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

Version: 2025-06-18

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-Kode 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

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.

- 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.

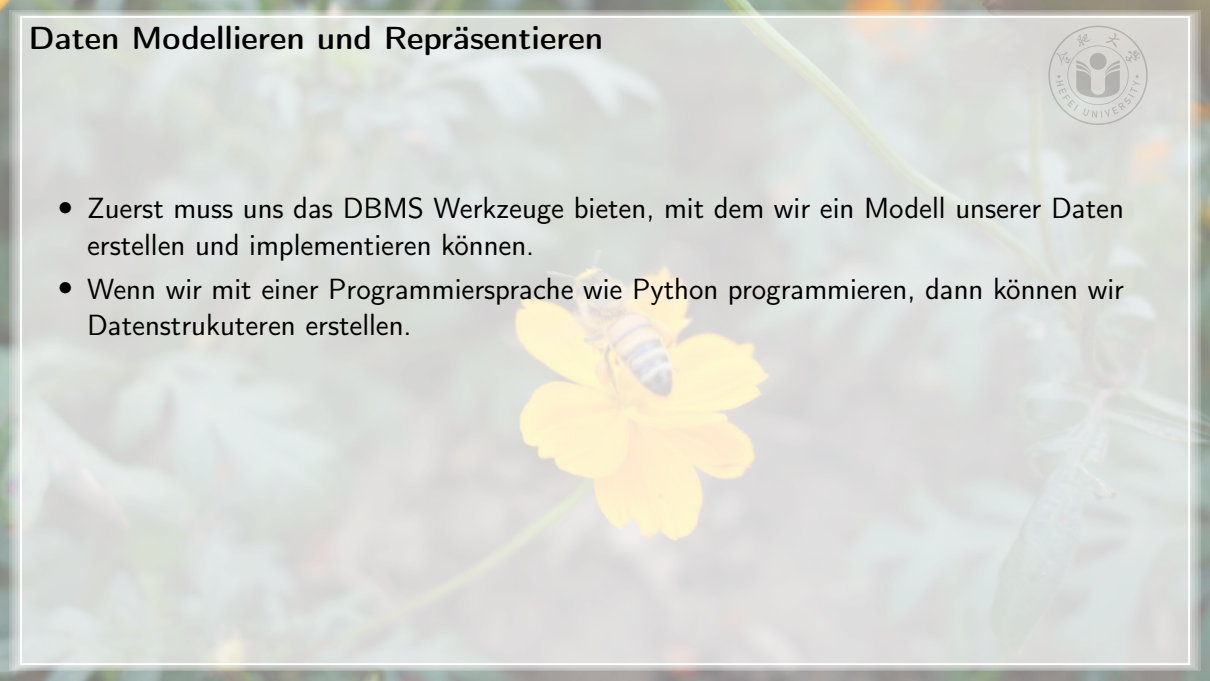
- Stellen wir uns dafür vor, dass wir eine Datenbank für eine Bank entwickeln wollen.
- Datenbank soll die Informationen über die Kunden, ihre Konten, und Transaktionen speichern, sowie die Daten über die Angestellten der Bank.
- Davon ausgehend bauen wir uns eine Wunschliste von Features und Anforderungen 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 Datenstrukuteren 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 Datenstrukuteren 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 Datenstrukuteren 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 für die Integrität der Daten definieren, z.B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)[7] 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 für die Integrität der Daten defininieren, z.B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)[7] 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 das 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 für die Integrität der Daten defininieren, z.B. daß Ausweisnummern dem Standard für Chinesische Identifikationsnummern (中国公民身份号码)[7] 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 das 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 *phyische 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 *phyische 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 *phyische 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

Unabhängikeiten

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

Unabhängikeiten

- 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 seien sollen.

Unabhängikeiten

- 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 seien sollen.
- Das ist wie beim Programmieren mit einer Sprache wie Python

Unabhängikeiten

- 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 seien 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 Maschinenkode repräsentiert, der völlig anders aussehen kann

Unabhängikeiten

- 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 seien 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 Maschinenkode repräsentiert, der völlig anders aussehen kann. . . was der Pythonprogrammierer aber nicht wissen muss.

Unabhängikeiten

- 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 seien 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 Maschinenkode 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.

Unabhängikeiten

- 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 seien 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 Maschinenkode 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 phyischen Schema, davon, die die Daten auf der Festplatte organisiert sind.

Unabhängikeiten

- 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 seien 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 Maschinenkode 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 phyischen Schema, davon, die 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.

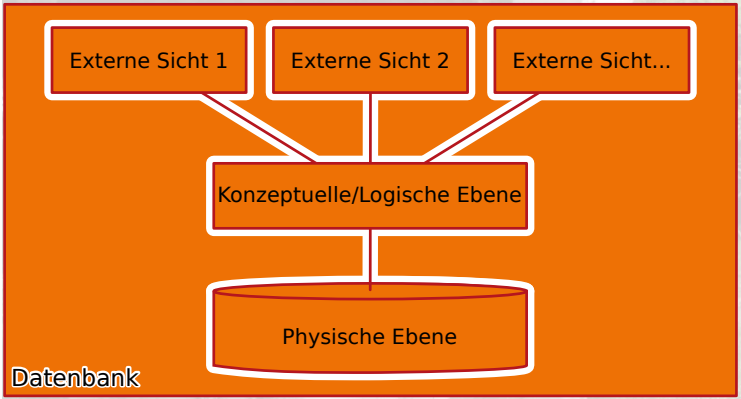
Drei-Schema-Architektur

- 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 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.
- Ein DBMS muss uns erlauben, so eine Drei-Schema-Architektur[1,2,4,5] zu entwickeln, bei der jede Ebene möglichst unabhängig von der darunterliegenden seien 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.

Sources: [1, 2, 4, 5]

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 Schreibgeschwindikeit 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 Schreibgeschwindikeit 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 Schreibgeschwindikeit 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 Schreibgeschwindikeit 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.

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 Schreibgeschwindikeit 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 under Integrität der Daten sicher.

Integrität

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

Integrität

- Ein DBMS stellt die Korrektheit under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte entsteht, das nicht einem Kunden gehört.

Integrität

- Ein DBMS stellt die Korrektheit under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte 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 under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte 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 under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte 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 under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte 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 under 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 Bankkonte immer einem Kunden gehört, dann darf das DBMS nie zulassen, das irgendwie ein Bankkonte 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 transferriern, 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 voneinenader 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 voneinenader isoliert.
- Wenn in unserer Bankdatenbank Geld von Konto A zu Konto B transferriert 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 voneinenader isoliert.
- Wenn in unserer Bankdatenbank Geld von Konto A zu Konto B transferriert 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 Netwerk 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[3,6]

ACID

- Damit kennen wir die vier sogenannten ACID properties[3,6]
  1. Atomizität – Atomicity
  2. Konsistenz – Consistency
  3. Isolation – Isolation
  4. Dauerhaftigkeit – 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)

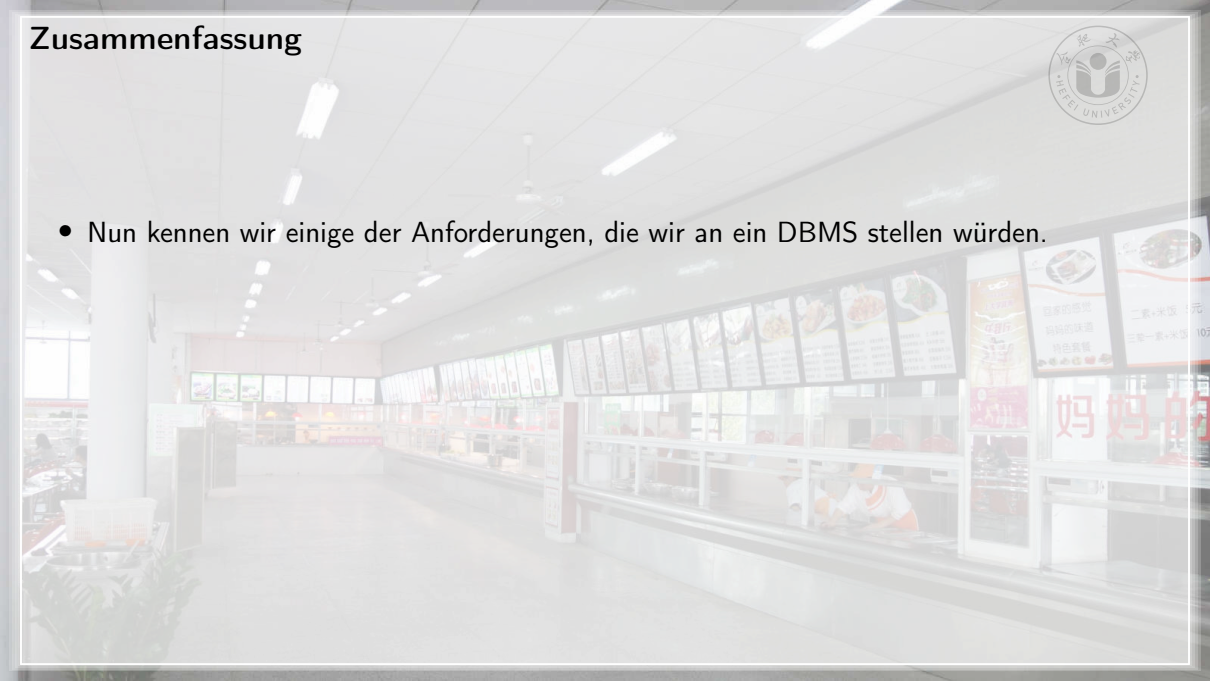
- Ein DBMS muss es uns erlauben, unsere Daten vor unautorisiertem Zugriff sowohl vom Inneren als auch vom Äußeren unserer Organisation zu schützen.
  - Es muss uns erlauben, Rollen und Benutzerkonten zu spezifizieren und zu definieren, auf welche Daten diese Lesend und auf welche sie Schreibend zugreifen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

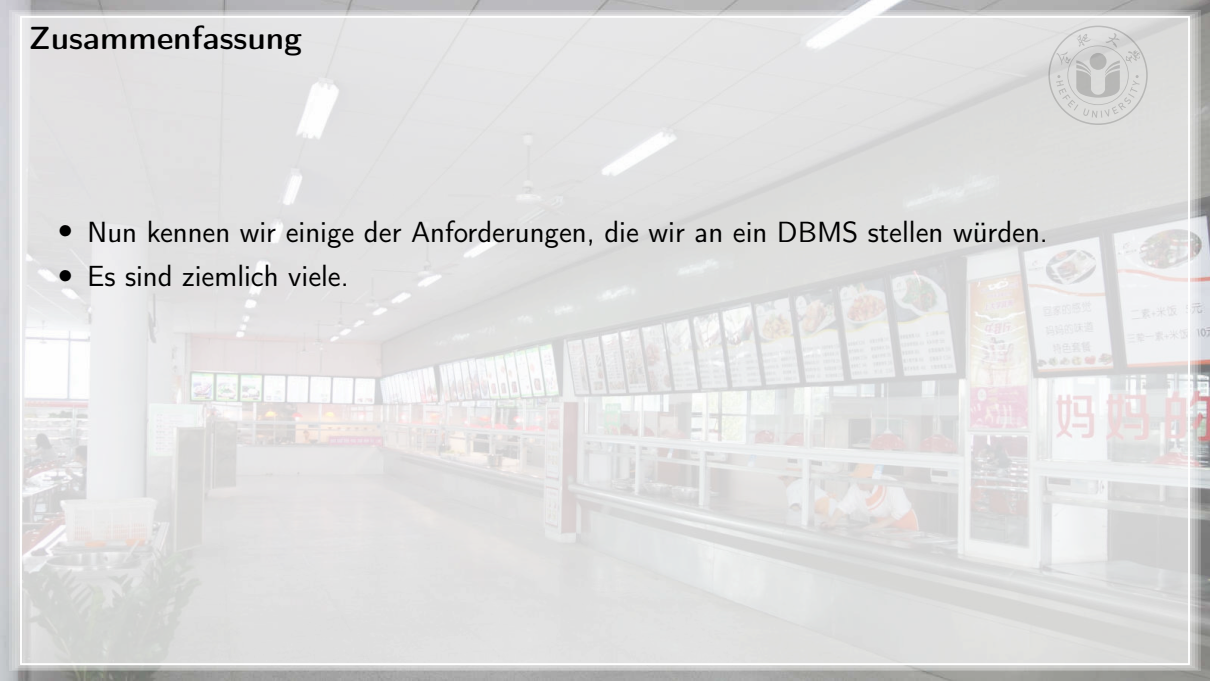
Zusammenfassung

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 DBMSe sehr komplexe Systeme.

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.

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.

谢谢您们！

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 as[5] (siehe S. 39–47, 97).

[2] 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. 39–47).

[3] 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. 79, 80).

[4] 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. 39–47).

[5] 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 as[1] (siehe S. 39–47, 97).

[6] 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. 79, 80).

[7] 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. 18–23).