



合肥大學
HEFEI UNIVERSITY



Datenbanken

3. Anforderungen an ein DBMS

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Databases



Dies ist ein Kurs über Datenbanken an der Universität Hefei (合肥大学).

Die Webseite mit dem Lehrmaterial dieses Kurses ist <https://thomasweise.github.io/databases> (siehe auch den QR-Code unten rechts). Dort können Sie das Kursbuch (in Englisch) und diese Slides finden. Das Repository mit den Beispielen finden Sie unter <https://github.com/thomasWeise/databasesCode>.



Outline



1. Einleitung
2. Daten Modellieren und Repräsentieren
3. Unabhängigkeiten
4. Nutzbarkeit und Performanz
5. Integrität und Gleichzeitiger Zugriff
6. Dauerhaftigkeit und Sicherheit (Safety)
7. Datenschutz und Datensicherheit (Security)
8. Zusammenfassung





Einleitung



Anforderungen an ein DBMS

- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.



Anforderungen an ein DBMS



- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.
- Lassen Sie uns nun zusammen erforschen, welche Anforderungen Organisationen haben könnten, die mit DBs bzw. DBMSes arbeiten.

Anforderungen an ein DBMS



- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.
- Lassen Sie uns nun zusammen erforschen, welche Anforderungen Organisationen haben könnten, die mit DBs bzw. DBMSes arbeiten.
- Wir wollen jetzt also eine Vorstellung von den Features entwickeln, die ein DBMS haben soll und welche wir später nutzen werden.

Anforderungen an ein DBMS



- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.
- Lassen Sie uns nun zusammen erforschen, welche Anforderungen Organisationen haben könnten, die mit DBs bzw. DBMSes arbeiten.
- Wir wollen jetzt also eine Vorstellung von den Features entwickeln, die ein DBMS haben soll und welche wir später nutzen werden.
- Stellen wir uns dafür vor, dass wir eine Datenbank für eine Bank entwickeln wollen.

Anforderungen an ein DBMS



- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.
- Lassen Sie uns nun zusammen erforschen, welche Anforderungen Organisationen haben könnten, die mit DBs bzw. DBMSes arbeiten.
- Wir wollen jetzt also eine Vorstellung von den Features entwickeln, die ein DBMS haben soll und welche wir später nutzen werden.
- Stellen wir uns dafür vor, dass wir eine Datenbank für eine Bank entwickeln wollen.
- Datenbank soll die Informationen über die Kunden, ihre Konten und Transaktionen speichern, sowie die Daten über die Angestellten der Bank.

Anforderungen an ein DBMS



- Wir haben nun eine ungefähre Idee, was eine Datenbank (DB) und was ein Datenbankmanagementsystem (DBMS) sind.
- Lassen Sie uns nun zusammen erforschen, welche Anforderungen Organisationen haben könnten, die mit DBs bzw. DBMSes arbeiten.
- Wir wollen jetzt also eine Vorstellung von den Features entwickeln, die ein DBMS haben soll und welche wir später nutzen werden.
- Stellen wir uns dafür vor, dass wir eine Datenbank für eine Bank entwickeln wollen.
- Datenbank soll die Informationen über die Kunden, ihre Konten und Transaktionen speichern, sowie die Daten über die Angestellten der Bank.
- Davon ausgehend bauen wir uns eine Wunschliste von Features und Anforderungen zusammen.



Daten Modellieren und Repräsentieren



Daten Modellieren und Repräsentieren



- Zuerst muss uns das DBMS Werkzeuge bieten, mit dem wir ein Modell unserer Daten erstellen und implementieren können.

Daten Modellieren und Repräsentieren



- Zuerst muss uns das DBMS Werkzeuge bieten, mit dem wir ein Modell unserer Daten erstellen und implementieren können.
- Wenn wir mit einer Programmiersprache wie Python programmieren, dann können wir Datenstrukturen erstellen.

Daten Modellieren und Repräsentieren



- Zuerst muss uns das DBMS Werkzeuge bieten, mit dem wir ein Modell unserer Daten erstellen und implementieren können.
- Wenn wir mit einer Programmiersprache wie Python programmieren, dann können wir Datenstrukturen erstellen.
- Wir können sagen: „Das hier ist eine Liste mit Ganzzahlen.“

Daten Modellieren und Repräsentieren



- Zuerst muss uns das DBMS Werkzeuge bieten, mit dem wir ein Modell unserer Daten erstellen und implementieren können.
- Wenn wir mit einer Programmiersprache wie Python programmieren, dann können wir Datenstrukturen erstellen.
- Wir können sagen: „Das hier ist eine Liste mit Ganzzahlen.“ oder „Das hier ist eine Klasse, mit der Informationen über Studenten gespeichert werde. Jede Instanz davon speichert den Namen und die Ausweisnummer eines Studenten.“

Daten Modellieren und Repräsentieren



- Zuerst muss uns das DBMS Werkzeuge bieten, mit dem wir ein Modell unserer Daten erstellen und implementieren können.
- Wenn wir mit einer Programmiersprache wie Python programmieren, dann können wir Datenstrukturen erstellen.
- Wir können sagen: „Das hier ist eine Liste mit Ganzzahlen.“ oder „Das hier ist eine Klasse, mit der Informationen über Studenten gespeichert werde. Jede Instanz davon speichert den Namen und die Ausweisnummer eines Studenten.“
- Also wir brauchen sowas auch für Datenbanken.

Relationales Datenmodell und Logisches Schema



- Wir fokussieren uns hier auf relationale Datenmodelle.

Relationales Datenmodell und Logisches Schema



- Wir fokussieren uns hier auf relationale Datenmodelle.
- *Modellieren* bedeutet also zu definieren, welche Tabellen es gibt, welche Spalten diese Tabellen haben und welche Datentypen die Spalten haben, sowie festzulegen, wie die Tabellen und deren Datensätze zueinander in Beziehung stehen.

Relationales Datenmodell und Logisches Schema



- Wir fokussieren uns hier auf relationale Datenmodelle.
- *Modellieren* bedeutet also zu definieren, welche Tabellen es gibt, welche Spalten diese Tabellen haben und welche Datentypen die Spalten haben, sowie festzulegen, wie die Tabellen und deren Datensätze zueinander in Beziehung stehen.
- Diese Definitionen nennt man das *logische Schema* (auch: logisches Modell) der Datenbank.

Logisches Schema für die Bank



- Für unsere Bank bedeutet dies zum Beispiel

Logisches Schema für die Bank



- Für unsere Bank bedeutet dies zum Beispiel:
 - Wir wollen vielleicht eine Tabelle mit Kundeninformationen erstellen, eine Tabelle mit Informationen über Bankangestellte, eine Tabelle für Konten und eine Tabelle mit Transaktionen.

Logisches Schema für die Bank



- Für unsere Bank bedeutet dies zum Beispiel:
 - Wir wollen vielleicht eine Tabelle mit Kundeninformationen erstellen, eine Tabelle mit Informationen über Bankangestellte, eine Tabelle für Konten und eine Tabelle mit Transaktionen.
 - Wir wollen definieren, welche Informationen wir über Kunden speichern, z. B. ihre Namen, Ausweisnummern, Mobiltelefonnummern, usw.

Logisches Schema für die Bank



- Für unsere Bank bedeutet dies zum Beispiel:
 - Wir wollen vielleicht eine Tabelle mit Kundeninformationen erstellen, eine Tabelle mit Informationen über Bankangestellte, eine Tabelle für Konten und eine Tabelle mit Transaktionen.
 - Wir wollen definieren, welche Informationen wir über Kunden speichern, z. B. ihre Namen, Ausweisnummern, Mobiltelefonnummern, usw.
 - Wir müssen auch Einschränkungen (EN: *constraints*) für die Integrität der Daten definieren, z. B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)⁷⁹ entsprechen müssen, usw.

Logisches Schema für die Bank



- Für unsere Bank bedeutet dies zum Beispiel:
 - Wir wollen vielleicht eine Tabelle mit Kundeninformationen erstellen, eine Tabelle mit Informationen über Bankangestellte, eine Tabelle für Konten und eine Tabelle mit Transaktionen.
 - Wir wollen definieren, welche Informationen wir über Kunden speichern, z. B. ihre Namen, Ausweisnummern, Mobiltelefonnummern, usw.
 - Wir müssen auch Einschränkungen (EN: *constraints*) für die Integrität der Daten definieren, z. B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)⁷⁹ entsprechen müssen, usw.
 - Wir wollen auch Beziehungen zwischen den Tabellen spezifizieren, z. B. daß jeder Eintrag in der Tabelle für Bankkonten mit genau einem Eintrag in der Tabelle für Kunden verbunden ist, aber daß jeder Kunde mehrere Bankkonten haben darf.

Logisches Schema für die Bank

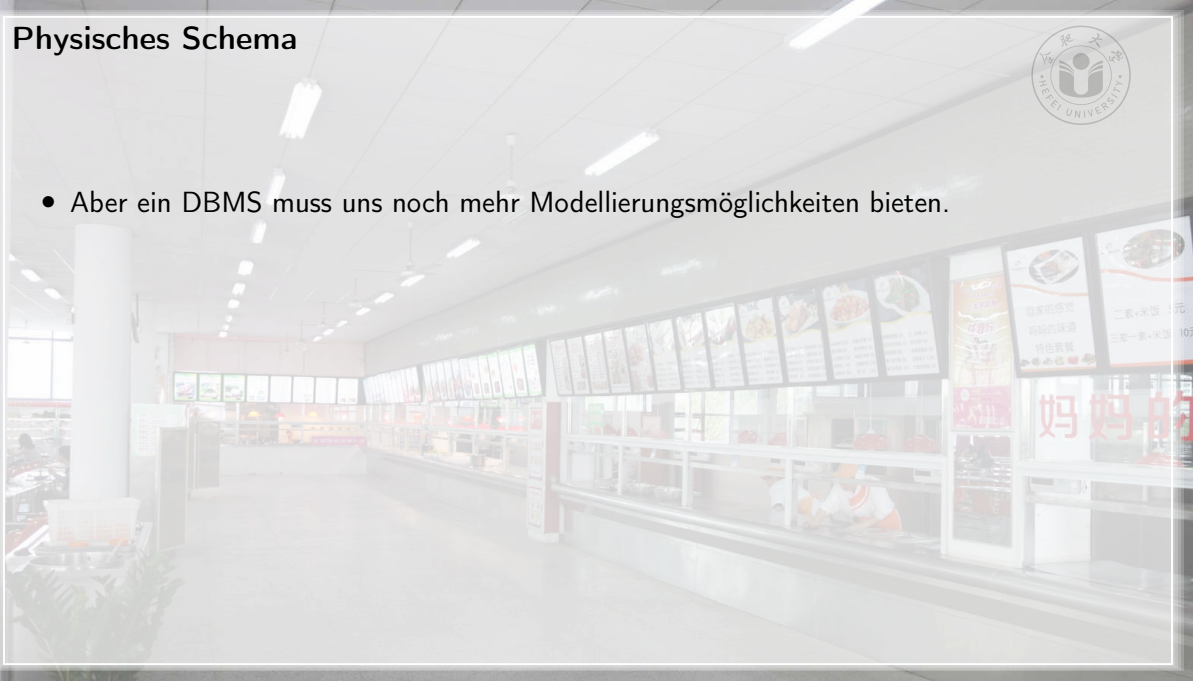


- Für unsere Bank bedeutet dies zum Beispiel:
 - Wir wollen vielleicht eine Tabelle mit Kundeninformationen erstellen, eine Tabelle mit Informationen über Bankangestellte, eine Tabelle für Konten und eine Tabelle mit Transaktionen.
 - Wir wollen definieren, welche Informationen wir über Kunden speichern, z. B. ihre Namen, Ausweisnummern, Mobiltelefonnummern, usw.
 - Wir müssen auch Einschränkungen (EN: *constraints*) für die Integrität der Daten definieren, z. B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)⁷⁹ entsprechen müssen, usw.
 - Wir wollen auch Beziehungen zwischen den Tabellen spezifizieren, z. B. daß jeder Eintrag in der Tabelle für Bankkonten mit genau einem Eintrag in der Tabelle für Kunden verbunden ist, aber daß jeder Kunde mehrere Bankkonten haben darf.
 - Idealerweise sollte es dafür eine Programmiersprache geben.

Physisches Schema



- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.



Physisches Schema

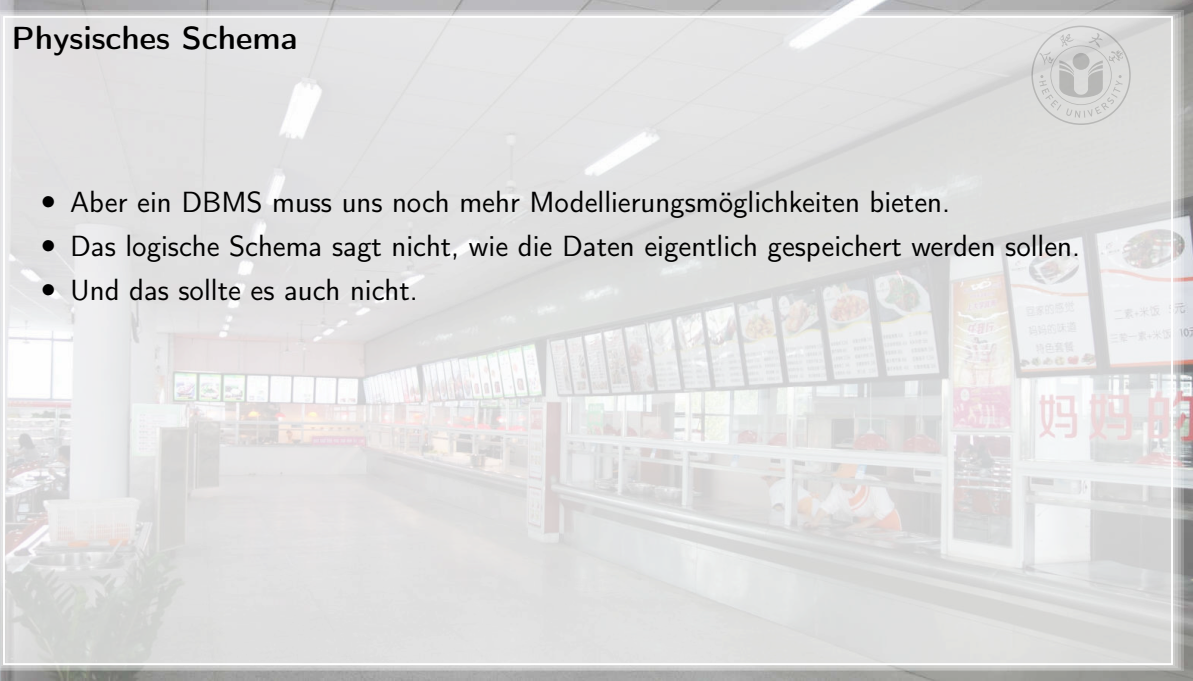


- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.
- Das logische Schema sagt nicht, wie die Daten eigentlich gespeichert werden sollen.

Physisches Schema



- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.
- Das logische Schema sagt nicht, wie die Daten eigentlich gespeichert werden sollen.
- Und das sollte es auch nicht.



Physisches Schema



- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.
- Das logische Schema sagt nicht, wie die Daten eigentlich gespeichert werden sollen.
- Und das sollte es auch nicht.
- Dafür gibt es darunterliegende *physische Schema*.

Physisches Schema



- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.
- Das logische Schema sagt nicht, wie die Daten eigentlich gespeichert werden sollen.
- Und das sollte es auch nicht.
- Dafür gibt es darunterliegende *physische Schema*.
- Ein DBMS soll es den Datenbankdesignern erlauben, Indices für Tabellen zu definieren, um den Zugriff auf Daten zu beschleunigen.

Physisches Schema



- Aber ein DBMS muss uns noch mehr Modellierungsmöglichkeiten bieten.
- Das logische Schema sagt nicht, wie die Daten eigentlich gespeichert werden sollen.
- Und das sollte es auch nicht.
- Dafür gibt es darunterliegende *physische Schema*.
- Ein DBMS soll es den Datenbankdesignern erlauben, Indices für Tabellen zu definieren, um den Zugriff auf Daten zu beschleunigen.
- Aber es sollte sie nicht zwingen, das genaue Layout der Daten auf der Festplatte zu kennen.



Unabhängigkeiten



Unabhängigkeiten

- Das logische und das physische Schema sind zwei verschiedene Dinge.



Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python: Die Hochsprachenkonstrukte wie Schleifen, Klassen und Exceptions existieren auf der Ebene von Python, aber letztendlich auf der Ebene der Hardware werden sie durch Maschinencode repräsentiert, der völlig anders aussehen kann

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python: Die Hochsprachenkonstrukte wie Schleifen, Klassen und Exceptions existieren auf der Ebene von Python, aber letztendlich auf der Ebene der Hardware werden sie durch Maschinencode repräsentiert, der völlig anders aussehen kann... was der Pythonprogrammierer aber nicht wissen muss.

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python: Die Hochsprachenkonstrukte wie Schleifen, Klassen und Exceptions existieren auf der Ebene von Python, aber letztendlich auf der Ebene der Hardware werden sie durch Maschinencode repräsentiert, der völlig anders aussehen kann... was der Pythonprogrammierer aber nicht wissen muss.
- Die Programme, die auf der Datenbank „sitzen“ greifen auf die Daten gemäß des logischen Schemas zu.

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python: Die Hochsprachenkonstrukte wie Schleifen, Klassen und Exceptions existieren auf der Ebene von Python, aber letztendlich auf der Ebene der Hardware werden sie durch Maschinencode repräsentiert, der völlig anders aussehen kann... was der Pythonprogrammierer aber nicht wissen muss.
- Die Programme, die auf der Datenbank „sitzen“ greifen auf die Daten gemäß des logischen Schemas zu.
- Das ist unabhängig vom physischen Schema, davon, wie die Daten auf der Festplatte organisiert sind.

Unabhängigkeiten



- Das logische und das physische Schema sind zwei verschiedene Dinge.
- Wenn wir Datenbankapplikationen entwerfen, gibt es mehrere Abstraktionsebenen, die zu einem gewissen Grad unabhängig von einander sein sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python: Die Hochsprachenkonstrukte wie Schleifen, Klassen und Exceptions existieren auf der Ebene von Python, aber letztendlich auf der Ebene der Hardware werden sie durch Maschinencode repräsentiert, der völlig anders aussehen kann... was der Pythonprogrammierer aber nicht wissen muss.
- Die Programme, die auf der Datenbank „sitzen“ greifen auf die Daten gemäß des logischen Schemas zu.
- Das ist unabhängig vom physischen Schema, davon, wie die Daten auf der Festplatte organisiert sind.
- Das physische Schema kann geändert werden, ohne die Programme, die auf dem logischen Schema „sitzen“, zu beeinflussen.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.
- Diese Sichten liegen „oberhalb“ des logischen Schemas.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.
- Diese Sichten liegen „oberhalb“ des logischen Schemas.
- Für unsere Bank bedeutet das zum Beispiel

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.
- Diese Sichten liegen „oberhalb“ des logischen Schemas.
- Für unsere Bank bedeutet das zum Beispiel
 - Eine Onlinebankinganwendung, wird nur die Kunden und deren Kontent sehen.

Drei-Schema-Architektur



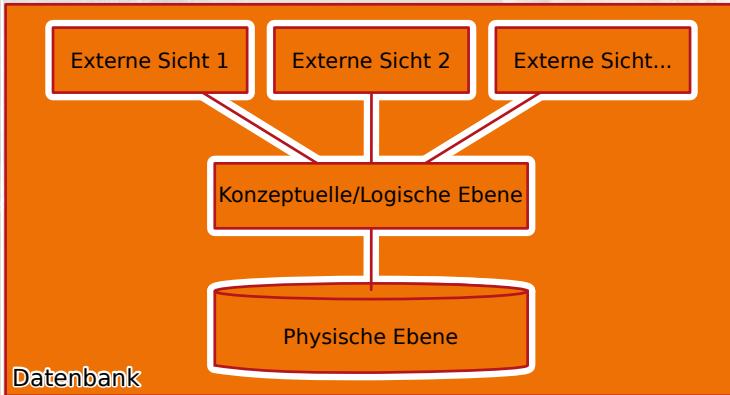
- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.
- Diese Sichten liegen „oberhalb“ des logischen Schemas.
- Für unsere Bank bedeutet das zum Beispiel
 - Eine Onlinebankinganwendung, wird nur die Kunden und deren Kontent sehen.
 - Das HR-Programm kann auf die Mitarbeiter-Daten unserer Bank und deren Performanz-Indikatoren zugreifen, aber nicht auf Kontostände der Kunden.

Drei-Schema-Architektur



- Datenbanken sind oft groß und erfüllen mehrere Funktionen auf einmal.
- Daher gibt es auch oft mehr als eine Anwendung, die auf die Datenbank zugreift.
- Jede Anwendung kann eine eigene Sicht auf die Daten haben.
- Diese Sichten liegen „oberhalb“ des logischen Schemas.
- Für unsere Bank bedeutet das zum Beispiel
 - Eine Onlinebanking-anwendung, wird nur die Kunden und deren Kontent sehen.
 - Das HR-Programm kann auf die Mitarbeiter-Daten unserer Bank und deren Performanz-Indikatoren zugreifen, aber nicht auf Kontostände der Kunden.
- Ein DBMS muss uns erlauben, so eine Drei-Schema-Architektur^{1,13,56,67} zu entwickeln, bei der jede Ebene möglichst unabhängig von der darunterliegenden sein soll.

Drei-Schema-Architektur



Verschiedene Sichten (views) auf die Daten:
Verschiedene Applikationen sehen verschiedene Untermengen der Daten.

Definition von allen Tabellen in der Datenbank und deren Relationen.

Wie die Daten tatsächlich gespeichert werden.
Dies geschieht versteckt von den Anwendungen.



Nutzbarkeit und Performanz



Nutzbarkeit und Performanz



- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.

Nutzbarkeit und Performanz



- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.
- Wir erwarten, dass die Lese- und Schreibgeschwindigkeit hoch sind.

Nutzbarkeit und Performanz



- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.
- Wir erwarten, dass die Lese- und Schreibgeschwindigkeit hoch sind.
- Wir erwarten, dass Daten schnell eingefügt, geändert und gelöscht werden können.

Nutzbarkeit und Performanz



- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.
- Wir erwarten, dass die Lese- und Schreibgeschwindigkeit hoch sind.
- Wir erwarten, dass Daten schnell eingefügt, geändert und gelöscht werden können.
- Eine einfache Programmiersprache für den Datenzugriff sollte existieren.

Nutzbarkeit und Performanz



- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.
- Wir erwarten, dass die Lese- und Schreibgeschwindigkeit hoch sind.
- Wir erwarten, dass Daten schnell eingefügt, geändert und gelöscht werden können.
- Eine einfache Programmiersprache für den Datenzugriff sollte existieren.
- Es sollte weitere Werkzeuge geben, mit denen die Datenmodellierung und der Zugriff weiter vereinfacht werden können.

- Ein DBMS stellt den Nutzern und Anwendungen Daten in einem brauchbaren Format und einer akzeptablen Geschwindigkeit zur Verfügung.
- Wir erwarten, dass die Lese- und Schreibgeschwindigkeit hoch sind.
- Wir erwarten, dass Daten schnell eingefügt, geändert und gelöscht werden können.
- Eine einfache Programmiersprache für den Datenzugriff sollte existieren.
- Es sollte weitere Werkzeuge geben, mit denen die Datenmodellierung und der Zugriff weiter vereinfacht werden können.
- Wir wollen Formulare zum Eingeben und Berichte zum Ausgeben der Daten erstellen können.



Integrität und Gleichzeitiger Zugriff



Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.
 - **Transaktionen:** Eine Änderung der Daten, die ggf. mehrere Schritte erfordert, kann in eine Transaktion gruppiert werden.

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.
 - **Transaktionen:** Eine Änderung der Daten, die ggf. mehrere Schritte erfordert, kann in eine Transaktion gruppiert werden.
 - **Atomizität:** Eine Transaktion findet entweder vollständig statt (alle Schritte werden erfolgreich ausgeführt) oder gar nicht (kein Schritt wird ausgeführt).

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.
 - **Transaktionen:** Eine Änderung der Daten, die ggf. mehrere Schritte erfordert, kann in eine Transaktion gruppiert werden.
 - **Atomizität:** Eine Transaktion findet entweder vollständig statt (alle Schritte werden erfolgreich ausgeführt) oder gar nicht (kein Schritt wird ausgeführt).

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.
 - **Transaktionen:** Eine Änderung der Daten, die ggf. mehrere Schritte erfordert, kann in eine Transaktion gruppiert werden.
 - **Atomizität:** Eine Transaktion findet entweder vollständig statt (alle Schritte werden erfolgreich ausgeführt) oder gar nicht (kein Schritt wird ausgeführt).
 - Das DBMS stellt sicher, dass Transaktionen nur ausgeführt werden, wenn die Konsistenz vor und nach der Transaktion eingehalten wird.

Integrität



- Ein DBMS stellt die Korrektheit und Integrität der Daten sicher.
- Dies beinhaltet mehrere Aspekte
 - **Konsistenz:** Das DBMS stellt sicher, dass die modellierten Beziehungen und Einschränkungen für die Daten immer eingehalten werden. Wenn wir festgelegt haben, dass ein Bankkonto immer einem Kunden gehört, dann darf das DBMS nie zulassen, dass irgendwie ein Bankkonto entsteht, das nicht einem Kunden gehört.
 - **Transaktionen:** Eine Änderung der Daten, die ggf. mehrere Schritte erfordert, kann in eine Transaktion gruppiert werden.
 - **Atomizität:** Eine Transaktion findet entweder vollständig statt (alle Schritte werden erfolgreich ausgeführt) oder gar nicht (kein Schritt wird ausgeführt).
 - Das DBMS stellt sicher, dass Transaktionen nur ausgeführt werden, wenn die Konsistenz vor und nach der Transaktion eingehalten wird. Wenn wir Geld von einem Bankkonto auf ein anderes transferieren, dann muss es zuerst vom ersten Konto abgezogen werden und dann zum zweiten dazuaddiert werden. Entweder beides wird gemacht oder keines von beiden.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.
- Dabei müssen natürlich die Korrektheit und Integrität der Daten immer gewährleistet sein.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.
- Dabei müssen natürlich die Korrektheit und Integrität der Daten immer gewährleistet sein.
- **Isolation**: Auch wenn mehrere Transaktionen aus jeweils mehreren Schritten gleichzeitig stattfinden, darf zu keiner Zeit die Integrität und Korrektheit der Daten verletzt werden.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.
- Dabei müssen natürlich die Korrektheit und Integrität der Daten immer gewährleistet sein.
- **Isolation**: Auch wenn mehrere Transaktionen aus jeweils mehreren Schritten gleichzeitig stattfinden, darf zu keiner Zeit die Integrität und Korrektheit der Daten verletzt werden. Die Transaktionen werden voneinander isoliert.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.
- Dabei müssen natürlich die Korrektheit und Integrität der Daten immer gewährleistet sein.
- **Isolation**: Auch wenn mehrere Transaktionen aus jeweils mehreren Schritten gleichzeitig stattfinden, darf zu keiner Zeit die Integrität und Korrektheit der Daten verletzt werden. Die Transaktionen werden voneinander isoliert.
- Wenn in unserer Bankdatenbank Geld von Konto A zu Konto B transferiert wird und *gleichzeitig* Geld von Konto B to Konto C , dann darf keine der beiden Transaktionen die andere beeinflussen.

Gleichzeitigkeit und Isolierung



- Ein DBMS muss es mehreren Nutzern und Anwendungen erlauben, gleichzeitig auf die Datenbank zuzugreifen.
- Dabei müssen natürlich die Korrektheit und Integrität der Daten immer gewährleistet sein.
- **Isolation**: Auch wenn mehrere Transaktionen aus jeweils mehreren Schritten gleichzeitig stattfinden, darf zu keiner Zeit die Integrität und Korrektheit der Daten verletzt werden. Die Transaktionen werden voneinander isoliert.
- Wenn in unserer Bankdatenbank Geld von Konto A zu Konto B transferiert wird und *gleichzeitig* Geld von Konto B to Konto C , dann darf keine der beiden Transaktionen die andere beeinflussen.
- Das muss selbst dann funktionieren, wenn eine Datenbank über mehrere Computer / im Netzwerk verteilt aufgebaut ist.



Dauerhaftigkeit und Sicherheit (Safety)



Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.
- Dies beinhaltet die Möglichkeit, Checkpoints zu erstellen und Daten beim Neustart wiederherzustellen.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.
- Dies beinhaltet die Möglichkeit, Checkpoints zu erstellen und Daten beim Neustart wiederherzustellen.
- Natürlich hat das Grenzen

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.
- Dies beinhaltet die Möglichkeit, Checkpoints zu erstellen und Daten beim Neustart wiederherzustellen.
- Natürlich hat das Grenzen: Kein DBMS kann uns irgendwie schützen, wenn Datenträger physisch zerstört werden.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.
- Dies beinhaltet die Möglichkeit, Checkpoints zu erstellen und Daten beim Neustart wiederherzustellen.
- Natürlich hat das Grenzen: Kein DBMS kann uns irgendwie schützen, wenn Datenträger physisch zerstört werden.
- Es sollte also möglich sein, regelmäßig Backups (Sicherheitskopien) der Daten zu erstellen und auch wieder einzuspielen.

Dauerhaftigkeit und Sicherheit (Safety)



- **Dauerhaftigkeit:** Sobald eine Transaktion erfolgreich abgeschlossen (committed) ist, sind die vorgenommenen Änderungen *dauerhaft* gespeichert.
- Darüberhinaus sollte ein DBMS auch sicherstellen, dass alle Daten auch in unvorhergesehenen Situationen gesichert sind.
- Dauerhaftigkeit sollte auch bei Systemabstürzen und Hardwareausfällen – soweit physisch möglich – sichergestellt werden.
- Dies beinhaltet die Möglichkeit, Checkpoints zu erstellen und Daten beim Neustart wiederherzustellen.
- Natürlich hat das Grenzen: Kein DBMS kann uns irgendwie schützen, wenn Datenträger physisch zerstört werden.
- Es sollte also möglich sein, regelmäßig Backups (Sicherheitskopien) der Daten zu erstellen und auch wieder einzuspielen.
- Dies wäre für unsere Bank sehr sehr wichtig.

ACID



- Damit kennen wir die vier sogenannten **ACID properties**^{28,71}

- Damit kennen wir die vier sogenannten **ACID properties**^{28,71}
 1. **Atomizität** (EN: *atomicity*)
 2. **Konsistenz** (EN: *consistency*)
 3. **Isolation** (EN: *isolation*)
 4. **Dauerhaftigkeit** (EN: *durability*)



Datenschutz und Datensicherheit (Security)



Datenschutz und Datensicherheit (Security)



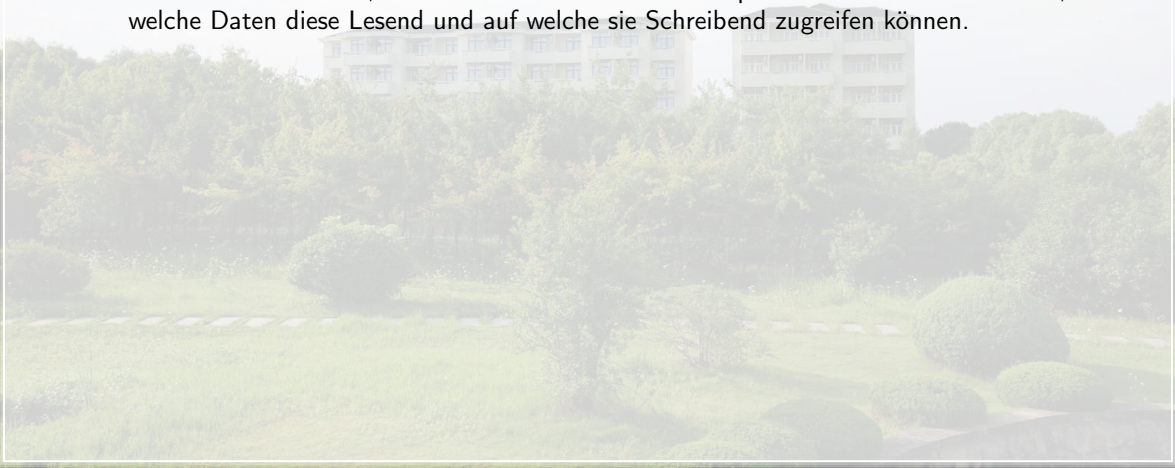
- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.



Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.



Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.

Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.
 - Ein Bankangestellter kann auf die Konten seiner Kunden zugreifen, nicht jedoch auf andere Konten und auch nicht auf die Daten anderer Mitarbeiter.

Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.
 - Ein Bankangestellter kann auf die Konten seiner Kunden zugreifen, nicht jedoch auf andere Konten und auch nicht auf die Daten anderer Mitarbeiter.
 - Ein Mitarbeiter unserer HR-Abteilung kann auf die Mitarbeiterdaten zugreifen, nicht jedoch auf Bankkonten.

Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.
 - Ein Bankangestellter kann auf die Konten seiner Kunden zugreifen, nicht jedoch auf andere Konten und auch nicht auf die Daten anderer Mitarbeiter.
 - Ein Mitarbeiter unserer HR-Abteilung kann auf die Mitarbeiterdaten zugreifen, nicht jedoch auf Bankkonten.
 - Dies geht Hand-in-Hand mit der Drei-Schema-Architektur, die es uns erlaubt, verschiedene Sichten auf die Daten zu definieren.

Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.
 - Ein Bankangestellter kann auf die Konten seiner Kunden zugreifen, nicht jedoch auf andere Konten und auch nicht auf die Daten anderer Mitarbeiter.
 - Ein Mitarbeiter unserer HR-Abteilung kann auf die Mitarbeiterdaten zugreifen, nicht jedoch auf Bankkonten.
 - Dies geht Hand-in-Hand mit der Drei-Schema-Architektur, die es uns erlaubt, verschiedene Sichten auf die Daten zu definieren.
- Weiterhin muss ein DBMS Dinge wie Passwort-Schutz und verschlüsselte Kommunikation implementieren.

Datenschutz und Datensicherheit (Security)



- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
 - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen können.
 - Ein Bankkunde kann nur auf sein eigenes Konto zugreifen.
 - Ein Bankangestellter kann auf die Konten seiner Kunden zugreifen, nicht jedoch auf andere Konten und auch nicht auf die Daten anderer Mitarbeiter.
 - Ein Mitarbeiter unserer HR-Abteilung kann auf die Mitarbeiterdaten zugreifen, nicht jedoch auf Bankkonten.
 - Dies geht Hand-in-Hand mit der Drei-Schema-Architektur, die es uns erlaubt, verschiedene Sichten auf die Daten zu definieren.
- Weiterhin muss ein DBMS Dinge wie Passwort-Schutz und verschlüsselte Kommunikation implementieren.
- Es muss uns auch ermöglichen, später festzustellen, welcher Nutzer wann auf welche Daten zugegriffen und diese verändert hat.



Zusammenfassung



- Nun kennen wir einige der Anforderungen, die wir an ein DBMS stellen würden.

Zusammenfassung



- Nun kennen wir einige der Anforderungen, die wir an ein DBMS stellen würden.
- Es sind ziemlich viele.

Zusammenfassung



- Nun kennen wir einige der Anforderungen, die wir an ein DBMS stellen würden.
- Es sind ziemlich viele.
- Daher sind DBMSse sehr komplexe Systeme.

- Nun kennen wir einige der Anforderungen, die wir an ein DBMS stellen würden.
- Es sind ziemlich viele.
- Daher sind DBMSe sehr komplexe Systeme.
- Es ist auch klar, warum wir diese Anforderungen nicht mit Programmen wie Microsoft Excel oder LibreOffice Calc erfüllen können.

Zusammenfassung



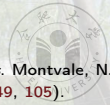
- Nun kennen wir einige der Anforderungen, die wir an ein DBMS stellen würden.
- Es sind ziemlich viele.
- Daher sind DBMSe sehr komplexe Systeme.
- Es ist auch klar, warum wir diese Anforderungen nicht mit Programmen wie Microsoft Excel oder LibreOffice Calc erfüllen können.
- Zum Glück gibt es viele spannende Dinge zu lernen.



谢谢你们！
Thank you!
Vielen Dank!



References I



- [1] Database Management Systems ANSI/X3/SPARC Study Group. *Framework Report on Database Management Systems*. Montvale, NJ, USA: American Federation of Information Processing Societies (AFIPS) Press, 1978. Also published as⁶⁷ (siehe S. 41–49, 105).
- [2] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume I: Introduction to SQL Queries*. Tetras Publishing, Okt. 2018. ISBN: 978-1-9996172-4-0. See also³ (siehe S. 99, 109).
- [3] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume II: In-Depth Querying*. Tetras Publishing, Okt. 2018. ISBN: 978-1-9996172-5-7. See also² (siehe S. 99, 109).
- [4] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (siehe S. 109, 111).
- [5] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Okt. 2019. ISBN: 978-1-4842-5514-8 (siehe S. 109).
- [6] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 109).
- [7] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) und Newsgroups: alt.hypertext, 6. Aug. 1991. URL: <https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt> (besucht am 2025-02-05) (siehe S. 111).
- [8] Alex Berson. *Client/Server Architecture*. 2. Aufl. Computer Communications Series. New York, NY, USA: McGraw-Hill, 29. März 1996. ISBN: 978-0-07-005664-0 (siehe S. 108).
- [9] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (siehe S. 109).
- [10] Silvia Botros und Jeremy Tinley. *High Performance MySQL*. 4. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Nov. 2021. ISBN: 978-1-4920-8051-0 (siehe S. 110).
- [11] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 110).

References II



- [12] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 108).
- [13] Thomas Burns, Elizabeth N. Fong, David Jefferson, Richard Knox, Leo Mark, Christopher Reedy, Louis Reich, Nick Roussopoulos und Walter Truszkowski. "Reference Model for DBMS Standardization, Database Architecture Framework Task Group (DAFTG) of the ANSI/X3/SPARC Database System Study Group". *ACM SIGMOD Record* 15(1):19–58, Mai 1985–März 1986. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0163-5808. doi:10.1145/16342.16343 (siehe S. 41–49).
- [14] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 109, 110).
- [15] 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> (besucht am 2025-01-09) (siehe S. 111).
- [16] 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 (siehe S. 109).
- [17] David Clinton und Christopher Negus. *Ubuntu Linux Bible*. 10. Aufl. Bible Series. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 10. Nov. 2020. ISBN: 978-1-119-72233-5 (siehe S. 111).
- [18] Edgar Frank „Ted“ Codd. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM (CACM)* 13(6):377–387, Juni 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> (besucht am 2025-01-05) (siehe S. 110).
- [19] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 111).
- [20] Matt David und Blake Barnhill. *How to Teach People SQL*. San Francisco, CA, USA: The Data School, Chart.io, Inc., 10. Dez. 2019–10. Apr. 2023. URL: <https://dataschool.com/how-to-teach-people-sql> (besucht am 2025-02-27) (siehe S. 111).
- [21] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 111).

References III



- [22] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide WebW[?]: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: **978-0-13-299045-5** (siehe S. 111).
- [23] Pooyan Doozandeh und Frank E. Ritter. "Some Tips for Academic Writing and Using Microsoft Word". *XRDS: Crossroads, The ACM Magazine for Students* 26(1):10–11, Herbst 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: **1528-4972**. doi:**10.1145/3351470** (siehe S. 110).
- [24] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2015. ISBN: **978-1-4493-6290-4** (siehe S. 109, 110).
- [25] Steve Fanning, Vasudev Narayanan, „flywire“, Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffrers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen und 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> (besucht am 2025-01-13) (siehe S. 109).
- [26] Luca Ferrari und Enrico Pirozzi. *Learn PostgreSQL*. 2. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2023. ISBN: **978-1-83763-564-1** (siehe S. 110).
- [27] Jonas Gamalielsson und 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*. 10.–13. Sep. 2012, Hammamet, Tunisia. Hrsg. von Imed Hammouda, Björn Lundell, Tommi Mikkonen und Walt Scacchi. Bd. 378. IFIP Advances in Information and Communication Technology (IFIPACT). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2012, S. 29–47. ISSN: **1868-4238**. ISBN: **978-3-642-33441-2**. doi:**10.1007/978-3-642-33442-9_3** (siehe S. 109).
- [28] Jim Gray und Andreas Reuter. *Transaction Processing: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Sep. 1992. ISBN: **978-1-55860-190-1** (siehe S. 81, 82).
- [29] Dawn Griffiths. *Excel Cookbook – Receipts for Mastering Microsoft Excel*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2024. ISBN: **978-1-0981-4332-9** (siehe S. 109).

References IV



- [30] Terry Halpin und Tony Morgan. *Information Modeling and Relational Databases*. 3. Aufl. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Juli 2024. ISBN: **978-0-443-23791-1** (siehe S. 110).
- [31] Jan L. Harrington. *Relational Database Design and Implementation*. 4. Aufl. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Apr. 2016. ISBN: **978-0-12-849902-3** (siehe S. 110).
- [32] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: **978-1-0981-0894-6** (siehe S. 109).
- [33] Matthew Helmke. *Ubuntu Linux Unleashed 2021 Edition*. 14. Aufl. Reading, MA, USA: Addison-Wesley Professional, Aug. 2020. ISBN: **978-0-13-668539-5** (siehe S. 109, 111).
- [34] Manuel Hoffmann, Frank Nagle und Yanuo Zhou. *The Value of Open Source Software*. Working Paper 24-038. Boston, MA, USA: Harvard Business School, 1. Jan. 2024. URL: https://www.hbs.edu/ris/Publication%20Files/24-038_51f8444f-502c-4139-8bf2-56eb4b65c58a.pdf (besucht am 2025-06-04) (siehe S. 110).
- [35] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: **978-3-031-35121-1**. doi:10.1007/978-3-031-35122-8 (siehe S. 110).
- [36] *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) und International Electrotechnical Commission (IEC), Juni 2023. URL: [https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_\(en\).zip](https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip) (besucht am 2025-01-08). Consists of several parts, see <https://modern-sql.com/standard> for information where to obtain them. (Siehe S. 111).
- [37] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: **978-1-80323-424-3** (siehe S. 110).
- [38] Joan Lambert und Curtis Frye. *Microsoft Office Step by Step (Office 2021 and Microsoft 365)*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Juni 2022. ISBN: **978-0-13-754493-6** (siehe S. 109).

References V



- [39] Kent D. Lee und 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 (siehe S. 110).
- [40] LibreOffice – The Document Foundation. Berlin, Germany: The Document Foundation, 2024. URL: <https://www.libreoffice.org> (besucht am 2024-12-12) (siehe S. 109).
- [41] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra und William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 3. Jan. 2025. URL: <https://www.britannica.com/technology/client-server-architecture> (besucht am 2025-01-20) (siehe S. 108).
- [42] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 110).
- [43] MariaDB Server Documentation. Milpitas, CA, USA: MariaDB, 2025. URL: <https://mariadb.com/kb/en/documentation> (besucht am 2025-04-24) (siehe S. 109).
- [44] Ron McFadyen und Cindy Miller. *Relational Databases and Microsoft Access*. 3. Aufl. Palatine, IL, USA: Harper College, 2014–2019. URL: <https://harpercollege.pressbooks.pub/relationaldatabases> (besucht am 2025-04-11) (siehe S. 109).
- [45] Jim Melton und 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, Juni 2001. ISBN: 978-1-55860-456-8 (siehe S. 111).
- [46] Microsoft Word. Redmond, WA, USA: Microsoft Corporation, 2024. URL: <https://www.microsoft.com/en-us/microsoft-365/word> (besucht am 2024-12-12) (siehe S. 110).
- [47] Cameron Newham und Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (siehe S. 108).
- [48] Regina O. Obe und Leo S. Hsu. *PostgreSQL: Up and Running*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Okt. 2017. ISBN: 978-1-4919-6336-4 (siehe S. 110).
- [49] Robert Orfali, Dan Harkey und Jeri Edwards. *Client/Server Survival Guide*. 3. Aufl. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 25. Jan. 1999. ISBN: 978-0-471-31615-2 (siehe S. 108).

References VI



- [50] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglén, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin und Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), 14. Juli 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: **1553-7358**. doi:[10.1371/JOURNAL.PCBI.1004947](https://doi.org/10.1371/JOURNAL.PCBI.1004947) (siehe S. **108**).
- [51] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. **110**).
- [52] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: **978-1-78883-546-6** (siehe S. **108**).
- [53] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: **978-1-78398-154-0** (siehe S. **109**).
- [54] Mike Reichardt, Michael Gundall und 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)*. 13.–15. Okt. 2021, Toronto, ON, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2021, S. 1–8. ISSN: **2577-1647**. ISBN: **978-1-6654-3554-3**. doi:[10.1109/IECON48115.2021.9589382](https://doi.org/10.1109/IECON48115.2021.9589382) (siehe S. **110**).
- [55] Mark Richards und Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: **978-1-4920-4345-4** (siehe S. **108**).
- [56] Heinz Schweppe und Manuel Scholz. "Introduction". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 1. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs045-01-Intro-2.pdf> (besucht am 2025-01-08) (siehe S. **41–49**).
- [57] 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** (siehe S. **109**).
- [58] Ellen Siever, Stephen Figgins, Robert Love und Arnold Robbins. *Linux in a Nutshell*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2009. ISBN: **978-0-596-15448-6** (siehe S. **109**).
- [59] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: **978-1-0981-3391-7** (siehe S. **108**).

References VII



- [60] John Miles Smith und Philip Yen-Tang Chang. "Optimizing the Performance of a Relational Algebra Database Interface". *Communications of the ACM (CACM)* 18(10):568–579, Okt. 1975. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/361020.361025 (siehe S. 110).
- [61] "SQL Commands". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), 20. Feb. 2025. Kap. Part VI. Reference. URL: <https://www.postgresql.org/docs/17/sql-commands.html> (besucht am 2025-02-25) (siehe S. 111).
- [62] Ryan K. Stephens und Ronald R. Plew. *Sams Teach Yourself SQL in 21 Days*. 4. Aufl. Sams Tech Yourself. Indianapolis, IN, USA: SAMS Technical Publishing und Hoboken, NJ, USA: Pearson Education, Inc., Okt. 2002. ISBN: 978-0-672-32451-2 (siehe S. 105, 111).
- [63] Ryan K. Stephens, Ronald R. Plew, Bryan Morgan und Jeff Perkins. *SQL in 21 Tagen. Die Datenbank-Abfragesprache SQL vollständig erklärt (in 14/21 Tagen)*. 6. Aufl. Burghann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of⁶² (siehe S. 111).
- [64] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 110, 111).
- [65] Alkin Tezuysal und Ibrar Ahmed. *Database Design and Modeling with PostgreSQL and MySQL*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2024. ISBN: 978-1-80323-347-5 (siehe S. 110).
- [66] 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 (siehe S. 109).
- [67] Dennis Tsichritzis und Anthony Klug. "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems". *Information Systems: Databases: Their Creation, Management and Utilization* 3(3):173–191, 1978. Oxford, Oxfordshire, England, UK: Pergamon Press Ltd., now Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0306-4379. Also published as¹ (siehe S. 41–49, 99).
- [68] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0215-7 (siehe S. 108, 111).

References VIII



- [69] Laurie A. Ulrich und Ken Cook. *Access For Dummies*. Hoboken, NJ, USA: For Dummies (Wiley), Dez. 2021. ISBN: **978-1-119-82908-9** (siehe S. **109**).
- [70] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: **978-0-13-792931-3** (siehe S. **109**).
- [71] Gerhard Weikum und Gottfried Vossen. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Juni 2001. ISBN: **978-1-55860-508-4** (siehe S. **81, 82**).
- [72] 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), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: <https://thomasweise.github.io/databases> (besucht am 2025-01-05) (siehe S. **108–110**).
- [73] 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), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: <https://thomasweise.github.io/programmingWithPython> (besucht am 2025-01-05) (siehe S. **110**).
- [74] *What is a Relational Database?* Armonk, NY, USA: International Business Machines Corporation (IBM), 20. Okt. 2021–12. Dez. 2024. URL: <https://www.ibm.com/think/topics/relational-databases> (besucht am 2025-01-05) (siehe S. **110**).
- [75] Ulf Michael „Monty“ Widenius, David Axmark und Uppsala, Sweden: MySQL AB. *MySQL Reference Manual – Documentation from the Source*. Sebastopol, CA, USA: O'Reilly Media, Inc., 9. Juli 2002. ISBN: **978-0-596-00265-7** (siehe S. **110**).
- [76] Kinza Yasar und Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., Juni 2024. URL: <https://www.techtarget.com/searchdatamanagement/definition/database-management-system> (besucht am 2025-01-11) (siehe S. **108**).

References IX



- [77] Pavlo V. Zahorodko und 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)*. 18. Dez. 2021, Virtual Event and Kryvyi Rih, Ukraine. Hrsg. von Arnold E. Kiv, Serhiy O. Semerikov, Vladimir N. Soloviev und Andrii M. Striuk. Bd. 3077 der Reihe CEUR Workshop Proceedings ([CEUR-WS.org](https://ceur-ws.org)). Aachen, Nordrhein-Westfalen, Germany: CEUR-WS Team, Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen, 2022, S. 66–82. ISSN: 1613-0073. URL: <https://ceur-ws.org/Vol-3077/paper12.pdf> (besucht am 2025-10-04) (siehe S. 109, 110).
- [78] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 108).
- [79] 公民身份号码 (*Citizen Identification Number*). 中华人民共和国国家标准 (National Standard of the People's Republic of China, GB) GB11643-1999. China, Beijing (中国北京市): 中华人民共和国国家质量监督检验检疫总局 (General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China), 中国国家标准化管理委员会 (Standardization Administration of the People's Republic of China, SAC) und 中国标准出版社 (Standards Press of China), 19. Jan.–3. Nov. 1999. URL: <https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=080D6FBF2BB468F9007657F26D60013E> (besucht am 2024-07-26) (siehe S. 20–25).

Glossary (in English) I



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^{12,47,78}. 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 database management systems (DBMSes), such as `psql`.

client-server architecture is a system design where a central server receives requests from one or multiple clients^{8,41,49,52,55}. 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 database (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*⁷².

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 DB⁷⁶.

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.

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^{59,68}. Learn more at <https://git-scm.com>.

GitHub is a website where software projects can be hosted and managed via the Git VCS^{50,68}. Learn more at <https://github.com>.

HR human resources department

Glossary (in English) II



IT information technology

- LAMP Stack** A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP^{14,33}.
- LibreOffice** is an open source office suite^{27,40,57} which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See⁷² 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^{25,57}. It is part of LibreOffice^{27,40,57} and has functionality that is comparable to Microsoft Access^{6,16,69}.
- 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 spreadsheet software^{40,57}, i.e., an alternative to Microsoft Excel. It is part of LibreOffice^{27,40,57}.
- LibreOffice Writer** is a free and open source text writing program⁷⁷ and part of LibreOffice^{27,40,57}. It is a good alternative to Microsoft Word.
- Linux** is the leading open source operating system, i.e., a free alternative for Microsoft Windows^{4,32,58,66,70}. 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^{2,3,5,24,43,53}. 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^{6,16,44,69}. 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^{9,29,38}. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc^{40,57}.
- Microsoft Office** is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access³⁸. LibreOffice is a free and open source alternative.

Glossary (in English) III



- Microsoft Windows is a commercial proprietary operating system¹¹. 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^{23,46,77} and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.
- MySQL An open source relational database management system^{10,24,54,65,75}. MySQL is famous for its use in the LAMP Stack. See <https://www.mysql.com> for more information.
- OSS Open source software, i.e., software that can freely be used, whose source code is made available in the internet, and which is usually developed cooperatively over the internet as well³⁴. Typical examples are Python, Linux, Git, and PostgreSQL.
- PostgreSQL An open source object-relational DBMS^{26,48,51,65}. See <https://postgresql.org> for more information.
- psql is the client program used to access the PostgreSQL DBMS server.
- Python The Python programming language^{35,39,42,73}, i.e., what you will learn about in our book⁷³. 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^{18,30,31,60,64,72,74}.
- server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carrying 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 servers¹⁴ 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“³⁷.

Glossary (in English) IV



SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases^{15,19–21,36,45,61–64}. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference⁶¹.

terminal A terminal is a text-based window where you can enter commands and execute them^{4,17}. 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 Druck auf  + , dann Schreiben von `cmd`, dann Druck auf . Under Ubuntu Linux,  +  +  opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux^{17,33}. 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>.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code⁶⁸. 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^{7,22}