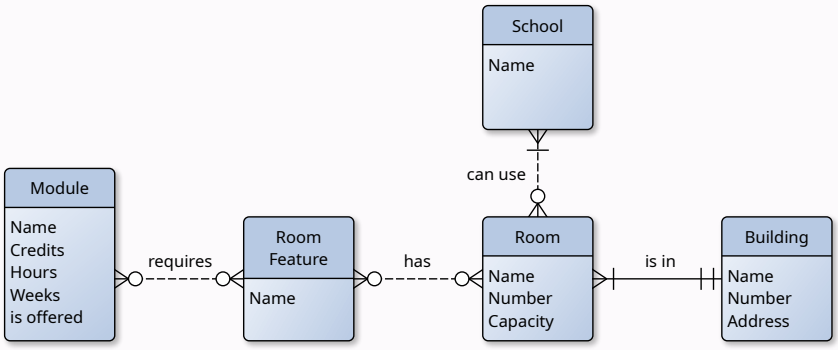
DE Erklären Sie die Bedeutung des Beziehungsformats C ⟊—‖ D. Geben Sie ein Beispiel.

4.32 Airline ERD / Fluggesellschaft ERD (Q67)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: Every airline has a name, home country, hometown, and abbreviation. An airplane has an airplane number and date of last inspection. Each airplane belongs to a given type, which is described by its name, number of seats, and top speed. Each airplane also belongs to a given airline since a certain date. Each pilot has a name, pilot number, qualification level, and flight hours. Each pilot works for a given airline since a specific date. Passengers are characterized by a passenger number, name, data of birth, and address. Flights have flight number, date, start airport, destination airport, and flight time. Each passenger can book an arbitrary number of such flights. A flight is realized by a given pilot and airplane.

DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Jede Fluggesellschaft hat einen Namen, ein Heimatland, eine Heimatstadt und eine Abkürzung. Jedes Flugzeug hat eine Flugzeugnummer und ein Datum der letzten Inspektion. Jedes Flugzeug hat einen bestimmten Typ, der durch seinen Name, die Anzahl der Sitzplätze und die Höchstgeschwindikeit charakterisiert wird. Jedes Flugzeug gehört zu einer Fluggesellschaft ab einem bestimmten Datum. Jeder Pilot hat einen Namen, eine Pilotennummer, eine Qualifikation und eine Anzahl von Flugstunden. Jeder Pilot arbeitet für eine bestimmte Fluggesellschaft ab einem bestimmten Datum. Passagiere werden charakterisiert durch eine Passagiernummer, Name, Geburtsdatum und Adresse. Flüge haben Flugnummern, Datum, Startflughafen, Zielflughafen und eine Flugzeit. Jeder Passagier kann eine beliebge Anzahl Flüge buchen. Ein Flug wird von einem bestimmten Pilot und Flugzeug realisiert.

4.33 ERDs (Q68)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

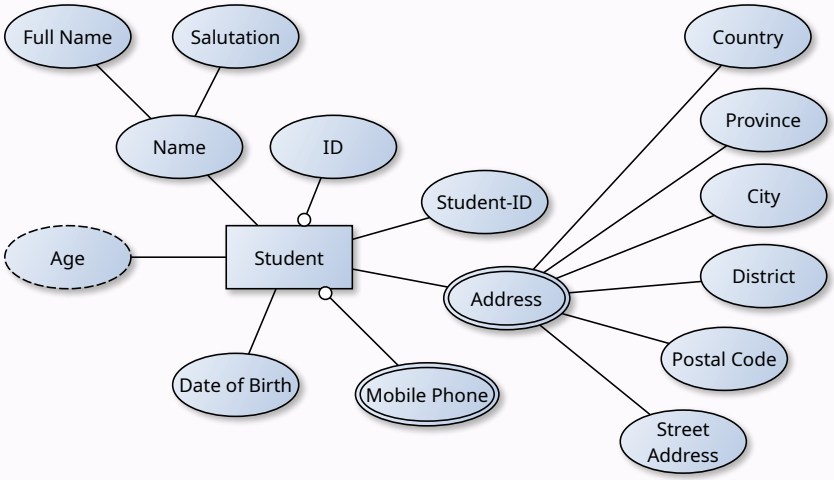
DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.34 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q69)

EN Explain the meaning of the K ||—o< L relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats K ||—o< L. Geben Sie ein Beispiel.

4.35 ERDs (Q70)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.36 Bad Practices / Schleche Praxis (Q71)

EN Why is it generally a bad idea to assume that names can be primary keys for identifying people? Give an example when this becomes problematic.

DE Warum ist es prinzipiell eine schlechte Idee, Namen als Primärschlüssel zum Identifizieren von Leuten zu verwenden? Geben Sie ein Beispielszenario, wo das schief geht.

4.37 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q72)

EN Explain the meaning of the G o—|< H relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats G o—|< H. Geben Sie ein Beispiel.

4.38 Company ERD / Firma ERD (Q73)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: Every work group has a group number, name, and location. Every employee has a worker's number, name, date of birth, salary, address, and job. Every employee belongs to a work group. Each project has a name and deadline. Every employee works on at least one, but maybe multiple projects. Each project has an employee as a leader.

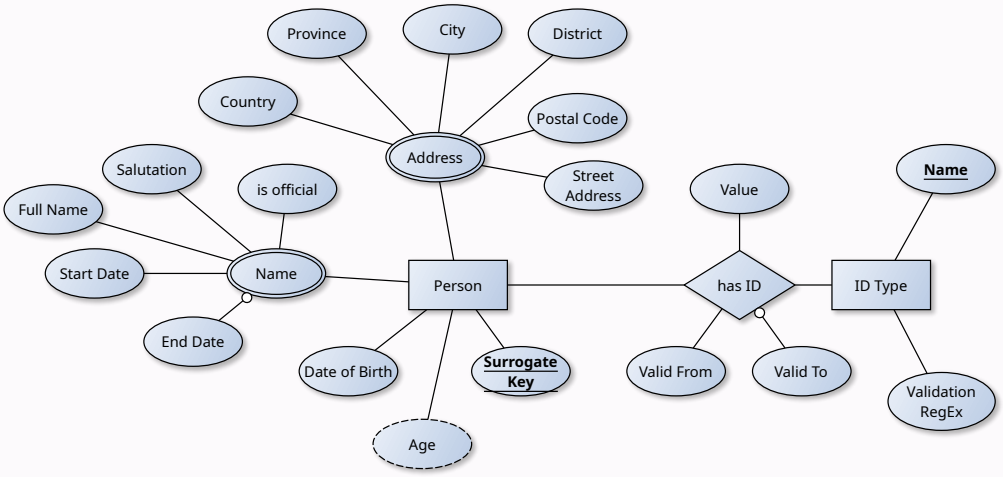
DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Jede Arbeitsgruppe hat eine Gruppennummer, Name und Ort. Jeder Mitarbeiter hat eine Arbeiternummer, Name, Geburtsdatum, Gehalt, Adresse und Arbeitsaufgabe. Jeder Mitarbeiter gehört zu einer Arbeitsgruppe. Jedes Projekt hat einen Namen und eine Deadline. Jeder Mitarbeiter arbeitet an mindestens einem, vielleicht aber auch mehreren Projekten. Jedes Projekt hat einen Mitarbeiter als Leiter.

4.39 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q74)

EN Explain the meaning of the E ┼○─○≪ F relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats E ┼○─○≪ F. Geben Sie ein Beispiel.

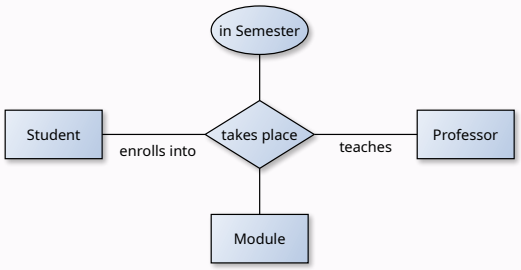
4.40 ERDs (Q75)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.41 ERDs (Q76)



EN Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

DE Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

4.42 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q77)

EN Express the following relationship with Crow's Foot notation: "Every student is enrolled into at least one lecture. Into each lecture, at least one student is enrolled."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jeder Student ist in mindestens eine Vorlesung eingeschrieben. In jede Vorlesung ist mindestens ein Student eingeschrieben."

4.43 Simple Multi-Valued Attribute / Einfaches Mehrwertiges Attribut (Q78)

EN Explain what a simple multi-valued attribute is in the entity-relationship model. Give an example.

DE Erklären Sie, was ein einfaches mehrwertiges Attribute in einem Entity-Relationship-Modell ist. Geben Sie ein Beispiel.

4.44 Sports ERD / Sportarten ERD (Q79)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: Every sports discipline has a name and a world, asia-, and olympic record. There are competitions which have a competition number, date, location, start time, and type. The competition type has a name, e.g., Asian Championship, Olympics, etc. Every athlete has a name, age, sports club, and home country. Every competition offers a set of sports disciplines and athletes can take part in multiple disciplines in multiple competitions. Each athlete has a trainer, for whom we store the qualification, name, and date of birth. A trainer can train arbitrarily many athletes.

DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Jede Sportdisziplin hat einen Namen und jeweils eine Welt-, Asien- und Olympiarekord. Es werden Wettkämpfe ausgetragen, die jeweils eine Wettkampfnummer, Datum, Ort, Uhrzeit und Art haben. Jede Wettkampfart hat einen Namen, z. B. Asienmeisterschaft, Olympiade, usw. Jeder Sportler hat einen Namen, Alter, Sportklub und Heimatland. Jeder Wettkampf kann bestimmte Sportarten anbieten und Sportler können an Wettkämpfen teilnehmen und zwar jeweils in beliebig vielen Disziplinen. Jeder Sportler hat einen Trainer, über den wir den Name, die Qualifikation und das Geburtsdatum speichern.Ein Trainer kann beliebig viele Sportler betreuen.

4.45 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q80)

EN Explain the meaning of the Q ⩾○—⊣⩽ R relationship pattern. Give one example.

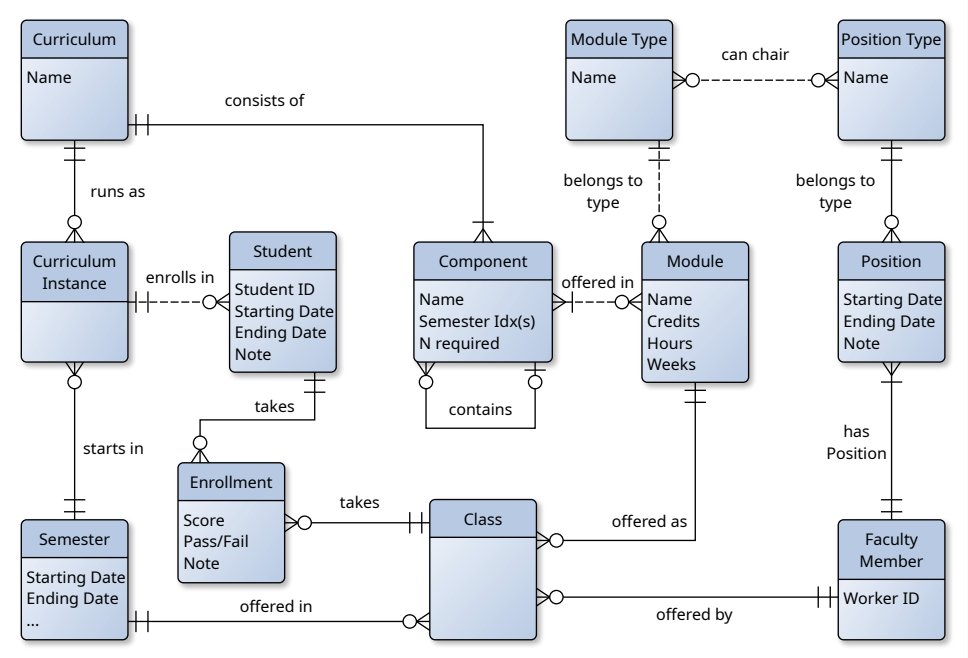
DE Erklären Sie die Bedeutung des Beziehungsformats Q ⩾○—⊣⩽ R. Geben Sie ein Beispiel.

4.46 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q81)

EN Express the following relationship with Crow's Foot notation: "Every cook has at least one cook's hat. Each cook's hat belongs to exactly one cook."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jede Koch hat mindestens eine Kochmütze. Jede Kochmütze gehört genau einem Koch."

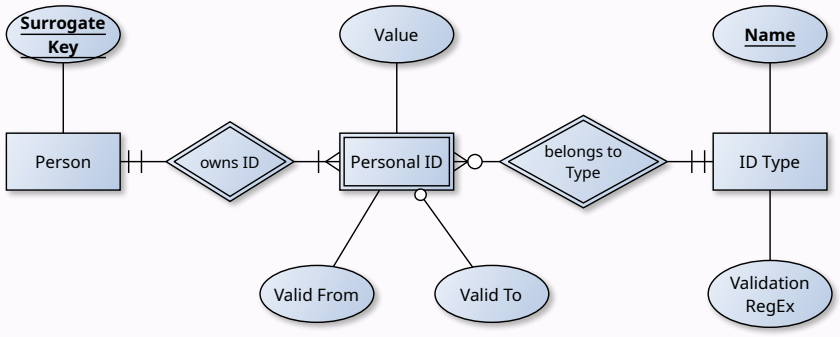
4.47 ERDs (Q82)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.48 ERDs (Q83)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.49 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q84)

EN Express the following relationship with Crow's Foot notation: "Every email can either be new or the answer to an existing email. Each email can be answered an arbitrary number of times."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jede Email ist entweder neu oder ist die Antwort auf eine existierende Email. Jede Email kann beliebig oft beantwortet werden."

4.50 Hospital ERD / Krankenhaus ERD (Q85)

EN Design an Entity-Relationship Model (using the original notation with rectangles, diamonds, and ellipses) for this scenario: The work of a hospital should be stored and represented by a database. Each patient has a patient-ID, a name, a date of birth, an address, an entry date, and exit date, and a diagnosis. Each medicine has a name, a main chemical compound, and a producer. Each department has a name and a specialization. Each doctor has a doctor-ID, a name, and a phone number. Each doctor belongs to a department. Each room belongs to a department, has a room number, a number of beds, and a phone number. Each patient belongs to a room. Each patient is treated by one or multiple doctors. Each patient gets a specific dosage (number of administrations per day, amount per administration) of one or multiple medicines.

DE Entwerfen Sie ein Entity-Relationship Modell (in der Originalnotation mit Rechtecken, Rauten und Ellipsen) für folgende Situation: Die Arbeit eines Krankenhauses soll in einer Datenbank repräsentiert werden. Jeder Patient hat eine Patienten-ID, einen Namen, ein Geburtsdatum, eine Adresse, ein Einlieferungsdatum, ein Entlassungsdatum und eine Diagnose. Jede Medizin hat einen Namen, einen Hauptwirkstoff und einen Produzenten. Jede Abteilung hat einen Namen und eine Spezialisierung. Jeder Arzt hat eine Arzt-ID, einen Namen und eine Telefonnummer. Jeder Arzt gehört zu einer Abteilung. Jeder Raum gehört zu einer Abteilung und hat eine Raumnummer, eine Anzahl von Betten und eine Telefonnummer. Jeder Patient gehört zu einem Raum. Jeder Patient kann von einem oder mehreren Ärzten behandelt werden. Jeder Patient bekommt eine bestimmte Dosierung (Anzahl der Anwendungen pro Tag, Menge pro Anwendung) von einer oder mehrerer Medizinen.

4.51 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q86)

EN Express the following relationship with Crow's Foot notation: "A trainer trains at least one tennis player. Each tennis player either has one trainer or no trainer."

DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jeder Trainer trainiert mindestens einen Tennisspieler. Jeder Tennisspieler hat mindestens einen Trainer."

4.52 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q87)

EN Express the following relationship with Crow's Foot notation: "Each vehicle has at least one wheel. A wheel is attached to exactly one vehicle (or stored somewhere)."

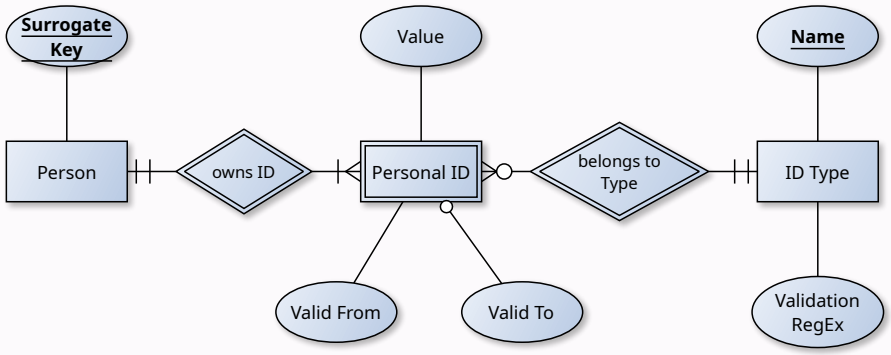
DE Drücken Sie die folgende Beziehung mit der Krähenfußnotation aus: "Jedes Fahrzeug hat mindestens ein Rad. Jedes Rad ist mit genau einem Fahrzeug verbunden (oder wird irgendwo gelagert.)"

4.53 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q88)

EN Explain the meaning of the I ╫──╫ J relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats I ╫──╫ J. Geben Sie ein Beispiel.

4.54 ERDs (Q89)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

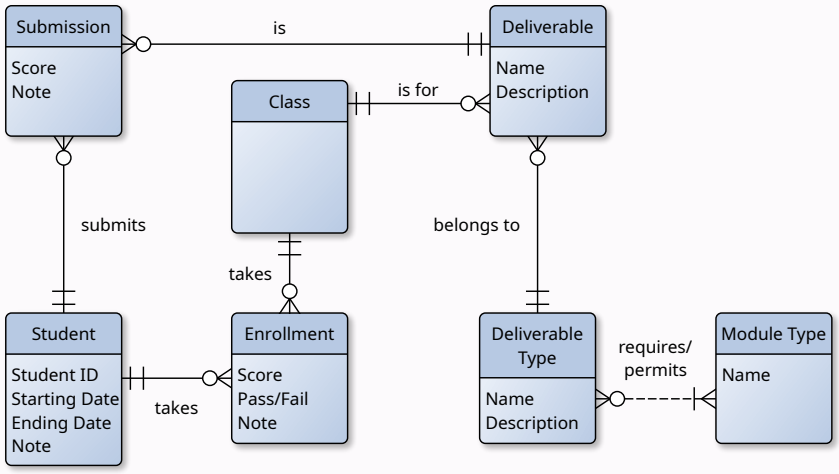
DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.55 Compositie Single-Valued Attribute / Zusammengesetztes Einwertiges Attribut (Q90)

EN Explain what a compositve single-valued attribute is in the entity-relationship model. Give an example.

DE Erklären Sie, was ein zusammengesetztes einwertiges Attribute in einem Entity-Relationship-Modell ist. Geben Sie ein Beispiel.

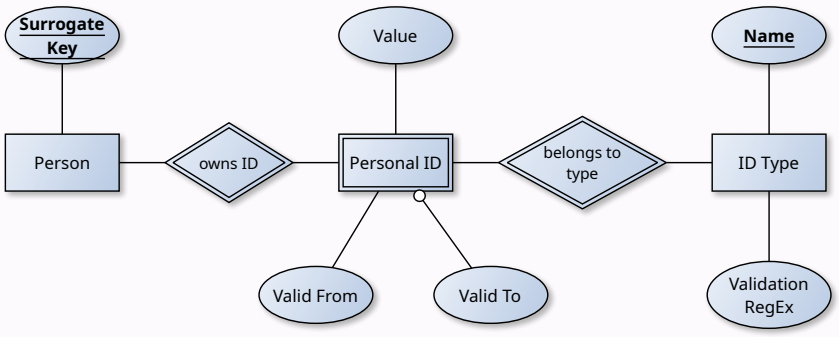
4.56 ERDs (Q91)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

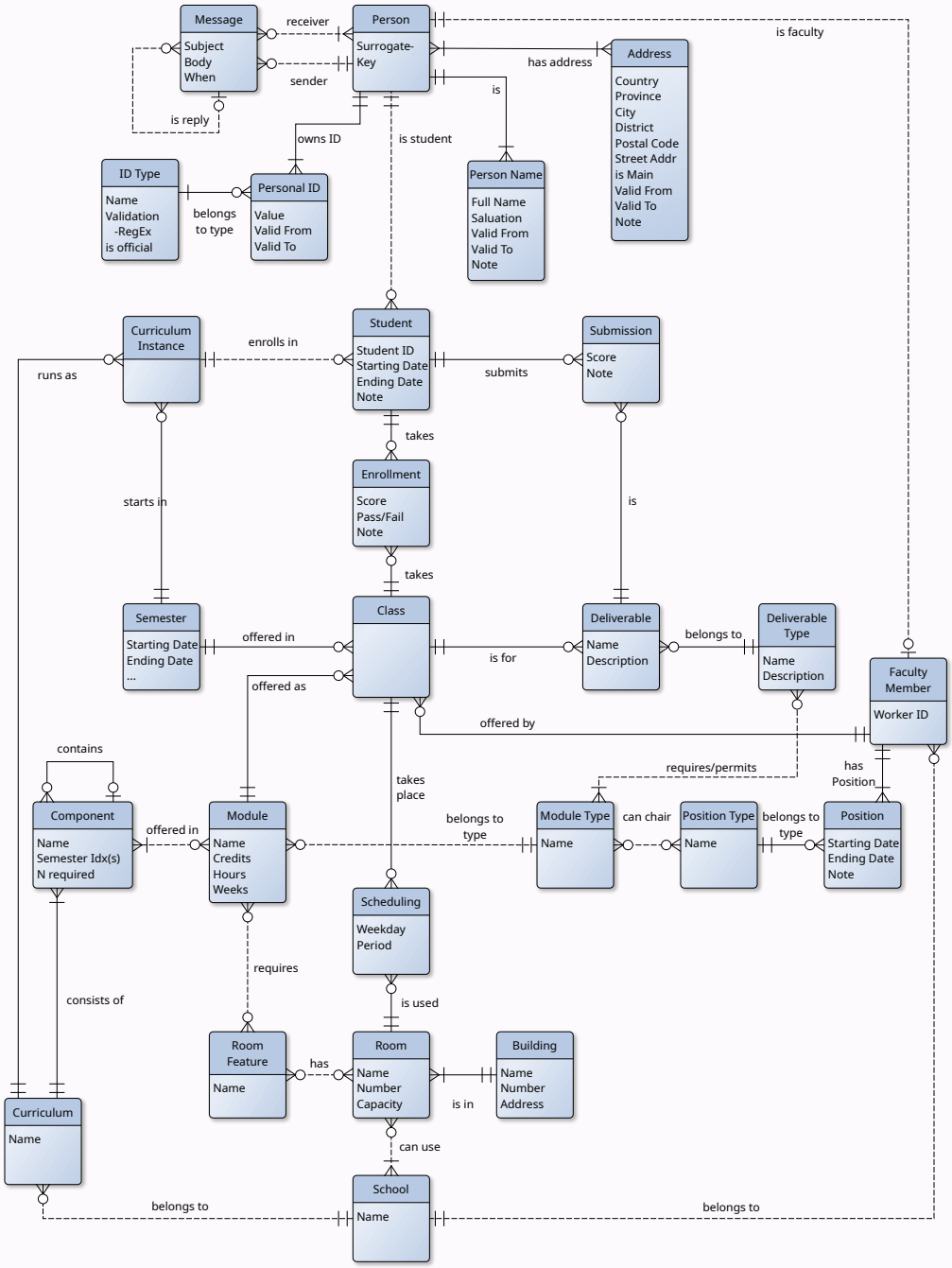
4.57 ERDs (Q92)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.58 ERDs (Q93)



EN Explain all entity types, attributes, primary keys, relationships, as well as relationship cardinalities and -modalities (as far as defined) in the ERD above.

DE Erklären Sie alle Entitätstypen, Attribute, Primärschlüssel, Beziehungen, sowie Beziehungskardinalitäten und -modalitäten, soweit vorhanden, in dem ERD oben.

4.59 Weak and Strong Entities / Schwache und Starke Entitäten (Q94)

EN Explain what weak and strong entities are in the entity-relationship model. What is their difference? Why do we need weak entities? Give an example for each.

DE Erklären Sie, was schwache und starke Entitäten in einem Entity-Relationship-Modell sind. Was ist der Unterschied zwischen beiden? Wofür brauchen wir schwachte Entitäten? Geben Sie jeweils ein Beispiel.

4.60 Relationship Cardinality and Modality / Beziehungskardinalität und -modalität (Q95)

EN Explain the meaning of the A ┼○─○┼ B relationship pattern. Give one example.

DE Erklären Sie die Bedeutung des Beziehungsformats A ┼○─○┼ B. Geben Sie ein Beispiel.

Chapter 5

Logical Modeling / Logische Modellierung

5.1 Entity Types / Entitätstypen (Q96)

EN How will *entity types* in the conceptual model be represented in a (relational) logical model?

DE Wie werden *Entitätstypen* aus dem konzeptuellen Modell im (relationalen) logischen Modell dargestellt?

5.2 Attributes / Attribute (Q97)

EN How will *simple single-valued attributes* in the conceptual model be represented in a (relational) logical model?

DE Wie werden *einfache einwertige Attribute* aus dem konzeptuellen Modell im (relationalen) logischen Modell dargestellt?

5.3 Attributes / Attribute (Q98)

EN How will *simple multi-valued attributes* in the conceptual model be represented in a (relational) logical model?

DE Wie werden *einfache mehrwertige Attribute* aus dem konzeptuellen Modell im (relationalen) logischen Modell dargestellt?

5.4 Attributes / Attribute (Q99)

EN How will *composite single-valued attributes* in the conceptual model be represented in a (relational) logical model?

DE Wie werden *zusammengesetzte einwertige Attribute* aus dem konzeptuellen Modell im (relationalen) logischen Modell dargestellt?

5.5 Attributes / Attribute (Q100)

EN How will *composite multi-valued attributes* in the conceptual model be represented in a (relational) logical model?

DE Wie werden *zusammengesetzte mehrwertige Attribute* aus dem konzeptuellen Modell im (relationalen) logischen Modell dargestellt?

5.6 Modelling / Modellierung (Q101)

EN Assume that the relationship between listeners and songs follows the schema listener ⩾○—○⩽ song. Every listener has a name and age. Every song has a title, singer, and duration. Which tables, attributes, and constraints do you need to represent this situation correctly in the relational model?

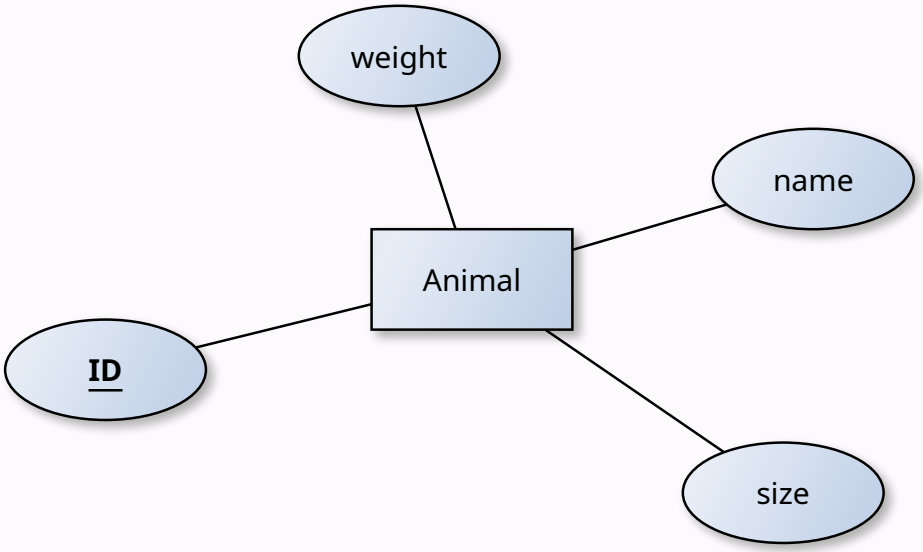
DE Nehmen Sie an, dass die Beziehung zwischen Zuhörern und Songs dem Schema Zuhörer ⩾○—○⩽ Song entspricht. Jede Zuhörer hat einen Namen und ein Alter. Jeder Song hat einen Titel, Sänger und eine Dauer. Welche Tabellen, Attribute und Einschränkungen brauchen Sie, um dieses Situation im relationalen Modell korrekt darzustellen?

5.7 Surrogate Keys / Ersatzschlüssel (Q102)

EN What is a *surrogate key*? Why does it often make sense to use one?

DE Was ist ein *Ersatzschlüssel*? Warum ist es oft sinnvoll, einen zu verwenden?

5.8 Conceptual to Logical / Konzeptuell zu Logisch (Q103)



EN Transform the ERD above to a logical model obeying the relational data model. Explain which tables, columns, and constraints you would create.

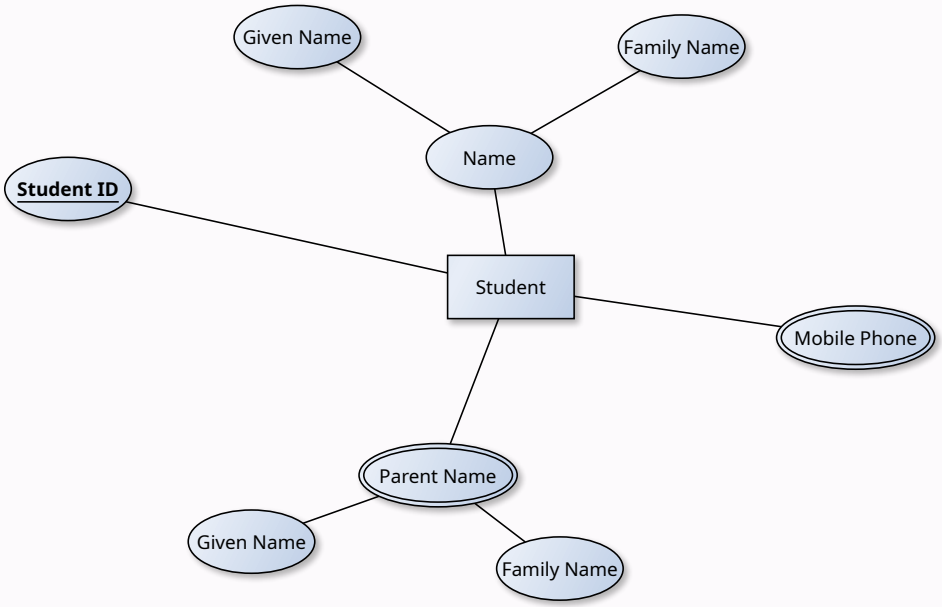
DE Übertragen Sie das ERD oben zu einem logischen Modell unter dem relationalen Datenmodell. Erklären Sie, welche Tabellen, Spalten und Einschränkungen Sie erstellen würden.

5.9 Modelling / Modellierung (Q104)

EN Assume that the relationship between tickets and spectators of a concert follows the schema ticket ||—o| spectator. Every ticket has a date and a price. Every spectator has a name. Which tables, attributes, and constraints do you need to represent this situation correctly in the relational model?

DE Nehmen Sie an, dass die Beziehung zwischen Eintrittskarten und Besuchern eines Konzerts dem Schema Eintrittskarte ||—o| Besucher entspricht. Jede Eintrittskarte hat ein Datum und einen Preis. Jeder Besucher hat einen Namen. Welche Tabellen, Attribute und Einschränkungen brauchen Sie, um dieses Situation im relationalen Modell korrekt darzustellen?

5.10 Conceptual to Logical / Konzeptuell zu Logisch (Q105)



EN Transform the ERD above to a logical model obeying the relational data model. Explain which tables, columns, and constraints you would create.

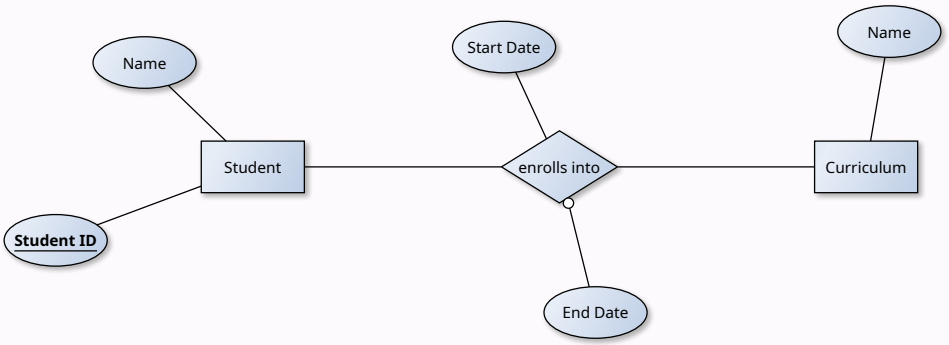
DE Übertragen Sie das ERD oben zu einem logischen Modell unter dem relationalen Datenmodell. Erklären Sie, welche Tabellen, Spalten und Einschränkungen Sie erstellen würden.

5.11 Modelling / Modellierung (Q106)

EN Assume that the relationship between salespersons and customers of a furniture store follows the schema customer ⟩o—o+ salesperson. Every customer has a name and address. Every salesperson has a worker's number, name, and salary. Which tables, attributes, and constraints do you need to represent this situation correctly in the relational model?

DE Nehmen Sie an, dass die Beziehung zwischen Verkäufer und Kunden eines Möbelgeschäftes dem Schema Kunde ⟩o—o+ Verkäufer entspricht. Jede Eintrittskarte hat ein Datum und einen Preis. Jeder Besucher hat einen Namen. Welche Tabellen, Attribute und Einschränkungen brauchen Sie, um dieses Situation im relationalen Modell korrekt darzustellen?

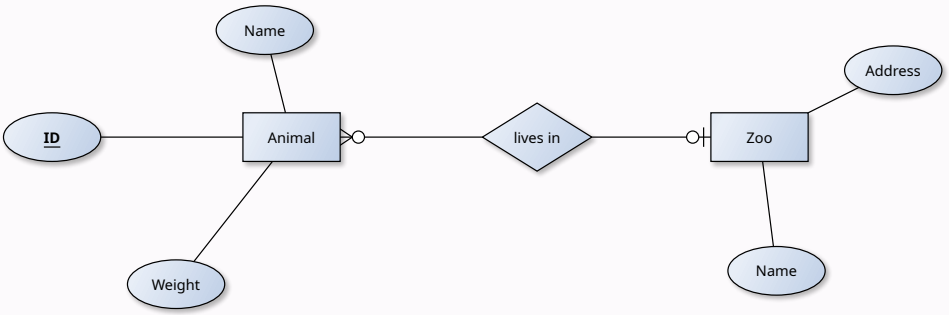
5.12 Conceptual to Logical / Konzeptuell zu Logisch (Q107)



EN Transform the ERD above to a logical model obeying the relational data model. Explain which tables, columns, and constraints you would create.

DE Übertragen Sie das ERD oben zu einem logischen Modell unter dem relationalen Datenmodell. Erklären Sie, welche Tabellen, Spalten und Einschränkungen Sie erstellen würden.

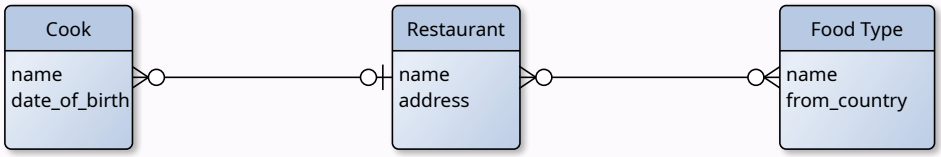
5.13 Conceptual to Logical / Konzeptuell zu Logisch (Q108)



EN Transform the ERD above to a logical model obeying the relational data model. Explain which tables, columns, and constraints you would create.

DE Übertragen Sie das ERD oben zu einem logischen Modell unter dem relationalen Datenmodell. Erklären Sie, welche Tabellen, Spalten und Einschränkungen Sie erstellen würden.

5.14 Conceptual to Logical / Konzeptuell zu Logisch (Q109)



EN Transform the ERD above to a logical model obeying the relational data model. Explain which tables, columns, and constraints you would create.

DE Übertragen Sie das ERD oben zu einem logischen Modell unter dem relationalen Datenmodell. Erklären Sie, welche Tabellen, Spalten und Einschränkungen Sie erstellen würden.

Chapter 6

SQL

6.1 General SQL / Generelles SQL

6.1.1 SQL (Q110)

EN What is SQL? For which kind of database is it used?

DE Was ist SQL? Für welche Art von Datenbank wird es benutzt?

6.1.2 Create a User / Benutzer Erstellen (Q111)

EN Write down the SQL (PostgreSQL) command for creating a new user "otto".

DE Schreiben Sie das SQL (PostgreSQL) Kommando, um einen neuen Benutzer "otto" zu erstellen.

6.1.3 Create a Database / Datenbank Erstellen (Q112)

EN Write down the SQL (PostgreSQL) command for creating a new database "production" for user "anna".

DE Schreiben Sie das SQL (PostgreSQL) Kommando, um eine neue Datenbank "production" für Benutzer "anna" zu erstellen.

6.1.4 Create a Table / Tabelle Erstellen (Q113)

EN Provide the general syntax and structure of the command for creating a table using SQL (PostgreSQL). Explain each element, such as table name, column names, datatypes, and constraints.

DE Geben Sie die generelle Syntax und Struktur des Kommandos zum Erstellen von Tabellen mit SQL (PostgreSQL) an. Erklären Sie jedes Element, wie Tabellenname, Spaltenname, Datentyp und Einschränkungen.

6.1.5 Datatypes / Datentypen (Q114)

EN Name AND explain four different datatypes of SQL (PostgreSQL). Provide one use case for each of these datatypes.

DE Nennen UND erklären Sie vier verschiedene Datentypen von SQL (PostgreSQL). Geben Sie jeweils eine Anwendung für jeden der dieser Datentypen an.

6.1.6 Create a Table / Tabelle Erstellen (Q115)

EN Provide the SQL (PostgreSQL) command to create a table "cars" with the following columns:

- "name": a character string of reasonable length (must be unique),
- "price": the costs of one such car in RMB (must always be provided),
- "top_speed": the maximum speed.

DE Geben Sie das SQL (PostgreSQL) Kommand an, um eine Tabelle "cars" mit den folgenden Spalten zu erstellen:

- "name": ein Text vernünftiger Länge, der eindeutig/einmalig sein muss,
- "price": die Kosten eines solchen Autos in RMB (muss immer angegeben werden),
- "top_speed": die Maximalgeschwindigkeit.

6.1.7 Constraints / Einschränkungen (Q116)

EN What is a `PRIMARY KEY` constraint? What is it used for?

DE Was ist eine `PRIMARY KEY`-Einschränkung? Wofür wird sie verwendet?

6.1.8 Command Understanding / Kommando Verstehen (Q117)

Listing 6.1: Das SQL (PostgreSQL) script `create_table_01.sql` (src)

```
1 CREATE TABLE food (
2     id            INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
3     name          VARCHAR(100)   NOT NULL UNIQUE,
4     is_vegetarian BOOLEAN        NOT NULL,
5     price         DECIMAL(10, 2) NOT NULL,
6     CONSTRAINT price_ok CHECK (price > 0)
7 );
```

EN What does the above SQL (PostgreSQL) command do? Explain each one of its lines.

DE Was macht das SQL (PostgreSQL) Kommando oben? Erklären Sie jede einzelne seiner Zeilen.

6.1.9 Command Understanding / Kommando Verstehen (Q118)

Listing 6.2: Das SQL (PostgreSQL) script select_01.sql (src)

```sql
SELECT name, customer_id, address
    FROM customer WHERE address LIKE '%Hefei%';
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.10 Constraints / Einschränkungen (Q119)

EN What is a `UNIQUE` constraint? What is it used for?

DE Was ist eine `UNIQUE`-Einschränkung? Wofür wird sie verwendet?

6.1.11 Command Understanding / Kommando Verstehen (Q120)

Listing 6.3: Das SQL (PostgreSQL) script insert_01.sql (src)

```sql
INSERT INTO team (name, mp, pts, last_game_on)
VALUES ('Shanghai Port',     27, 60, '2025-10-17'),
       ('Chengdu Rongcheng', 27, 58, '2025-10-21'),
       ('Shanghai Shenhua',  27, 57, '2025-10-22'),
       ('Qingdao Hainiu',    27, 18, '2025-10-17');
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist and which datatypes should they have for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren und welche Datentypen sollten sie haben, damit dieses Kommando funktioniert?

6.1.12 Command Understanding / Kommando Verstehen (Q121)

Listing 6.4: Das SQL (PostgreSQL) script select_02.sql (src)

```sql
SELECT player.name AS player_name, team.name AS team_name FROM player
    LEFT JOIN team ON (player.team = team.id)
    ORDER BY player_name, team_name;
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.13 Command Understanding / Kommando Verstehen (Q122)

Listing 6.5: Das SQL (PostgreSQL) script create_view_01.sql (src)

```sql
CREATE VIEW food_sales AS
    SELECT food.name as food_name, SUM(food.price * sale.amount) AS total
    FROM food INNER JOIN sale ON (sale.food = food.id)
    GROUP BY food_name ORDER BY food_name;
```

EN What does the above SQL (PostgreSQL) command do? Explain this in detail. Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Erklären Sie das im Detail. Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.14 Command Understanding / Kommando Verstehen (Q123)

Listing 6.6: Das SQL (PostgreSQL) script insert_02.sql (src)

```sql
INSERT INTO employee (name, task, since)
VALUES ('Bibbo Bobbson', 'goalkeeper', 2025),
       ('Bebbo Bebbenheimer', 'defender', 2025);
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.15 Constraints / Einschränkungen (Q124)

EN What is a `CHECK` constraint? What is it used for?

DE Was ist eine `CHECK`-Einschränkung? Wofür wird sie verwendet?

6.1.16 Command Understanding / Kommando Verstehen (Q125)

Listing 6.7: Das SQL (PostgreSQL) script select_03.sql (src)

```sql
SELECT name, address FROM passenger;
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.17 Constraints / Einschränkungen (Q126)

EN What is a `REFERENCES` / `FOREIGN KEY` constraint? What is it used for?

DE Was ist eine `REFERENCES`- / `FOREIGN KEY`-Einschränkung? Wofür wird sie verwendet?

6.1.18 Command Understanding / Kommando Verstehen (Q127)

Listing 6.8: Das SQL (PostgreSQL) script update_02.sql (src)

```sql
UPDATE student
SET academic_title = 'MSc'
WHERE (name = 'Bibbo Bobbson') AND (academic_title = 'BSc');
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.1.19 Constraints / Einschränkungen (Q128)

EN What is a `NOT NULL` constraint? What is it used for?

DE Was ist eine `NOT NULL`-Einschränkung? Wofür wird sie verwendet?

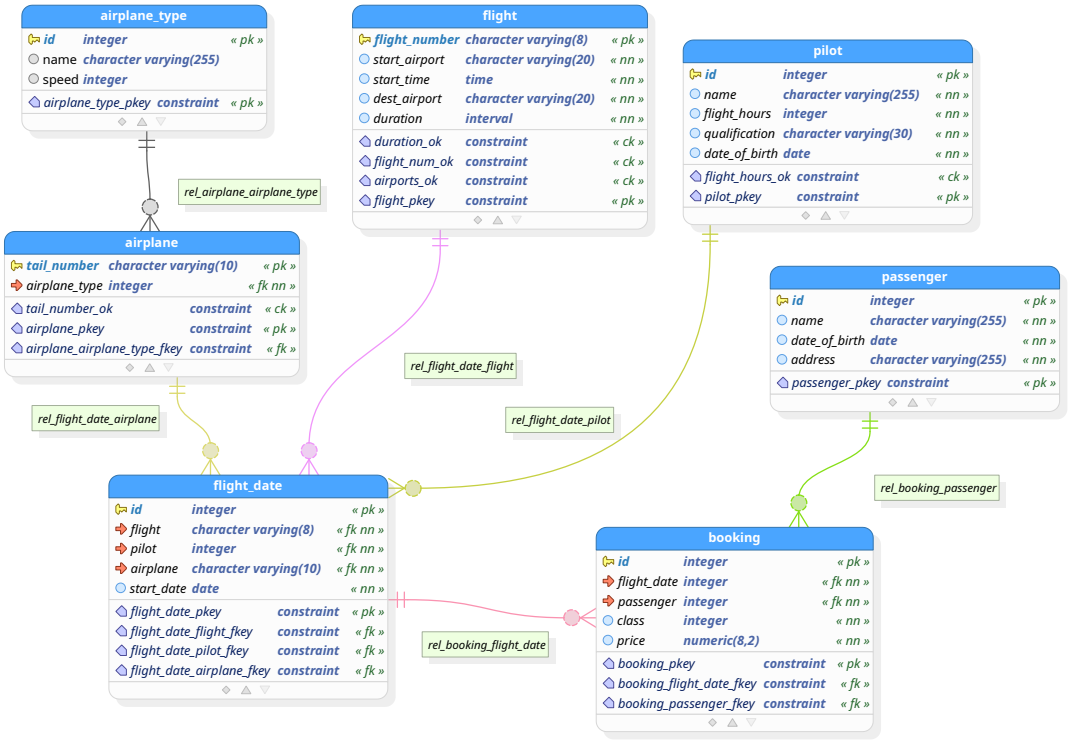


Figure 6.1: The logical model of the airline example. Das logische Modell des Airline-Beispiels.

6.1.20 Command Understanding / Kommando Verstehen (Q129)

Listing 6.9: Das SQL (PostgreSQL) script update_01.sql (src)

```
1 UPDATE student
2 SET name = 'Bibbo Bobbson'
3 WHERE name = 'Bibbo Bobson';
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2 Airline Example / Fluglienen-Beispiel

EN Now we look at a larger consistent example. The logical model of an airline DB is given in Figure 6.1.

DE Jetzt schauen wir uns ein größeres, konsistentes Beispiel an. Das logische Modell einer Airline-Datenbank ist in Figure 6.1 gegeben.

6.2.1 Database Creation / Datenbank Erstellen (Q130)

Listing 6.10: Das SQL (PostgreSQL) script airline_database.sql (src)

```sql
/** Create the airline example database. */
CREATE DATABASE airline;
```

EN What does the above SQL (PostgreSQL) command do?

DE Was macht das SQL (PostgreSQL) Kommando oben?

6.2.2 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q131)

Listing 6.11: Das SQL (PostgreSQL) script airline_table_airplane_type.sql. (src)

```sql
-- The airplane_type table of the airline example.
CREATE TABLE airplane_type (
    id        INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    name      VARCHAR(255),
    speed     INT);
```

Listing 6.12: Der Output eines SQL (PostgreSQL) Kommandos.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_airplane_type.sql
 id |          name          | speed
----+------------------------+-------
  1 | Junkers F 13           |   170
  2 | Douglas DC-3           |   370
  3 | Messerschmitt Me 262 |   870
  4 | Airbus A320            |   962
(4 rows)

INSERT 0 4
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script airline_table_airplane_type.sql in Listing 6.11 is executed and produces a table. Construct the command for inserting data into that table as shown in Listing 6.12.

DE Das SQL (PostgreSQL)-Skript airline_table_airplane_type.sql in Listing 6.11 wird ausgeführt und erstellt eine Tabelle. Bauen Sie ein SQL-Kommando, dass die selben Daten in diese Tabelle einfügt wie in Listing 6.12 angegeben.

6.2.3 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q132)

Listing 6.13: Das SQL (PostgreSQL) script airline_insert_airplane.sql. (stored in file airline_insert_airplane.sql; output in Listing 6.14)

```
-- Insert into the airplane table.
INSERT INTO airplane (tail_number, airplane_type) VALUES
    ('B-1000', 4),  ('B-1001', 3),  ('B-1002', 2),  ('B-1003', 1),
    ('D-2888', 3),  ('D-2889', 2),  ('D-2890', 2),  ('D-8891', 3),
    ('F-4444', 1),  ('F-5555', 4),  ('F-6666', 1),  ('F-7777', 4)
RETURNING *;
```

Listing 6.14: The standard output stream (stdout) of the program airline_insert_airplane.sql given in Listing 6.13.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_airplane.sql
 tail_number | airplane_type
-------------+---------------
 B-1000      |             4
 B-1001      |             3
 B-1002      |             2
 B-1003      |             1
 D-2888      |             3
 D-2889      |             2
 D-2890      |             2
 D-8891      |             3
 F-4444      |             1
 F-5555      |             4
 F-6666      |             1
 F-7777      |             4
(12 rows)

INSERT 0 12
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script airline_insert_airplane.sql in Listing 6.13 inserts data into a table. Construct the command for creating the table in a reasonable fashion.

- Assume that the field `airplane_type` is a foreign key to the table `airplane_type` created by script airline_table_airplane_type.sql in Listing 6.11 in the previous question.
- Assume that none of the columns are optional.
- Assume that tail numbers always start with one or multiple uppercase letters, followed by a dash, follwed by 2 to 8 digits. Create a `CHECK` constraint enforcing this?

DE Das SQL (PostgreSQL)-Skript airline_insert_airplane.sql in Listing 6.13 fügt Daten in eine Tabelle ein. Geben Sie ein Kommando an, dass die Tabelle auf vernünftige Art erstellen würde.

- Nehmen Sie an, dass das Feld `airplane_type` ein Fremdschlüssel auf die Tabelle `airplane_type` ist, die vom Skript airline_table_airplane_type.sql in Listing 6.11 in der vorigen Aufgabe erstellt wurde.
- Nehmen Sie an, das keine der Spalten optional ist.
- Nehmen Sie an, dass die `taile_number` immer mit einem oder mehreren Großbuchstaben beginnt, gefolgt von einem Bindestrich, gefolgt von 2 bis 8 Ziffern. Bauen Sie eine `CHECK`-Einschränkung die das erzwingt.

6.2.4 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q133)

Listing 6.15: Das SQL (PostgreSQL) script airline_table_pilot.sql. (src)

```
-- The pilot table of the airline example.
CREATE TABLE pilot (
    id             INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    name           VARCHAR(255)  NOT NULL,
    flight_hours   INT           NOT NULL,
    qualification  VARCHAR(30)   NOT NULL,
    date_of_birth  DATE          NOT NULL,
    CONSTRAINT flight_hours_ok  CHECK (flight_hours > 100)
);
```

Listing 6.16: Der Output eines SQL (PostgreSQL) Kommandos.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_pilot.sql
 id |     name      | flight_hours | qualification | date_of_birth
----+---------------+--------------+---------------+---------------
  1 | Fred Bibbo    |          400 | Captain       | 2000-01-12
  2 | Jane Bebbo    |         2200 | Chief Pilot   | 1995-03-21
  3 | Eugene Bobbo  |         1760 | Chief Pilot   | 1992-11-06
  4 | Mary Bebba    |         3310 | Chief Pilot   | 1987-08-15
  5 | Jake Babba    |          125 | Captain       | 1997-02-28
  6 | Aurelia Bibbi |          833 | Captain       | 1983-09-11
  7 | Luke Babbo    |          313 | Captain       | 1994-04-25
(7 rows)

INSERT 0 7
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script airline_table_pilot.sql in Listing 6.15 is executed and produces a table.

- Construct the command for inserting data into that table as shown in Listing 6.16.
- What does the CONSTRAINT at the bottom of the script airline_table_pilot.sql do?

DE Das SQL (PostgreSQL)-Skript airline_table_pilot.sql in Listing 6.15 wird ausgeführt und erstellt eine Tabelle.

- Bauen Sie ein SQL-Kommando, dass die selben Daten in diese Tabelle einfügt wie in Listing 6.16 angegeben.
- Was macht das CONSTRAINT am Fuß des Skriptes airline_table_pilot.sql do?

6.2.5 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q134)

Listing 6.17: Das SQL (PostgreSQL) script airline_insert_passenger.sql. (stored in file airline_insert_passenger.sql; output in Listing 6.18)

```
-- Insert into the passenger table.
INSERT INTO passenger
    (name,           date_of_birth, address)
VALUES
    ('Frank Pippo',   '1968-06-07', 'Osnabrück, Germany'),
    ('Jessica Peppo', '2008-03-05', 'London, UK'),
    ('Malik Poppo',   '1967-01-31', 'Chemnitz, Germany'),
    ('Lars Peppa',    '1983-03-09', 'Hefei, China'),
    ('Dieter Pappa',  '1991-11-15', 'Dresden, Germany'),
    ('Anne Pippi',    '1972-10-22', 'Prague, Czech Republic'),
    ('Liz Pappo',     '1993-06-12', 'Vienna, Austria'),
    ('Nicole Peppe',  '1956-08-22', 'Munich, Germany'),
    ('Daniel Pippo',  '2002-07-03', 'Ximaen, Hefei')
RETURNING *;
```

Listing 6.18: The stdout of the program airline_insert_passenger.sql given in Listing 6.17.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_passenger.sql
 id |     name      | date_of_birth |        address
----+---------------+---------------+------------------------
  1 | Frank Pippo   | 1968-06-07    | Osnabrück, Germany
  2 | Jessica Peppo | 2008-03-05    | London, UK
  3 | Malik Poppo   | 1967-01-31    | Chemnitz, Germany
  4 | Lars Peppa    | 1983-03-09    | Hefei, China
  5 | Dieter Pappa  | 1991-11-15    | Dresden, Germany
  6 | Anne Pippi    | 1972-10-22    | Prague, Czech Republic
  7 | Liz Pappo     | 1993-06-12    | Vienna, Austria
  8 | Nicole Peppe  | 1956-08-22    | Munich, Germany
  9 | Daniel Pippo  | 2002-07-03    | Ximaen, Hefei
(9 rows)

INSERT 0 9
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script in Listing 6.17 inserts data into a table.

- Construct the command for creating the table in a reasonable fashion.
- Assume none of the columns are optional.

DE Das SQL (PostgreSQL)-Skript in Listing 6.17 fügt Daten in eine Tabelle ein.

- Geben Sie ein Kommando an, dass die Tabelle auf vernünftige Art erstellen würde.
- Nehmen Sie an, das keine der Spalten optional ist.

6.2.6 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q135)

Listing 6.19: Das SQL (PostgreSQL) script airline_table_flight.sql. (src)

```
-- Create the flight table of the airline example.
CREATE TABLE flight (
    flight_number    VARCHAR(8)    NOT NULL PRIMARY KEY,
    start_airport    VARCHAR(20)   NOT NULL,
    start_time       TIME          NOT NULL,
    dest_airport     VARCHAR(20)   NOT NULL,
    duration         INTERVAL      NOT NULL,
    CONSTRAINT duration_ok    CHECK (duration      > '10 minutes'),
    CONSTRAINT flight_num_ok CHECK (flight_number ~ '^[A-Z]+\d{1,5}$'),
    CONSTRAINT airports_ok    CHECK (dest_airport != start_airport)
);
```

Listing 6.20: Der Output eines SQL (PostgreSQL) Kommandos.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_flight.sql
 flight_number | start_airport | start_time | dest_airport | duration
---------------+---------------+------------+--------------+----------
 XA1843        | PEK           | 13:30:00   | HFE          | 01:55:00
 XA1844        | HFE           | 16:15:00   | PEK          | 02:00:00
 XA1813        | PEK           | 07:55:00   | HFE          | 01:55:00
 XA1814        | HFE           | 10:45:00   | PEK          | 02:15:00
 XU489         | PEK           | 03:20:00   | BER          | 09:10:00
 XU490         | BER           | 12:55:00   | PEK          | 10:20:00
 XU6931        | HFE           | 09:10:00   | XMN          | 01:40:00
 XU6932        | XMN           | 12:50:00   | HFE          | 01:30:00
(8 rows)

INSERT 0 8
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script airline_table_flight.sql in Listing 6.19 is executed and produces a table.

- Construct the command for inserting data into that table as shown in Listing 6.20.
- What do the CONSTRAINTs at the bottom of the script airline_table_flight.sql do?

DE Das SQL (PostgreSQL)-Skript airline_table_flight.sql in Listing 6.19 wird ausgeführt und erstellt eine Tabelle.

- Bauen Sie ein SQL-Kommando, dass die selben Daten in diese Tabelle einfügt wie in Listing 6.20 angegeben.
- Was machen die CONSTRAINTs am Fuß des Skriptes airline_table_flight.sql do?

6.2.7 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q136)

Listing 6.21: Das SQL (PostgreSQL) script airline_insert_flight_date.sql. (stored in file airline_insert_flight_date.sql; output in Listing 6.22)

```
-- Insert into the flight date table.
INSERT INTO flight_date (flight, pilot, airplane, start_date)
VALUES ('XA1843',  2, 'B-1000', '2025-12-02'),
       ('XA1844',  3, 'D-2889', '2025-12-02'),
       ('XA1813',  1, 'F-6666', '2025-12-02'),
       ('XA1814',  4, 'D-8891', '2025-12-03'),
       ('XU489',   7, 'B-1002', '2025-12-03'),
       ('XU490',   5, 'F-7777', '2025-12-04'),
       ('XA1843',  6, 'B-1001', '2025-12-05'),
       ('XA1844',  1, 'D-2888', '2025-12-05'),
       ('XA1813',  3, 'F-7777', '2025-12-06'),
       ('XA1814',  2, 'D-2890', '2025-12-06'),
       ('XU489',   4, 'B-1003', '2025-12-07'),
       ('XU490',   7, 'F-4444', '2025-12-07'),
       ('XU6931',  4, 'B-1003', '2025-12-09'),
       ('XU6931',  3, 'D-2890', '2025-12-11'),
       ('XU6932',  5, 'F-6666', '2025-12-11')
RETURNING *;
```

Listing 6.22: The stdout of the program airline_insert_flight_date.sql given in Listing 6.21.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_flight_date.sql
 id | flight | pilot | airplane | start_date
----+--------+-------+----------+------------
  1 | XA1843 |     2 | B-1000   | 2025-12-02
  2 | XA1844 |     3 | D-2889   | 2025-12-02
  3 | XA1813 |     1 | F-6666   | 2025-12-02
  4 | XA1814 |     4 | D-8891   | 2025-12-03
  5 | XU489  |     7 | B-1002   | 2025-12-03
  6 | XU490  |     5 | F-7777   | 2025-12-04
  7 | XA1843 |     6 | B-1001   | 2025-12-05
  8 | XA1844 |     1 | D-2888   | 2025-12-05
  9 | XA1813 |     3 | F-7777   | 2025-12-06
 10 | XA1814 |     2 | D-2890   | 2025-12-06
 11 | XU489  |     4 | B-1003   | 2025-12-07
 12 | XU490  |     7 | F-4444   | 2025-12-07
 13 | XU6931 |     4 | B-1003   | 2025-12-09
 14 | XU6931 |     3 | D-2890   | 2025-12-11
 15 | XU6932 |     5 | F-6666   | 2025-12-11
(15 rows)

INSERT 0 15
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script in Listing 6.21 inserts data into a table.

- Construct the command for creating the table in a reasonable fashion.
- Assume none of the columns are optional.
- `flight` be a foreign key referencing table `flight` created by airline_table_flight.sql in Listing 6.19 in one of the previous tasks.
- `pilot` be a foreign key referencing table `pilot` created by airline_table_pilot.sql in Listing 6.15 in one of the previous tasks.
- `airplane` be a foreign key referencing table `airplane` created in one of the previous tasks and filled with data by script Listing 6.13 in airline_insert_airplane.sql.

6.2 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q136) (continued/fortgesetzt)

DE Das SQL (PostgreSQL)-Skript in Listing 6.21 fügt Daten in eine Tabelle ein.

- Geben Sie ein Kommando an, dass die Tabelle auf vernünftige Art erstellen würde.
- Nehmen Sie an, das keine der Spalten optional ist.
- `flight` soll eine Fremdschlüssel-Referenz auf Tabelle `flight` erstellt von `airline_table_flight.sql` in Listing 6.19 in einer der vorigen Aufgaben sein
- `pilot` soll eine Fremdschlüssel-Referenz auf Tabelle `pilot` erstellt von `airline_table_pilot.sql` in Listing 6.15 in einer der vorigen Aufgaben sein.
- `airplane` soll eine Fremdschlüssel-Referenz auf Tabelle `airplane` erstellt in einer der vorigen Aufgaben und gefüllt mit Daten via Skript Listing 6.13 in `airline_insert_airplane.sql` sein.

6.2.8 Table Creation and Data Insertion / Tabelle Erstellen und Befüllen (Q137)

Listing 6.23: Das SQL (PostgreSQL) script airline_table_booking.sql. (src)

```
-- Create the booking table of the airline example.
CREATE TABLE booking (
    id          INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    flight_date INT           NOT NULL REFERENCES flight_date (id),
    passenger   INT           NOT NULL REFERENCES passenger   (id),
    class       INT           NOT NULL,
    price       DECIMAL(8, 2) NOT NULL
);
```

Listing 6.24: Der Output eines SQL (PostgreSQL) Kommandos.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_insert_booking.sql
 id | flight_date | passenger | class |  price
----+-------------+-----------+-------+---------
  1 |           1 |         2 |     2 |  300.00
  2 |           2 |         5 |     1 |  600.00
  3 |           3 |         4 |     1 | 1233.00
  4 |           4 |         7 |     1 | 2300.00
  5 |           5 |         1 |     2 | 1734.00
  6 |           6 |         2 |     2 |  925.00
  7 |           7 |         3 |     2 |  212.00
  8 |           8 |         4 |     1 |  566.00
  9 |           9 |         6 |     1 | 2451.00
 10 |          10 |         4 |     1 |  933.00
 11 |          11 |         5 |     2 |  600.00
 12 |          12 |         1 |     2 |  354.00
 13 |           1 |         3 |     2 |   56.00
 14 |           8 |         3 |     1 | 3450.00
 15 |           6 |         6 |     1 | 8943.00
 16 |          13 |         7 |     1 | 1332.00
 17 |          14 |         8 |     2 |  500.00
 18 |          15 |         9 |     2 | 1002.00
(18 rows)

INSERT 0 18
# psql 16.12 succeeded with exit code 0.
```

EN The SQL (PostgreSQL) script airline_table_booking.sql in Listing 6.23 is executed and produces a table.

- Construct the command for inserting data into that table as shown in Listing 6.24.
- What do the CONSTRAINTs at the bottom of the script airline_table_flight.sql do?

DE Das SQL (PostgreSQL)-Skript airline_table_booking.sql in Listing 6.23 wird ausgeführt und erstellt eine Tabelle.

- Bauen Sie ein SQL-Kommando, dass die selben Daten in diese Tabelle einfügt wie in Listing 6.24 angegeben.
- Was machen die CONSTRAINTs am Fuß des Skriptes airline_table_flight.sql do?

6.2.9 Queries / Anfragen (Q138)

EN Write an SQL (PostgreSQL) qery that yields the list of all passengers (name, address) that booked a flight for more than 1000 RMB..

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste aller Passagierte (name, address) zurückliefert, die einen Flug für mehr als 1000 RMB gebucht haben.

6.2.10 Command Understanding / Kommando Verstehen (Q139)

Listing 6.25: Das SQL (PostgreSQL) script airline_select_01.sql (stored in file airline_select_01.sql; output in Listing 6.26)

```
1 SELECT id, name, flight_hours
2     FROM pilot
3     WHERE qualification = 'Captain';
```

Listing 6.26: The stdout of the program airline_select_01.sql given in Listing 6.25.

```
1  $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
     ↪ airline_select_01.sql
2   id |      name       | flight_hours
3  ----+-----------------+--------------
4    1 | Fred Bibbo      |          400
5    5 | Jake Babba      |          125
6    6 | Aurelia Bibbi   |          833
7    7 | Luke Babbo      |          313
8  (4 rows)
9
10 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.11 Command Understanding / Kommando Verstehen (Q140)

Listing 6.27: Das SQL (PostgreSQL) script airline_select_02.sql (stored in file airline_select_02.sql; output in Listing 6.28)

```
1 SELECT DISTINCT start_airport, dest_airport, duration
2     FROM flight;
```

Listing 6.28: The stdout of the program airline_select_02.sql given in Listing 6.27.

```
1  $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
       ↪ airline_select_02.sql
2   start_airport | dest_airport | duration
3  ---------------+--------------+----------
4   PEK           | HFE          | 01:55:00
5   PEK           | BER          | 09:10:00
6   BER           | PEK          | 10:20:00
7   HFE           | PEK          | 02:15:00
8   HFE           | PEK          | 02:00:00
9   XMN           | HFE          | 01:30:00
10  HFE           | XMN          | 01:40:00
11 (7 rows)
12
13 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.12 Queries / Anfragen (Q141)

EN Write an SQL (PostgreSQL) qery that yields the list of airports from which we can fly to Hefei.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Flughäfen, von denen aus wir nach Hefei fliegen können, liefert.

6.2.13 Command Understanding / Kommando Verstehen (Q142)

Listing 6.29: Das SQL (PostgreSQL) script airline_select_03.sql (stored in file airline_select_03.sql; output in Listing 6.30)

```
1 SELECT name, address FROM passenger
2     WHERE address ILIKE '%Hefei%';
```

Listing 6.30: The stdout of the program airline_select_03.sql given in Listing 6.29.

```
1 $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_select_03.sql
2      name      |     address
3 ---------------+---------------
4  Lars Peppa    | Hefei, China
5  Daniel Pippo  | Ximaen, Hefei
6 (2 rows)
7
8 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.14 Queries / Anfragen (Q143)

EN Write an SQL (PostgreSQL) qery that yields the list of all flight routes (start and destination airport, duration).

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste aller Flugrouten (Start- und Zielflughafen, Flugdauer) zurückliefert.

6.2.15 Command Understanding / Kommando Verstehen (Q144)

Listing 6.31: Das SQL (PostgreSQL) script airline_select_04.sql (stored in file airline_select_04.sql; output in Listing 6.32)

```
1 SELECT name, date_of_birth FROM pilot
2     WHERE name LIKE '% Bobbo';
```

Listing 6.32: The stdout of the program airline_select_04.sql given in Listing 6.31.

```
1 $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_select_04.sql
2       name      | date_of_birth
3 --------------+---------------
4  Eugene Bobbo | 1992-11-06
5 (1 row)
6
7 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.16 Queries / Anfragen (Q145)

EN Write an SQL (PostgreSQL) qery that yields the list of the names, flight hours, and IDs of all pilots with qualification *Captain*.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Namen, Flugstunden und IDs aller Piloten mit Qualifikation *Captain* zurückliefert.

6.2.17 Command Understanding / Kommando Verstehen (Q146)

Listing 6.33: Das SQL (PostgreSQL) script airline_select_05.sql (stored in file airline_select_05.sql; output in Listing 6.34)

```
1 SELECT start_time FROM flight
2 WHERE dest_airport IN ('HFE', 'BER');
```

Listing 6.34: The stdout of the program airline_select_05.sql given in Listing 6.33.

```
1  $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
     ↪ airline_select_05.sql
2   start_time
3  ------------
4   13:30:00
5   07:55:00
6   03:20:00
7   12:50:00
8  (4 rows)
9
10 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.18 Queries / Anfragen (Q147)

EN Write an SQL (PostgreSQL) qery that yields the list of all flight dates that go to Berlin (BER).

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste aller Flugdaten (flight dates) die nach Berlin (BER) gehen, zurückliefert.

6.2.19 Command Understanding / Kommando Verstehen (Q148)

Listing 6.35: Das SQL (PostgreSQL) script airline_select_06.sql (stored in file airline_select_06.sql; output in Listing 6.36)

```
1 SELECT name FROM pilot
2     WHERE (qualification != 'Chief Pilot') AND (flight_hours > 500);
```

Listing 6.36: The stdout of the program airline_select_06.sql given in Listing 6.35.

```
1 $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_select_06.sql
2      name
3 ---------------
4  Aurelia Bibbi
5 (1 row)
6
7 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.20 Command Understanding / Kommando Verstehen (Q149)

Listing 6.37: Das SQL (PostgreSQL) script airline_select_07.sql (stored in file airline_select_07.sql; output in Listing 6.38)

```
1 SELECT id, qualification, flight_hours FROM pilot
2     WHERE qualification IN ('Captain', 'Chief Pilot')
3                 AND (flight_hours > 600);
```

Listing 6.38: The stdout of the program airline_select_07.sql given in Listing 6.37.

```
1 $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_select_07.sql
2  id | qualification | flight_hours
3 ----+---------------+--------------
4   2 | Chief Pilot   |         2200
5   3 | Chief Pilot   |         1760
6   4 | Chief Pilot   |         3310
7   6 | Captain       |          833
8 (4 rows)
9
10 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.21 Queries / Anfragen (Q150)

EN Write an SQL (PostgreSQL) qery that yields the list of the pilots (name) who have over 300 flight hours but are not yet *Chief Pilot*.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Piloten (name) zurückliefert, die über 300 Flugstunden haben, aber noch nicht *Chief Pilot* sind.

6.2.22 Command Understanding / Kommando Verstehen (Q151)

Listing 6.39: Das SQL (PostgreSQL) script airline_join_01.sql (stored in file airline_join_01.sql; output in Listing 6.40)

```
SELECT tail_number, airplane_type.name as type, airplane_type.speed
    FROM airplane
    INNER JOIN airplane_type ON airplane_type.id = airplane.airplane_type
    WHERE airplane_type.speed > 400;
```

Listing 6.40: The stdout of the program airline_join_01.sql given in Listing 6.39.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_join_01.sql
 tail_number |          type          | speed
-------------+------------------------+-------
 B-1000      | Airbus A320            |   962
 B-1001      | Messerschmitt Me 262   |   870
 D-2888      | Messerschmitt Me 262   |   870
 D-8891      | Messerschmitt Me 262   |   870
 F-5555      | Airbus A320            |   962
 F-7777      | Airbus A320            |   962
(6 rows)

# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.23 Queries / Anfragen (Q152)

EN Write an SQL (PostgreSQL) qery that yields the list of the names and addresses of all passengers who live in Hefei.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Namen und Adressen aller Passagiere, die in Hefei leben, zurückliefert.

6.2.24 Command Understanding / Kommando Verstehen (Q153)

Listing 6.41: Das SQL (PostgreSQL) script airline_join_02.sql (stored in file airline_join_02.sql; output in Listing 6.42)

```
SELECT passenger.name AS name, flight.start_airport, flight.dest_airport,
       flight_date.start_date
    FROM flight_date
    INNER JOIN flight    ON flight.flight_number = flight_date.flight
    INNER JOIN booking   ON booking.flight_date  = flight_date.id
    INNER JOIN passenger ON passenger.id         = booking.passenger
    WHERE passenger.name ILIKE '%Pepp%';
```

Listing 6.42: The stdout of the program airline_join_02.sql given in Listing 6.41.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
   ↪ airline_join_02.sql
     name      | start_airport | dest_airport | start_date
---------------+---------------+--------------+------------
 Jessica Peppo | PEK           | HFE          | 2025-12-02
 Lars Peppa    | PEK           | HFE          | 2025-12-02
 Jessica Peppo | BER           | PEK          | 2025-12-04
 Lars Peppa    | HFE           | PEK          | 2025-12-05
 Lars Peppa    | HFE           | PEK          | 2025-12-06
 Nicole Peppe  | HFE           | XMN          | 2025-12-11
(6 rows)

# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.25 Command Understanding / Kommando Verstehen (Q154)

Listing 6.43: Das SQL (PostgreSQL) script airline_join_03.sql (stored in file airline_join_03.sql; output in Listing 6.44)

```
SELECT DISTINCT pilot.id AS id, pilot.name AS name
    FROM flight_date
    INNER JOIN flight    ON flight.flight_number = flight_date.flight
    INNER JOIN pilot     ON pilot.id             = flight_date.pilot
    WHERE flight.dest_airport = 'BER';
```

Listing 6.44: The stdout of the program airline_join_03.sql given in Listing 6.43.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_join_03.sql
 id |    name
----+------------
  4 | Mary Bebba
  7 | Luke Babbo
(2 rows)

# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.26 Command Understanding / Kommando Verstehen (Q155)

Listing 6.45: Das SQL (PostgreSQL) script airline_join_04.sql (stored in file airline_join_04.sql; output in Listing 6.46)

```
SELECT pilot.id AS id, pilot.name AS name,
       MAX(airplane_type.speed) AS max_speed
   FROM flight_date
   INNER JOIN pilot      ON pilot.id              = flight_date.pilot
   INNER JOIN airplane   ON flight_date.airplane = airplane.tail_number
   INNER JOIN airplane_type ON airplane_type.id = airplane.airplane_type
   GROUP BY pilot.id, pilot.name;
```

Listing 6.46: The stdout of the program airline_join_04.sql given in Listing 6.45.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_join_04.sql
 id |     name      | max_speed
----+---------------+-----------
  3 | Eugene Bobbo  |       962
  5 | Jake Babba    |       962
  4 | Mary Bebba    |       870
  6 | Aurelia Bibbi |       870
  2 | Jane Bebbo    |       962
  7 | Luke Babbo    |       370
  1 | Fred Bibbo    |       870
(7 rows)

# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.27 Command Understanding / Kommando Verstehen (Q156)

Listing 6.47: Das SQL (PostgreSQL) script airline_update_01.sql (stored in file airline_update_01.sql; output in Listing 6.48)

```
UPDATE flight
SET duration = duration + '1 hour'
WHERE flight_number = 'XU489';
```

Listing 6.48: The stdout of the program airline_update_01.sql given in Listing 6.47.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_update_01.sql
UPDATE 1
# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.28 Queries / Anfragen (Q157)

EN Write an SQL (PostgreSQL) qery that yields the list of the names and addresses of all passengers.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Namen und Adressen aller Passagiere zurückliefert.

6.2.29 Command Understanding / Kommando Verstehen (Q158)

Listing 6.49: Das SQL (PostgreSQL) script airline_update_02.sql (stored in file airline_update_02.sql; output in Listing 6.50)

```
UPDATE booking
    SET price = price * 2, class = 1
    WHERE passenger IN (
        SELECT id FROM passenger
            WHERE name = 'Jessica Peppo')
    AND (class = 2) AND (flight_date = 1)
RETURNING *;
```

Listing 6.50: The stdout of the program airline_update_02.sql given in Listing 6.49.

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_update_02.sql
 id | flight_date | passenger | class | price
----+-------------+-----------+-------+--------
  1 |           1 |         2 |     1 | 600.00
(1 row)

UPDATE 1
# psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.30 Queries / Anfragen (Q159)

EN Write an SQL (PostgreSQL) qery that yields the list of the names and dates of birth of all pilots whose names contain either *Babba* or *Bebbo*.

DE Schreiben Sie eine SQL (PostgreSQL)-Anfrage, die die Liste der Namen und Geburtsdaten aller Piloten, deren Namen entweder *Babba* oder *Bebbo* beinhalten, zurückliefert.

6.2.31 Command Understanding / Kommando Verstehen (Q160)

Listing 6.51: Das SQL (PostgreSQL) script airline_delete_01.sql (stored in file airline_delete_01.sql; output in Listing 6.52)

```
1 DELETE FROM booking WHERE id = 11;
```

Listing 6.52: The stdout of the program airline_delete_01.sql given in Listing 6.51.

```
1 $ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf
    ↪ airline_delete_01.sql
2 DELETE 1
3 # psql 16.12 succeeded with exit code 0.
```

EN What does the above SQL (PostgreSQL) command do? Which tables and columns must exist for this command to work?

DE Was macht das SQL (PostgreSQL) Kommando oben? Welche Tabellen und Spalten müssen existieren, damit dieses Kommando funktioniert?

6.2.32 Joins (Q161)

Listing 6.53: Alle Daten "zusammengezogen".

```
$ psql "postgres://postgres:XXX@localhost/airline" -v ON_ERROR_STOP=1 -ebf airline_join_05.sql
 flight_number |    date    |   time   | start | duration | dest | tail_number |          type          | speed |     pilot     | hrs |    qual     | pilot_dob  |   passenger   | passenger_dob |         address          | class |  price
---------------+------------+----------+-------+----------+------+-------------+------------------------+-------+---------------+-----+-------------+------------+---------------+---------------+--------------------------+-------+---------
 XA1813        | 2025-12-02 | 07:55:00 | PEK   | 01:55:00 | HFE  | F-6666      | Junkers F 13           |   170 | Fred Bibbo    |  400 | Captain     | 2000-01-12 | Lars Peppa    | 1983-03-09    | Hefei, China             |     1 | 1233.00
 XA1843        | 2025-12-02 | 13:30:00 | PEK   | 01:55:00 | HFE  | B-1000      | Airbus A320            |   962 | Jane Bebbo    | 2200 | Chief Pilot | 1995-03-21 | Jessica Peppo | 2008-03-05    | London, UK               |     1 |  600.00
 XA1843        | 2025-12-02 | 13:30:00 | PEK   | 01:55:00 | HFE  | B-1000      | Airbus A320            |   962 | Jane Bebbo    | 2200 | Chief Pilot | 1995-03-21 | Malik Poppo   | 1967-01-31    | Chemnitz, Germany        |     2 |   56.00
 XA1844        | 2025-12-02 | 16:15:00 | HFE   | 02:00:00 | PEK  | D-2889      | Douglas DC-3           |   370 | Eugene Bobbo  | 1760 | Chief Pilot | 1992-11-06 | Dieter Pappa  | 1991-11-15    | Dresden, Germany         |     1 |  600.00
 XU489         | 2025-12-03 | 03:20:00 | PEK   | 10:10:00 | BER  | B-1002      | Douglas DC-3           |   370 | Luke Babbo    |  313 | Captain     | 1994-04-25 | Frank Pippo   | 1968-06-07    | Osnabrück, Germany       |     2 | 1734.00
 XA1814        | 2025-12-03 | 10:45:00 | HFE   | 02:15:00 | PEK  | D-8891      | Messerschmitt Me 262   |   870 | Mary Bebba    | 3310 | Chief Pilot | 1987-08-15 | Liz Pappo     | 1993-06-12    | Vienna, Austria          |     1 | 2300.00
 XU490         | 2025-12-04 | 12:55:00 | BER   | 10:20:00 | PEK  | F-7777      | Airbus A320            |   962 | Jake Babba    |  125 | Captain     | 1997-02-28 | Anne Pippi    | 1972-10-22    | Prague, Czech Republic   |     1 | 8943.00
 XU490         | 2025-12-04 | 12:55:00 | BER   | 10:20:00 | PEK  | F-7777      | Airbus A320            |   962 | Jake Babba    |  125 | Captain     | 1997-02-28 | Jessica Peppo | 2008-03-05    | London, UK               |     2 |  925.00
 XA1843        | 2025-12-05 | 13:30:00 | PEK   | 01:55:00 | HFE  | B-1001      | Messerschmitt Me 262   |   870 | Aurelia Bibbi |  833 | Captain     | 1983-09-11 | Malik Poppo   | 1967-01-31    | Chemnitz, Germany        |     2 |  212.00
 XA1844        | 2025-12-05 | 16:15:00 | HFE   | 02:00:00 | PEK  | D-2888      | Messerschmitt Me 262   |   870 | Fred Bibbo    |  400 | Captain     | 2000-01-12 | Lars Peppa    | 1983-03-09    | Hefei, China             |     1 |  566.00
 XA1844        | 2025-12-05 | 16:15:00 | HFE   | 02:00:00 | PEK  | D-2888      | Messerschmitt Me 262   |   870 | Fred Bibbo    |  400 | Captain     | 2000-01-12 | Malik Poppo   | 1967-01-31    | Chemnitz, Germany        |     1 | 3450.00
 XA1813        | 2025-12-06 | 07:55:00 | PEK   | 01:55:00 | HFE  | F-7777      | Airbus A320            |   962 | Eugene Bobbo  | 1760 | Chief Pilot | 1992-11-06 | Anne Pippi    | 1972-10-22    | Prague, Czech Republic   |     1 | 2451.00
 XA1814        | 2025-12-06 | 10:45:00 | HFE   | 02:15:00 | PEK  | D-2890      | Douglas DC-3           |   370 | Jane Bebbo    | 2200 | Chief Pilot | 1995-03-21 | Lars Peppa    | 1983-03-09    | Hefei, China             |     1 |  933.00
 XU490         | 2025-12-07 | 12:55:00 | BER   | 10:20:00 | PEK  | F-4444      | Junkers F 13           |   170 | Luke Babbo    |  313 | Captain     | 1994-04-25 | Frank Pippo   | 1968-06-07    | Osnabrück, Germany       |     2 |  354.00
 XU6931        | 2025-12-09 | 09:10:00 | HFE   | 01:40:00 | XMN  | B-1003      | Junkers F 13           |   170 | Mary Bebba    | 3310 | Chief Pilot | 1987-08-15 | Liz Pappo     | 1993-06-12    | Vienna, Austria          |     1 | 1332.00
 XU6931        | 2025-12-11 | 09:10:00 | HFE   | 01:40:00 | XMN  | D-2890      | Douglas DC-3           |   370 | Eugene Bobbo  | 1760 | Chief Pilot | 1992-11-06 | Nicole Pappe  | 1956-08-22    | Munich, Germany          |     2 |  500.00
 XU6932        | 2025-12-11 | 12:50:00 | XMN   | 01:30:00 | HFE  | F-6666      | Junkers F 13           |   170 | Jake Babba    |  125 | Captain     | 1997-02-28 | Daniel Pippo  | 2002-07-03    | Ximen, Hefei             |     2 | 1002.00
(17 rows)

# psql 16.12 succeeded with exit code 0.
```

EN Create an SQL (PostgreSQL) query that merges all the data from all the table, i.e., shows us for each booking all the data ranging from the passenger's and pilot's names to the airplane type and speed as well as the start and destinaton airports, etc. In other words, create a command producing the same humonguous output as shown in Listing 6.53.

DE Erstellen Sie eine SQL (PostgreSQL)-Anfrage, die alle Daten von allen Tabellen zusammenzieht, die also für jede Buchung die Daten angefangen von Passagier- und Pilotenname über den Flugzeugtype und die Flugzeuggeschwindigkiet hin zum Start- und End-Flughafen usw. zeigt. In anderen Worten, bauen Sie eine Anfrage zusammen, die die selben gewaltigen Ausgabedaten wie in Listing 6.53 gezeigt produziert.

6.2.33 Database Deletion / Datenbank Löschen (Q162)

Listing 6.54: Das SQL (PostgreSQL) script cleanup.sql (src)

```sql
/* Cleanup after the example: Delete the airline database. */

DROP DATABASE IF EXISTS airline;
```

EN What does the above SQL (PostgreSQL) command do?

DE Was macht das SQL (PostgreSQL) Kommando oben?

Chapter 7

Normalization, Anomalies, and Keys / Normalisation, Anomalien und Schlüssel

7.1 First Normal Form / Erste Normalform (Q163)

EN Explain in plain English what conditions a table must fulfill to be in the first normal form (1NF).

DE Erklären Sie in einfachem Deutsch, welche Bedingungen eine Tabelle erfüllen muss, um in der ersten Normalform (1NF) zu sein.

7.2 Second Normal Form / Zweite Normalform (Q164)

EN Explain in plain English what conditions a table must fulfill to be in the second normal form (2NF).

DE Erklären Sie in einfachem Deutsch, welche Bedingungen eine Tabelle erfüllen muss, um in der zweiten Normalform (2NF) zu sein.

7.3 Third Normal Form / Erste Normalform (Q165)

EN Explain in plain English what conditions a table must fulfill to be in the third normal form (3NF).

DE Erklären Sie in einfachem Deutsch, welche Bedingungen eine Tabelle erfüllen muss, um in der dritten Normalform (3NF) zu sein.

7.4 Functional Dependencies / Funktionale Abhängigkeiten (Q166)

EN Explain in plain English what conditions two columns X and Y must fulfill such that Y is functionaly dependent on X, i.e., $Y \rightarrow X$.

DE Erklären Sie in einfachem Deutsch, welche Bedingungen zwei Spalten X und Y erfüllen müssen, so das Y funktional von X abhängt, also das $Y \rightarrow X$.

7.5 Insertion Anomaly / Einfüge-Anomalie (Q167)

EN Explain in plain English what an insertion anomaly is.

DE Erklären Sie in einfachem Deutsch was eine Einfüge-Anomalie ist.

7.6 Deletion Anomaly / Lösch-Anomalie (Q168)

EN Explain in plain English what an deletion anomaly is.

DE Erklären Sie in einfachem Deutsch was eine Lösch-Anomalie ist.

7.7 Update Anomaly / Update-Anomalie (Q169)

EN Explain in plain English what an update anomaly is.

DE Erklären Sie in einfachem Deutsch was eine Update-Anomalie ist.

7.8 Functional Dependencies / Funktionale Abhängigkeiten (Q170)

EN Assume that a table stors data about people in the columns given below. Explain which columns are probably functionally dependent on which other columns.

- national_id: the Chinese mainland ID number,
- mobile: the mobile phone number,
- name: the full name of the person,
- dob: the date of birth of the person,
- province: the province where the person lives,
- city: the city where the person lives,
- address: the street address of the person.

DE Nehmen Sie an, dass eine Tabelle Daten über Leute in den Spalten wie unten angegeben speichert. Erklären Sie, welche Spalten wahrscheinlich funktional von welchen anderen Spalten abhängen werden.

- national_id: die chinesische Ausweisnummer,
- mobile: die Mobiltelefonnummer,
- name: der volle Name der Person,
- dob: das Geburtsdatum der Person,
- province: die Provinz, in der die Person lebt,
- city: die Stadt, in der die Person lebt,
- address: die Adresse (Straße, Hausnummer, ...), unter der die Person lebt.

7.9 Functional Dependencies and Keys / Funktionale Abhängigkeiten und Schlüssel (Q171)

Lecture	Room	Day	Lesson	Students	Professor
Databases	202	Mon	2	40	Weise
Deep Learning	201	Tue	2	30	Wu
C Programming	202	Wed	5	55	Zhou
Databases	206	Wed	2	40	Weise
System's Security	208	Thu	2	36	Wang
Deep Learning	201	Fri	4	30	Wu
System's Security	210	Fri	2	36	Wang
Embedded Systems	210	Fri	4	25	Chen
Python Programming	204	Fri	4	60	Weise

EN Find all the meaningful functional dependencies in the table above. Which candidate keys do exist in this table?

DE Finden Sie all sinnvollen funktionalen Abhängigkeiten in der Tabelle oben. Welche Schlüsselkandidaten gibt es?

7.10 First Normal Form / Erste Normalform (Q172)

EN The table *guest* has the columns guest_id, guest_family_name, guest_given_name, guest_address_1, guest_address_2, guest_mobile_1, guest_mobile_2. It is used to store the information of guests of a hotel.

1. Find the proper key for the table.
2. What happens if a guest wants to provide three contact addresses?
3. Transform the table to the first normal form (1NF).

DE Die Tabelle *gast* hat die Spalten gast_id, gast_familienname, guest_vorname, gast_adresse_1, gast_adresse_2, gast_mobil_1, gast_mobil_2. Sie wird verwendet, um die Informationen über Gäste eines Hotels zu speichern.

1. Finden Sie die passenden Schlüssel der Tabelle.
2. Was passiert, wenn ein Gast drei Kontaktadressen hinterlegen möchte?
3. Überführen Sie die Tabelle in die erste Normalform (1NF).

7.11 Second Normal Form / Zweite Normalform (Q173)

EN The table *occupation* has the columns guest_id, room_id, day, guest_family_name, and guest_given_name. It is used to store the day-to-day occupation in the rooms of a hotel.

1. Find the proper key(s) for the table.
2. Analyze the table for potential anomalies.
3. Transform the table to the second normal form (2NF).

DE Die Tabelle *belegung* hat die Spalten gast_id, raum_id, tag, gast_familienname und gast_vorname. Sie wird verwendet, um die tägliche Belegung der Räume eines Hotels zu speichern.

1. Finden Sie die passenden Schlüssel der Tabelle.
2. Analysieren Sie die Tabelle auf mögliche Anomalien.
3. Überführen Sie die Tabelle in die zweite Normalform (2NF).

7.12 Third Normal Form / Dritte Normalform (Q174)

EN The table *bill* has the columns bill_id, guest_id, guest_name, guest_address, and amount. The primary key is bill_id. The table is used to store they billing information of a hotel.

1. Is this table in the second normal form (2NF)? Why/why not?
2. Analyze the table for potential anomalies.
3. Transform the table to the third normal form (3NF).

DE Die Tabelle *rechnung* hat die Spalten rechnungs_id, gast_id, gast_name, gast_adresse und betrag. Der Primärschlüssel ist rechnungs_id. Die Tabelle wird verwendet, um die Rechnungsinformation eines Hotels zu speichern.

1. Ist die Tabelle in der zweiten Normalform (2NF)? Warum/Warum nicht?
2. Analysieren Sie die Tabelle auf mögliche Anomalien.
3. Überführen Sie die Tabelle in die dritte Normalform (3NF).

Backmatter

Glossary

1NF The first normal form (NF) in relational DBs [25, 28, 34, 48].

2NF The second normal form (NF) in relational DBs [23, 26, 28, 34, 48].

3NF The third normal form (NF) in relational DBs [23, 26, 28, 34, 48].

Bash is a the shell used under `Ubuntu` `Linux`, i.e., the program that "runs" in the terminal and interprets your commands, allowing you to start and interact with other programs [14, 59, 98]. Learn more at `https://www.gnu.org/software/bash`.

client In a client-server architecture, the client is a device or process that requests a service from the server. It initiates the communication with the server, sends a request, and receives the response with the result of the request. Typical examples for clients are web browsers in the internet as well as clients for DBMSes, such as `psql`.

client-server architecture is a system design where a central server receives requests from one or multiple clients [9, 53, 62, 65, 68]. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of DB software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as `psql`, or the different applications that access the DBs.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases* [90].

DBMS A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB [95].

DBS A *database system* is the combination of a DB and a the corresponding DBMS, i.e., basically, an installation of a DBMS on a computer together with one or multiple DBs. DBS = DB + DBMS.

ERD Entity relationship diagrams show the relationships between objects, e.g., between the tables in a DB and how they reference each other [4, 15, 18–20, 47, 72, 92]

FD A *functional dependency* exists between two groups of attributes Y and X if the values of X determine the values of Y. This is written as $X \to Y$.

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes [77, 85]. Learn more at `https://git-scm.com`.

GitHub is a website where software projects can be hosted and managed via the Git VCS [63, 85]. Learn more at `https://github.com`.

IT information technology

LAMP Stack A system setup for web applications: `Linux`, Apache (a web server), `MySQL`, and the server-side scripting language PHP [16, 42].

LibreOffice is on open source office suite [37, 52, 71] which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice `Writer`, LibreOffice `Calc`, and LibreOffice `Base`. See [90] for more information and installation instructions.

LibreOffice Base is a DBMS that can work on stand-alone files but also connect to other popular relational databases [35, 71]. It is part of LibreOffice [37, 52, 71] and has functionality that is comparable to Microsoft `Access` [7, 21, 86].

LibreOffice Calc is a spreadsheet software that allows you to arrange and perform calculations with data in a tabular grid. It is a free and open source spread sheet software [52, 71], i.e., an alternative to Microsoft `Excel`. It is part of LibreOffice [37, 52, 71].

LibreOffice Writer is a free and open source text writing program [97] and part of LibreOffice [37, 52, 71]. It is a good alternative to Microsoft `Word`.

Linux is the leading open source operating system, i.e., a free alternative for Microsoft `Windows` [5, 41, 76, 84, 89]. We recommend using it for this course, for software development, and for research. Learn more at `https://www.linux.org`. Its variant `Ubuntu` is particularly easy to use and install.

MariaDB An open source relational database management system that has forked off from `MySQL` [2, 3, 6, 33, 55, 66]. See `https://mariadb.org` for more information.

Microsoft Access is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases [7, 21, 56, 86]. It is part of Microsoft Office. A free and open source alternative to this commercial software is LibreOffice `Base`.

Microsoft Excel is a spreadsheet program that allows users to store, organize, manipulate, and calculate data in tabular structures [10, 38, 50]. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice `Calc` [52, 71].

Microsoft Office is a commercial suite of office software, including Microsoft `Excel`, Microsoft `Word`, and Microsoft `Access` [50]. LibreOffice is a free and open source alternative.

Microsoft Windows is a commercial proprietary operating system [13]. It is widely spread, but we recommend using a `Linux` variant such as `Ubuntu` for software development and for our course. Learn more at `https://www.microsoft.com/windows`.

Microsoft Word is one of the leading text writing programs [32, 58, 97] and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice `Writer`.

MySQL An open source relational database management system [12, 33, 67, 83, 94]. `MySQL` is famous for its use in the `LAMP` Stack. See `https://www.mysql.com` for more information.

NF The *normal forms* define guidelines for the design of relational DBs with the goal to avoid redundancy and to prevent inconsistencies and anomalies [28, 34, 48, 74, 75]. There are several normal forms, first normal form (1NF), second normal form (2NF), third normal form (3NF), and so on, each more restrictive than the other.

OSS Open source software, i.e., software that can freely be used, whose source code is made availabe in the internet, and which is usually developed cooperatively over the internet as well [43]. Typical examples are `Python`, `Linux`, Git, and `PostgreSQL`.

PgModeler the `PostgreSQL` DB modeler is a tool that allows for graphical modeling of logical schemas for DBs using an entity relationship diagram (ERD)-like notation [1]. Learn more at `https://pgmodeler.io`.

PostgreSQL An open source object-relational DBMS [36, 60, 64, 83]. See `https://postgresql.org` for more information.

psql is the client program used to access the `PostgreSQL` DBMS server.

Python The Python programming language [44, 51, 54, 91], i.e., what you will learn about in our book [91]. Learn more at https://python.org.

relational database A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other [25, 39, 40, 78, 82, 90, 93].

server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carring the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers [16] in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the "server computer" [49].

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases [17, 27, 29, 30, 45, 57, 79–82]. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference [79].

`stderr` The *standard error stream* is one of the three pre-defined streams of a console process (together with the standard input stream (stdin) and the stdout) [46]. It is the text stream to which the process writes information about errors and exceptions. If an uncaught `Exception` is raised in Python and the program terminates, then this information is written to standard error stream (stderr). If you run a program in a terminal, then the text that a process writes to its stderr appears in the console.

`stdin` The *standard input stream* is one of the three pre-defined streams of a console process (together with the stdout and the stderr) [46]. It is the text stream from which the process reads its input text, if any. The Python instruction `input` reads from this stream. If you run a program in a terminal, then the text that you type into the terminal while the process is running appears in this stream.

`stdout` The *standard output stream* is one of the three pre-defined streams of a console process (together with the stdin and the stderr) [46]. It is the text stream to which the process writes its normal output. The `print` instruction of Python writes text to this stream. If you run a program in a terminal, then the text that a process writes to its stdout appears in the console.

terminal A terminal is a text-based window where you can enter commands and execute them [5, 22]. Knowing what a terminal is and how to use it is very essential in any programming- or system administration-related task. If you want to open a terminal under Microsoft Windows, you can press ⊞+R, type in `cmd`, and hit ↵. Under Ubuntu Linux, Ctrl+Alt+T opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux [22, 42]. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at https://ubuntu.com. If you are in China, you can download it from https://mirrors.ustc.edu.cn/ubuntu-releases.

UML The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems [11, 61, 87, 88]

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code [85]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

WWW World Wide Web [8, 31]

yEd is a graph editor for high-quality graph-based diagrams [70, 96], suitable to draw, e.g., technology-independent ERDs, control flow charts, or Unified Modeling Language (UML) class diagrams. An online version of the editor is available at https://www.yworks.com/yed-live. Learn more at https://www.yworks.com/products/yed.

Bibliography

[1] Raphael "rkhaotix" Araújo e Silva. *pgModeler – PostgreSQL Database Modeler*. Palmas, Tocantins, Brazil, 2006–2025. URL: https://pgmodeler.io (visited on 2025-04-12) (cit. on p. 73).

[2] Adam Aspin and Karine Aspin. *Query Answers with MariaDB – Volume I: Introduction to SQL Queries*. Tetras Publishing, Oct. 2018. ISBN: 978-1-9996172-4-0. See also [3] (cit. on pp. 73, 76).

[3] Adam Aspin and Karine Aspin. *Query Answers with MariaDB – Volume II: In-Depth Querying*. Tetras Publishing, Oct. 2018. ISBN: 978-1-9996172-5-7. See also [2] (cit. on pp. 73, 76).

[4] Richard Barker. *Case*Method: Entity Relationship Modelling (Oracle)*. 1st ed. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., Jan. 1990. ISBN: 978-0-201-41696-1 (cit. on p. 72).

[5] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (cit. on pp. 73, 74).

[6] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Oct. 2019. ISBN: 978-1-4842-5514-8 (cit. on p. 73).

[7] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, Mar. 2020. ISBN: 978-1-83898-747-3 (cit. on p. 73).

[8] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) and Newsgroups: alt.hypertext, Aug. 6, 1991. URL: https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt (visited on 2025-02-05) (cit. on p. 74).

[9] Alex Berson. *Client/Server Architecture*. 2nd ed. Computer Communications Series. New York, NY, USA: McGraw-Hill, Mar. 29, 1996. ISBN: 978-0-07-005664-0 (cit. on p. 72).

[10] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (cit. on p. 73).

[11] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language Reference Manual*. 1st ed. Reading, MA, USA: Addison-Wesley Professional, Jan. 1999. ISBN: 978-0-201-57168-4 (cit. on p. 74).

[12] Silvia Botros and Jeremy Tinley. *High Performance MySQL*. 4th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Nov. 2021. ISBN: 978-1-4920-8051-0 (cit. on p. 73).

[13] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (cit. on p. 73).

[14] Ron Brash and Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, July 2018. ISBN: 978-1-78862-936-2 (cit. on p. 72).

[15] Ben Brumm. "A Guide to the Entity Relationship Diagram (ERD)". In: *Database Star*. Armadale, VIC, Australia: Elevated Online Services PTY Ltd., July 30, 2019–Dec. 23, 2023. URL: https://www.databasestar.com/entity-relationship-diagram (visited on 2025-03-29) (cit. on p. 72).

[16] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, June 2022 (cit. on pp. 72, 74).

[17] Donald D. Chamberlin. "50 Years of Queries". *Communications of the ACM (CACM)* 67(8):110–121, Aug. 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/3649887. URL: https://cacm.acm.org/research/50-years-of-queries (visited on 2025-01-09) (cit. on p. 74).

[18] Peter Pin-Shan Chen. "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned". In: *Software Pioneers: Contributions to Software Engineering*. Ed. by Manfred Broy and Ernst Denert. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, Feb. 2002, pp. 296–310. doi:10.1007/978-3-642-59412-0_17. URL: http://bit.csc.lsu.edu/%7Echen/pdf/Chen_Pioneers.pdf (visited on 2025-03-06) (cit. on p. 72).

[19] Peter Pin-Shan Chen. "The Entity-Relationship Model – Toward a Unified View of Data". *ACM Transactions on Database Systems (TODS)* 1(1):9–36, Mar. 1976. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0362-5915. doi:10.1145/320434.320440 (cit. on pp. 72, 77).

[20] Peter Pin-Shan Chen. "The Entity-Relationship Model: Toward a Unified View of Data". In: *1st International Conference on Very Large Data Bases (VLDB'1975)*. Sept. 22–24, 1975, Framingham, MA, USA. Ed. by Douglas S. Kerr. New York, NY, USA: Association for Computing Machinery (ACM), 1975, p. 173. ISBN: 978-1-4503-3920-9. doi:10.1145/1282480.1282492. See [19] for a more comprehensive introduction. (Cit. on p. 72).

[21] Christmas, FL, USA: Simon Sez IT. *Microsoft Access 2021 – Beginner to Advanced*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2023. ISBN: 978-1-83546-911-8 (cit. on p. 73).

[22] David Clinton and Christopher Negus. *Ubuntu Linux Bible*. 10th ed. Bible Series. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., Nov. 10, 2020. ISBN: 978-1-119-72233-5 (cit. on p. 74).

[23] Edgar Frank "Ted" Codd. "Normalized Data Base Structure: A Brief Tutorial". In: *ACM SIGFIDET Workshop on Data Description, Access, and Control*. Nov. 11–12, 1971, San Diego, CA, USA. Ed. by Edgar Frank "Ted" Codd and Albert L. Dean Jr. New York, NY, USA: Association for Computing Machinery (ACM), 1971, pp. 1–17. ISBN: 978-1-4503-7300-5. doi:10.1145/1734714.1734716. See also [24] (cit. on p. 72).

[24] Edgar Frank "Ted" Codd. *Normalized Data Base Structure: A Brief Tutorial*. IBM Research Report RJ935. San Jose, CA, USA: IBM Research Laboratory, 1971. URL: https://www.fsmwarden.com/Codd/Normalized%20data%20base%20structure_%20a%20brief%20tutorial(1971,%20nov).pdf (visited on 2025-05-04) (cit. on p. 77).

[25] Edgar Frank "Ted" Codd. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM (CACM)* 13(6):377–387, June 1970. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/362384.362685. URL: https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf (visited on 2025-01-05) (cit. on pp. 72, 74).

[26] Edgar Frank "Ted" Codd. *Further Normalization of the Data Base Relational Model*. IBM Research Report RJ909. San Jose, CA, USA: IBM Research Laboratory, Aug. 31, 1971. URL: https://forum.thethirdmanifesto.com/wp-content/uploads/asgarosforum/987737/00-efc-further-normalization.pdf (visited on 2025-05-04). Reprinted in and presented at [69] (cit. on p. 72).

[27] *Database Language SQL*. Tech. rep. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (cit. on p. 74).

[28] Christopher J. Date. *An Introduction to Database Systems*. 8th ed. Hoboken, NJ, USA: Pearson Education, Inc., July 2003. ISBN: 978-0-321-19784-9 (cit. on pp. 72, 73).

[29] Matt David and Blake Barnhill. *How to Teach People SQL*. San Francisco, CA, USA: The Data School, Chart.io, Inc., Dec. 10, 2019–Apr. 10, 2023. URL: https://dataschool.com/how-to-teach-people-sql (visited on 2025-02-27) (cit. on p. 74).

[30] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (cit. on p. 74).

[31] Paul Deitel, Harvey Deitel, and Abbey Deitel. *Internet & World Wide Web: How to Program*. 5th ed. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (cit. on p. 74).

[32] Pooyan Doozandeh and Frank E. Ritter. "Some Tips for Academic Writing and Using Microsoft Word". *XRDS: Crossroads, The ACM Magazine for Students* 26(1):10–11, Aut. 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1528-4972. doi:10.1145/3351470 (cit. on p. 73).

[33] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2015. ISBN: 978-1-4493-6290-4 (cit. on p. 73).

[34] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. 7th ed. Hoboken, NJ, USA: Pearson Education, Inc., June 2015. ISBN: 978-0-13-397077-7 (cit. on pp. 72, 73).

[35] Steve Fanning, Vasudev Narayanan, "flywire", Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen, and Randolph Gam. *Base Guide 7.3. Revision 1. Based on LibreOffice 7.3 Community*. Berlin, Germany: The Document Foundation, Aug. 2022. URL: https://books.libreoffice.org/en/BG73/BG73-BaseGuide.pdf (visited on 2025-01-13) (cit. on p. 73).

[36] Luca Ferrari and Enrico Pirozzi. *Learn PostgreSQL*. 2nd ed. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2023. ISBN: 978-1-83763-564-1 (cit. on p. 73).

[37] Jonas Gamalielsson and Björn Lundell. "Long-Term Sustainability of Open Source Software Communities beyond a Fork: A Case Study of LibreOffice". In: *8th IFIP WG 2.13 International Conference on Open Source Systems: Long-Term Sustainability OSS'2012*. Sept. 10–13, 2012, Hammamet, Tunisia. Ed. by Imed Hammouda, Björn Lundell, Tommi Mikkonen, and Walt Scacchi. Vol. 378. Vol. 378 of IFIP Advances in Information and Communication Technology (IFIPAICT). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2012, pp. 29–47. ISSN: 1868-4238. ISBN: 978-3-642-33441-2. doi:10.1007/978-3-642-33442-9_3 (cit. on p. 73).

[38] Dawn Griffiths. *Excel Cookbook – Receipts for Mastering Microsoft Excel*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2024. ISBN: 978-1-0981-4332-9 (cit. on p. 73).

[39] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. 3rd ed. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, July 2024. ISBN: 978-0-443-23791-1 (cit. on p. 74).

[40] Jan L. Harrington. *Relational Database Design and Implementation*. 4th ed. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Apr. 2016. ISBN: 978-0-12-849902-3 (cit. on p. 74).

[41] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (cit. on p. 73).

[42] Matthew Helmke. *Ubuntu Linux Unleashed 2021 Edition*. 14th ed. Reading, MA, USA: Addison-Wesley Professional, Aug. 2020. ISBN: 978-0-13-668539-5 (cit. on pp. 72, 74).

[43] Manuel Hoffmann, Frank Nagle, and Yanuo Zhou. *The Value of Open Source Software*. Working Paper 24-038. Boston, MA, USA: Harvard Business School, Jan. 1, 2024. URL: https://www.hbs.edu/ris/Publication%20Files/24-038_51f8444f-502c-4139-8bf2-56eb4b65c58a.pdf (visited on 2025-06-04) (cit. on p. 73).

[44] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 74).

[45] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), June 2023. URL: https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip (visited on 2025-01-08). Consists of several parts, see https://modern-sql.com/standard for information where to obtain them. (Cit. on p. 74).

[46] "stderr, stdin, stdout – Standard I/O Streams". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Ed. by Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and San Francisco, CA, USA: The Open Group, Aug. 8, 2024. URL: https://pubs.opengroup.org/onlinepubs/9799919799/functions/stdin.html (visited on 2024-10-30) (cit. on p. 74).

[47] Shannon Kempe and Paul Williams. *A Short History of the ER Diagram and Information Modeling*. Studio City, CA, USA: Dataversity Digital LLC, Sept. 25, 2012. URL: https://www.dataversity.net/a-short-history-of-the-er-diagram-and-information-modeling (visited on 2025-03-06) (cit. on p. 72).

[48] William (Bill) Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory". *Communications of the ACM (CACM)* 26(2):120–125, Sept. 1982–Feb. 1983. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/358024.358054. URL: https://www.cs.dartmouth.edu/~cs61/Resources/Papers/CACM%20Kent%20Five%20Normal%20Forms.pdf (visited on 2025-05-03) (cit. on pp. 72, 73).

[49] Jay LaCroix. *Mastering Ubuntu Server*. 4th ed. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2022. ISBN: 978-1-80323-424-3 (cit. on p. 74).

[50] Joan Lambert and Curtis Frye. *Microsoft Office Step by Step (Office 2021 and Microsoft 365)*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., June 2022. ISBN: 978-0-13-754493-6 (cit. on p. 73).

[51] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 74).

[52] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: https://www.libreoffice.org (visited on 2024-12-12) (cit. on p. 73).

[53] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra, and William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Ed. by The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., Jan. 3, 2025. URL: https://www.britannica.com/technology/client-server-architecture (visited on 2025-01-20) (cit. on p. 72).

[54] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 74).

[55] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (visited on 2025-04-24) (cit. on p. 73).

[56] Ron McFadyen and Cindy Miller. *Relational Databases and Microsoft Access*. 3rd ed. Palatine, IL, USA: Harper College, 2014–2019. URL: https://harpercollege.pressbooks.pub/relationaldatabases (visited on 2025-04-11) (cit. on p. 73).

[57] Jim Melton and Alan R. Simon. *SQL: 1999 – Understanding Relational Language Components*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, June 2001. ISBN: 978-1-55860-456-8 (cit. on p. 74).

[58] *Microsoft Word*. Redmond, WA, USA: Microsoft Corporation, 2024. URL: https://www.microsoft.com/en-us/microsoft-365/word (visited on 2024-12-12) (cit. on p. 73).

[59] Cameron Newham and Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3rd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (cit. on p. 72).

[60] Regina O. Obe and Leo S. Hsu. *PostgreSQL: Up and Running*. 3rd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Oct. 2017. ISBN: 978-1-4919-6336-4 (cit. on p. 73).

[61] *OMG® Unified Modeling Language® (OMG UML®). Version 2.5.1*. OMG Document formal/2017-12-05. Milford, MA, USA: Object Management Group, Inc. (OMG), Dec. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (visited on 2025-03-30) (cit. on p. 74).

[62] Robert Orfali, Dan Harkey, and Jeri Edwards. *Client/Server Survival Guide*. 3rd ed. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., Jan. 25, 1999. ISBN: 978-0-471-31615-2 (cit. on p. 72).

[63] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin, and Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), July 14, 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (cit. on p. 72).

[64] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2024 (cit. on p. 73).

[65] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu, and Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6 (cit. on p. 72).

[66] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2014. ISBN: 978-1-78398-154-0 (cit. on p. 73).

[67] Mike Reichardt, Michael Gundall, and Hans D. Schotten. "Benchmarking the Operation Times of NoSQL and MySQL Databases for Python Clients". In: *47th Annual Conference of the IEEE Industrial Electronics Society (IECON'2021*. Oct. 13–15, 2021, Toronto, ON, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2021, pp. 1–8. ISSN: 2577-1647. ISBN: 978-1-6654-3554-3. doi:10.1109/IECON48115.2021.9589382 (cit. on p. 73).

[68] Mark Richards and Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: 978-1-4920-4345-4 (cit. on p. 72).

[69] Randall Rustin, ed. *Data Base Systems: Courant Computer Science Symposium 6*. May 24–25, 1971, New York, NY, USA. Englewood Cliffs, NJ, USA: Prentice-Hall, 1972. ISBN: 978-0-13-196741-0 (cit. on p. 77).

[70] Matthias Sedlmeier and Martin Gogolla. "Model Driven ActiveRecord with yEd". In: *25th International Conference on Information Modelling and Knowledge Bases XXVII (EJC'2015)*. June 8–12, 2015, Maribor, Štajerska, Podravska, Slovenia. Ed. by Tatjana Welzer, Hannu Jaakkola, Bernhard Thalheim, Yasushi Kiyoki, and Naofumi Yoshida. Vol. 280 of Frontiers in Artificial Intelligence and Applications. Amsterdam, The Netherlands: IOS Press BV, 2015, pp. 65–76. ISSN: 0922-6389. ISBN: 978-1-61499-610-1. doi:10.3233/978-1-61499-611-8-65 (cit. on p. 75).

[71] Winfried Seimert. *LibreOffice 7.3 – Praxiswissen für Ein- und Umsteiger*. Blaufelden, Schwäbisch Hall, Baden-Württemberg, Germany: mitp Verlags GmbH & Co. KG, Apr. 2022. ISBN: 978-3-7475-0504-5 (cit. on p. 73).

[72] Yuriy Shamshin. "Conceptual Database Model. Entity Relationship Diagram (ERD)". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, May 2024. Chap. 04. URL: https://dbs.academy.lv/lection/dbs_LS04EN_erd.pdf (visited on 2025-03-29) (cit. on p. 72).

[73] Yuriy Shamshin. *Databases*. Riga, Latvia: ISMA University of Applied Sciences, May 2024. URL: https://dbs.academy.lv (visited on 2025-01-11).

[74] Yuriy Shamshin. "Normalization". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, May 2024. Chap. 07a. URL: https://dbs.academy.lv/lection/dbs_LS07ENa_normalization.pdf (visited on 2025-05-03) (cit. on p. 73).

[75] Yuriy Shamshin. "RDM Normalization. Data Anomalies. Functional Dependency. Normal Forms." In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, May 2024. Chap. 07. URL: https://dbs.academy.lv/lection/dbs_LS07EN_normalization.pdf (visited on 2025-05-03) (cit. on p. 73).

[76] Ellen Siever, Stephen Figgins, Robert Love, and Arnold Robbins. *Linux in a Nutshell*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Sept. 2009. ISBN: 978-0-596-15448-6 (cit. on p. 73).

[77] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2023. ISBN: 978-1-0981-3391-7 (cit. on p. 72).

[78] John Miles Smith and Philip Yen-Tang Chang. "Optimizing the Performance of a Relational Algebra Database Interface". *Communications of the ACM (CACM)* 18(10):568–579, Oct. 1975. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/361020.361025 (cit. on p. 74).

[79] "SQL Commands". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), Feb. 20, 2025. Chap. Part VI. Reference. URL: https://www.postgresql.org/docs/17/sql-commands.html (visited on 2025-02-25) (cit. on p. 74).

[80] Ryan K. Stephens and Ronald R. Plew. *Sams Teach Yourself SQL in 21 Days*. 4th ed. Sams Tech Yourself. Indianapolis, IN, USA: SAMS Technical Publishing and Hoboken, NJ, USA: Pearson Education, Inc., Oct. 2002. ISBN: 978-0-672-32451-2 (cit. on pp. 74, 81).

[81] Ryan K. Stephens, Ronald R. Plew, Bryan Morgan, and Jeff Perkins. *SQL in 21 Tagen. Die Datenbank-Abfragesprache SQL vollständig erklärt (in 14/21 Tagen)*. 6th ed. Burgthann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of [80] (cit. on p. 74).

[82] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sept. 2018. ISBN: 978-1-4842-3841-7 (cit. on p. 74).

[83] Alkin Tezuysal and Ibrar Ahmed. *Database Design and Modeling with PostgreSQL and MySQL*. Birmingham, England, UK: Packt Publishing Ltd, July 2024. ISBN: 978-1-80323-347-5 (cit. on p. 73).

[84] Linus Torvalds. "The Linux Edge". *Communications of the ACM (CACM)* 42(4):38–39, Apr. 1999. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/299157.299165 (cit. on p. 73).

[85] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, Mar. 2024. ISBN: 979-8-8688-0215-7 (cit. on pp. 72, 74).

[86] Laurie A. Ulrich and Ken Cook. *Access For Dummies*. Hoboken, NJ, USA: For Dummies (Wiley), Dec. 2021. ISBN: 978-1-119-82908-9 (cit. on p. 73).

[87] *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. The Addison-Wesley Object Technology Series. Reading, MA, USA: Addison-Wesley Professional, Sept. 2003. ISBN: 978-0-321-19368-1 (cit. on p. 74).

[88] *UML Notation Guide*. Version 1.1. Santa Clara, CA, USA: Rational Software Corporation, Redmond, WA, USA: Microsoft Corporation, Palo Alto, CA, USA: Hewlett-Packard Company, Redwood Shores, CA, USA: Oracle Corporation, Dallas, TX, USA: Sterling Software, Ottawa, ON, Canada: MCI Systemhouse Corporation, Blue Bell, PA, USA: Unisys Corporation, Blue Bell, PA, USA: ICON Computing, Santa Clara, CA, USA: IntelliCorp, Burlington, MA, USA: i-Logix, Armonk, NY, USA: International Business Machines Corporation (IBM), Kanata, ON, Canada: ObjecTime Limited, Chicago, IL, USA: Platinum Technology Inc., Boston, MA, USA: Ptech Inc., Orlando, FL, USA: Taskon A/S, Paoli, PA, USA: Reich Technologies, and Paris, Île-de-France, France: Softeam, Sept. 1, 1997. URL: https://web.cse.msu.edu/~cse870/Materials/uml-notation-guide-9-97.pdf (visited on 2025-03-30) (cit. on p. 74).

[89] Sander van Vugt. *Linux Fundamentals*. 2nd ed. Hoboken, NJ, USA: Pearson IT Certification, June 2022. ISBN: 978-0-13-792931-3 (cit. on p. 73).

[90] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 1, ii, 72–74).

[91] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on p. 74).

[92] Matthew West. *Developing High Quality Data Models*. Version: 2.0, Issue: 2.1. London, England, UK: Shell International Limited and European Process Industries STEP Technical Liaison Executive (EPISTLE); Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Dec. 8, 1995–Dec. 2010. ISBN: 978-0-12-375107-2. URL: https://www.researchgate.net/publication/286610894 (visited on 2025-03-24). Edited by Julian Fowler (cit. on p. 72).

[93] *What is a Relational Database?* Armonk, NY, USA: International Business Machines Corporation (IBM), Oct. 20, 2021–Dec. 12, 2024. URL: https://www.ibm.com/think/topics/relational-databases (visited on 2025-01-05) (cit. on p. 74).

[94] Ulf Michael "Monty" Widenius, David Axmark, and Uppsala, Sweden: MySQL AB. *MySQL Reference Manual – Documentation from the Source*. Sebastopol, CA, USA: O'Reilly Media, Inc., July 9, 2002. ISBN: 978-0-596-00265-7 (cit. on p. 73).

[95] Kinza Yasar and Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., June 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (visited on 2025-01-11) (cit. on p. 72).

[96] *yEd Graph Editor Manual*. Tübingen, Baden-Württemberg, Germany: yWorks GmbH, 2011–2025. URL: https://yed.yworks.com/support/manual/index.html (visited on 2025-03-31) (cit. on p. 75).

[97] Pavlo V. Zahorodko and Pavlo V. Merzlykin. "An Approach for Processing and Document Flow Automation for Microsoft Word and LibreOffice Writer File Formats". In: *4th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW'2021)*. Dec. 18, 2021, Virtual Event and Kryvyi Rih, Ukraine. Ed. by Arnold E. Kiv, Serhiy O. Semerikov, Vladimir N. Soloviev, and Andrii M. Striuk. Vol. 3077 of CEUR Workshop Proceedings (CEUR-WS.org). Aachen, Nordrhein-Westfalen, Germany: CEUR-WS Team, Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen, 2022, pp. 66–82. ISSN: 1613-0073. URL: https://ceur-ws.org/Vol-3077/paper12.pdf (visited on 2025-10-04) (cit. on p. 73).

[98] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, June 2017. ISBN: 978-1-78439-687-9 (cit. on p. 72).